

Production scheduling for electronic component production using simulation based meta heuristic optimization

Diplomarbeit
von
Christoph Wolfsgruber

Technische Universität Graz

Fakultät für Maschinenbau und Wirtschaftswissenschaften

Institut für Maschinenbau und Betriebsinformatik

Univ.-Prof. Dipl.-Ing. Dr.techn. Siegfried Vössner

Graz, im März 2012

In cooperation with

TDK Deutschlandsberg



STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

.....
date

.....
signature

Abstract

TDK produces multilayer varistors in a complex multi stage production process in Deutschlandsberg. Many different types of these varistors are produced and due to the different properties of these varistors the machining times vary a lot. Therefore, the sequence of the production jobs has a big influence on the in-production inventory.

A production scheduling tool should optimize a weekly production plan with respect to criteria's like a low inventory. This production scheduling tool is using a simulation based meta heuristic optimization. That means that a simulation model is connected to an optimization tool. In this tool the meta heuristic improvement method Simulated Annealing is used, which consists of an acceptance algorithm and a permutation algorithm. Simulated Annealing simulates the physical cooling behavior of solid objects using the Boltzmann probability distribution to accept new sequences (acceptation algorithm). The permutation algorithm randomly generates a new sequence out of a previous production plan with the use of setup clusters. This new sequence gets handed over to the simulation module. Within the discrete event simulation a simplified production process of the multilayer varistor production is modeled. This simulation model calculates several objective values for the new sequence. One main objective is a low in-production inventory level measured with the indicator work in process. This and other objective values get passed over to the optimization tool in which the acceptance algorithm decides whether this new sequence gets the base of a further improvement. This iterative process continues until no further improvement occurs.

The planning horizon of this tool is one week and furthermore the optimization is done offline. That means that the production scheduling tool has no interaction with the real production system. The tool is using a predefined model and the optimization process is done once for the production demand of the upcoming week. The production scheduling tool is developed in Microsoft Excel 2003 Visual Basic for Application.

Through the use of this production scheduling tool the work in process of the multilayer varistor production in Deutschlandsberg could be reduced by 10% at planning level. Additionally, the developed tool offers an overview of the weekly production in a Gantt chart and the process of production scheduling gets standardized.

Kurzfassung

TDK produziert am Standort Deutschlandsberg neben anderen Produkten, Vielschicht-Varistoren in einem komplexen, mehrstufigen Produktionsprozess. Eine große Anzahl an verschiedenen Typen dieser Varistoren wird hergestellt, welche bedingt durch ihre Eigenschaften sehr unterschiedliche Prozesszeiten haben. Daher ist die Reihenfolge der Fertigungsaufträge für den Umlaufbestand sehr entscheidend.

Ein Programm zur Reihenfolgeplanung sollte die Fertigungsaufträge des wöchentlichen Produktionsplans auf Kriterien wie zum Beispiel einen geringen Umlaufbestand optimal anreihen. Dieses Softwareprogramm verwendet eine simulationsbasierte, metaheuristische Optimierung. Das bedeutet, dass ein Simulationsmodell mit einem Optimierungsalgorithmus gekoppelt ist. In diesem Programm wird der metaheuristische Verbesserungsalgorithmus Simulated Annealing verwendet, der aus einem Akzeptanz- und einem Vertauschungsalgorithmus besteht. Simulated Annealing ist dem Abkühlverhalten von Festkörpern nachgebildet und verwendet die Boltzmann Wahrscheinlichkeitsverteilung um neue Reihenfolgen zu akzeptieren (Akzeptanzalgorithmus). Der Vertauschungsalgorithmus generiert, ausgehend von dem hervorgehenden Produktionsplan, mittels Rüstgruppen zufällig neue Reihenfolgen. Danach wird dieser modifizierte Produktionsplan im ereignisorientierten Simulationsmodell simuliert und dabei werden diverse Zielwerte berechnet. Das Simulationsmodell bildet hierbei den vereinfachten Prozess der Vielschicht-Varistoren-Produktion am Standort Deutschlandsberg nach. Anschließend entscheidet der Akzeptanzalgorithmus anhand der Zielwerte, ob der neu modifizierte Produktionsplan die Basis für eine weitere Optimierung wird oder nicht. Dieser iterative Vorgang läuft so lange, bis keine weitere Verbesserung mehr erfolgt. Als Zielwerte dienen hierbei Indikatoren, wie der Umlaufbestand, um einen kontinuierlichen Produktionsfluss zu ermöglichen.

Der Planungshorizont dieses Programms ist eine Woche und des Weiteren verläuft die Optimierung offline. Das heißt, dass das Programm über keine Information des Ist-Zustands der Produktion verfügt. Das Programm wurde in Microsoft Excel 2003 Visual Basic for Application entwickelt. Durch die Verwendung des Programms konnte der Umlaufbestand der Fertigung auf Planungsebene um 10% verringert werden. Das entwickelte Programm zeigt in Form eines Gantt Diagramms eine übersichtliche Darstellung der Wochenproduktion. Zudem wird der Prozess der Reihenfolgeplanung objektiviert und vereinfacht.

Foreword

This diploma thesis was written during the time-period from October 2011 until March 2012, under the supervision of Univ.-Prof. Dipl.-Ing. Dr.techn. Siegfried Vössner and Ass.Prof. Dipl.-Ing. Dr.techn. Gerald Lichtenegger at the Graz University of Technology. The company partner of this diploma thesis is the electronic component manufacturer TDK. The intent of the thesis is to develop a software tool for the production scheduling of the multilayer varistor production in the production frontend in Deutschlandsberg, Austria.

I want to thank my supervisors from the company TDK Dipl.-Ing. Martin Thuy and Dipl.-Phys. Bernd Aigner, as well as my supervisors from the university Univ.-Prof. Dipl.-Ing. Dr.techn. Siegfried Vössner and Ass.Prof. Dipl.-Ing. Dr.techn. Gerald Lichtenegger, for their great help during the development of this thesis. Furthermore, I also want to thank my friend Philipp Thaler for his knowledge support during this diploma thesis and also Martin Führer and Robert Nini for their support during my Production Science and Management study at the Graz University of Technology. Moreover, I want to thank my parents for their financial support during my studies.

Table of Content

1	Introduction	1
1.1	Initial situation	1
1.2	Goals	1
1.3	Problem definition	1
1.4	Approach	1
1.5	Structure of this thesis	2
2	Production logistic	3
2.1	Logistic target values	4
2.1.1	Logistic output performance	4
2.1.2	Logistic costs	6
2.2	Dilemma of the production logistic	9
2.2.1	Operations planning dilemma	9
2.2.2	Disposition dilemma	10
2.2.3	Target dilemma of the production	10
3	Production planning	11
3.1	Core tasks of production planning	12
3.1.1	Production program planning	12
3.1.2	Production requirements planning	13
3.1.3	In-house production planning	13
4	Models and methods of production scheduling	16
4.1	Definitions	16
4.2	Advanced lot sizing models	17
4.3	Modeling of sequence problems	18
4.3.1	Mixed integer linear programming	18
4.3.2	Graph theoretical modeling	19
4.3.3	Discrete-event simulation	20
4.4	Optimization methods	22
4.4.1	Systematic search	22
4.4.2	Heuristic search	22
4.4.3	Meta heuristic improvement methods	23
4.5	Optimization software and tools	27
4.5.1	Software for systematic search	27
4.5.2	Software for heuristic search	27
4.5.3	Self made software	28

5	Production process of multilayer varistors	29
5.1	Product description	29
5.2	Production process	30
5.2.1	Overview	30
5.2.2	Main frontend processes	32
5.3	Summary of the frontend production processes	38
5.4	Simplification of the frontend production processes	39
6	Production scheduling tool	41
6.1	Simulation based optimization	42
6.2	Simulation	43
6.2.1	Core simulation modules	44
6.2.2	Discrete-event simulation	47
6.2.3	Implemented simulation model	50
6.3	Optimization	53
6.3.1	Acceptance algorithm	53
6.3.2	Permutation algorithm	55
6.4	Objective value	57
6.4.1	Implemented objective value	59
6.5	Graphical user interface	61
6.5.1	Input	61
6.5.2	Output	61
6.6	Testing results	65
7	Perspective	67
7.1	Online production scheduling	67
7.2	Case Studies	69
7.2.1	Simulation based optimization in semiconductor production	69
7.2.2	Production control using simulation based heuristic optimization	69
7.2.3	Simulation based scheduling optimization	70
8	Summary	71
	Bibliography	72
	List of Figures	75
	List of Tables	77
	List of Equations	78
	List of Algorithms	79
	List of Abbreviations	80
	Attachment	81

1 Introduction

1.1 Initial situation

TDK produces multilayer varistors (MLVs) in Deutschlandsberg. These MLVs are produced in a complex multi stage production process. Furthermore, many different types of these varistors are produced. The products differ in terms of the electronic behavior and the case size. Therefore, the machining times of different products vary a lot. Another big issue is the additional setup costs, which emerge on some machines during the changeover to another product type.

1.2 Goals

The goal of this thesis is to reduce the lead-time within the production in Deutschlandsberg. In recent times TDK was able to reduce the lead-time and therefore also the in process inventory by organizational improvements. However, one big problem is the sequencing of the production, which is done manually. As the production process is complex and interconnected, a human being needs a lot of time and effort for the production scheduling. And still this solution is often not optimal, because its hard to include all influences. Therefore, a software tool should support this process of production scheduling, which is able to take the whole complexity of the production process into account.

1.3 Problem definition

The task of this thesis is to develop an automated production scheduling algorithm to reduce the inventory and thereby also the lead-time. This production scheduling tool should optimize a weekly production plan with respect to a continuous flow in the frontend production process in Deutschlandsberg and also a continuous delivery of the production backend in Kutina, Croatia. Furthermore, the additional emerging costs for setup should be covered in the planning decisions.

1.4 Approach

The approach of the production scheduling tool is a simulation based meta heuristic optimization. For that the tool is using a discrete event simulation model of the simplified production process. The optimization is realized with the meta heuristic improvement method Simulated Annealing. Furthermore, the optimization is done offline. That means that the production scheduling tool has no interaction with the real

production system. The optimization tool is using a predefined model and the optimization process is done once for the production demand of the upcoming week. The production scheduling tool is developed in Microsoft Excel 2003 Visual Basic for Application (VBA).

1.5 Structure of this thesis

First of all this thesis consists of a theoretical part which comes from a literature study and a practical part. The theoretical part describes the objectives and the dilemma of production logistics in Section 2. Furthermore, this dilemma gives also the transition to Section 3, which deals with the tasks of production planning. In the next section different models and methods for production scheduling are explained. Section 5 gives a close description of the production process of MLV and how this process can be simplified for the scheduling tool. In the practical part in Section 6 the basic implementation of the developed production scheduling tool gets explained. In this section also the results and the achieved improvements are represented. Finally, the benefits of a possible online production scheduling and selected case studies get explained in Section 7.

2 Production logistic

The production logistic deals with the planning and the steering as well as the control of material and information flows within the company. Therefore, the production logistic is embedded within the supply chain between procurement logistic and distribution logistic (see Figure 1).

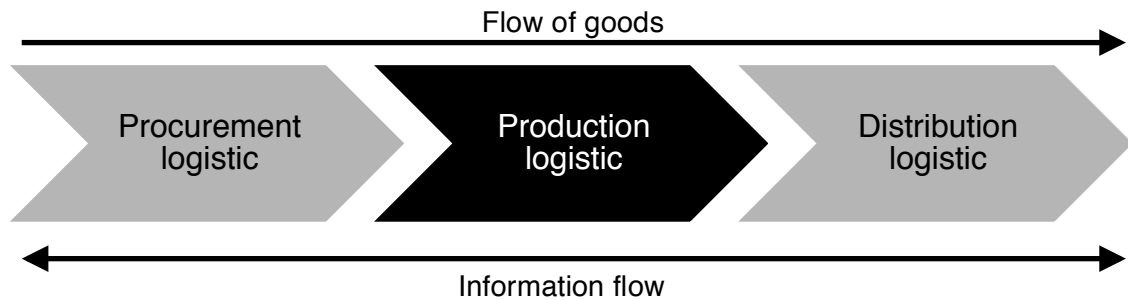


Figure 1: Production logistic within the supply chain [Arnold et al. 2008]

Bauer divides the aims of production logistic into three categories: time, quantity and finances (see Table 1).

Time	Quantity	Finances
Reduce lead-time	Reduce inventory	Increase of return of investment
Compliance of delivery death lines	Increase service level	Increase contribution margin
Increase utilization time	Increase output	Decrease production costs
		Decrease inventory costs
		Increase liquidity

Table 1: Aims of production logistic [Bauer 2011]

In the Section 2.1 the logistic target values are discussed in more detail. Additionally, the conflict between this target values is shown in the Section 2.2 with the dilemma of the production logistic. Thereby, also the aims of production logistic, shown in Table 1, get deducted.

2.1 Logistic target values

Wiendahl is differencing logistic target values into their influence on logistic output performance and logistic costs (see Table 2).

	Logistic output performance	Logistic costs
External	Delivery time	Price
	Delivery date deviation	
	Delivery reliability	
Internal	Lead-time	Inventory
	Date deviation	Performance and utilization
	Date reliability	Default costs

Table 2: Target values of the logistic output performance and costs [Wiendahl 1997]

External logistic target values are customer oriented target values. Delivery time, delivery date deviation and delivery reliability can be combined as logistic output performance. On the other hand, there are logistic costs with the price. The internal logistic target values can be deducted out of the external logistic target values. On the output side this are the lead-time, the date deviation and the date reliability. All three of them are measurable within the production. Internal logistic costs are the inventory, the performance and default costs. This internal logistic costs have an influence on the costs of production and therefore also on the price formation.

2.1.1 Logistic output performance

The logistic output performance is divided into the external logistic target values: delivery time (see Section 2.1.1.1), delivery date deviation and delivery reliability (see Section 2.1.1.2) and internal logistic target values: lead-time (see Section 2.1.1.3), date deviation and reliability (see Section 2.1.1.4).

2.1.1.1 Delivery time

The delivery time is defined as the duration between the order acceptance and the delivery of the order [Wiendahl 1997]. The length of the delivery time depends on the storage strategy. The delivery time has a strategic relevance for the economic success.

Empiric studies show that companies with short delivery times grow faster and have higher profit than companies with long delivery times [Stalk & Hout 1990].

The delivery time is next to price and quality an influencing parameter on the buying decision. If the quality and the price of two offers are the same then the decision is made over the delivery time. Furthermore, a manufacturer can make a delivery time dependent surcharge.

2.1.1.2 Delivery date deviation and reliability

The delivery date deviation identifies the difference between the planned and the actual delivery date [Lödding 2005].

$$LTA = TL_{Actual} - TL_{Plan} \quad (1)$$

with LTA delivery date deviation [days]
 TL delivery date [day]

The delivery reliability identifies the percentage of within the delivery tolerance delivered orders [Lödding 2005].

$$LT = \frac{\sum^{Orders} \text{with } LTA_{lowerLimit} \leq LTA \leq LTA_{upperLimit}}{\sum Orders} \times 100 \quad (2)$$

with LT delivery reliability [%]
 $LTA_{lowerLimit}$ lower limit of allowed delivery date deviation [days]
 $LTA_{upperLimit}$ upper limit of allowed delivery date deviation [days]

The delivery reliability is measured within a defined reference period. Similar to the delivery time the delivery reliability has a high strategic relevance for the business success. According to a study from Deloitte & Touche [1998] about competition factors in the 21th century, delivery reliability has a higher importance than quality, technology and price.

2.1.1.3 Lead-time

The lead-time, also known as throughput time, of a production order is defined as the duration between order release and the end of processing [Wiendahl 1997].

$$TH = TAE - TAB \quad (3)$$

with TH order lead-time [days]
 TAE end of processing [day]
 TAB start of processing [day]

Obviously the lead-time has a high impact on the delivery time, because the shortest possible delivery time cannot be smaller than the lead-time.

2.1.1.4 Date deviation and reliability

The date deviation identifies the deviation of the actual lead-time to the planned lead-time [Lödding 2005].

$$TAA = TAE_{Actual} - TAE_{Plan} \quad (4)$$

with TAA date deviation [days]
 TAE end of processing [day]

This date deviation can be identified for one order. In contrast stands the date reliability, which is a measure of the deviation of the lead-time for all orders within a certain period of time [Lödding 2005].

$$TT = \frac{\sum^{Orders} \text{with } TAA_{lowerLimit} \leq TAA \leq TAA_{upperLimit}}{\sum Orders} \times 100 \quad (5)$$

with TT date reliability [%]
 $TAA_{lowerLimit}$ lower limit of allowed date deviation [days]
 $TAA_{upperLimit}$ upper limit of allowed date deviation [days]

2.1.2 Logistic costs

Like the logistic output performance, the logistic costs also influence the competitiveness of a company. With a given price the profit rises with lower logistic costs [Lödding 2005]. The three main internal logistic costs are: inventory (see Section 2.1.2.1), performance and utilization (see Section 2.1.2.2) and default costs (see Section 2.1.2.3).

2.1.2.1 Inventory

In principle, inventory can be divided into warehouse and production inventory. Inventory includes raw materials, semi-finished and finished goods.

According to Hopp & Spearman [2001] inventory between the start and end points of a product routing is called work in process (WIP) or in process inventory. Since routings starts and end at stock points WIP is the total amount of products being fabricated, waiting in queue or storage, but not including the ending stock points.

John D.C. Little [1961] proved the relationship between the WIP, the throughput time and the expected mean arrival time between two units to the system.

$$WIP = TH - CT \quad (6)$$

with WIP work in process [#]
 TH throughput time [days]
 CT expected mean arrival time between two units [days]

Out of several reasons inventory has a high interest in production logistics: inventory as a target value, inventory as a logistic control variable and inventory as a control variable for the continuous improvement process (CIP) [Lödding 2005].

Inventory as a target value

The inventory influences, through its capital binding effect and furthermore through the costs for the capital binding, the finances of a company. Higher inventory means that a company needs more capital as current assets and therefore has lower financial freedom. Additionally, the company has to carry interest costs for this bonded capital. If it is possible to lower the inventory, the total costs will decrease. As the costs are normally insignificant lower than the revenue, such a cost reduction will lead to an even higher profit increase.

One of the most important financial indicators is the return of investment (ROI), which is ratio of profit to capital employed. An inventory reduction will lead to a higher ROI because of the higher profit and furthermore the lower capital employed [Lödding 2005].

Inventory as logistic control variable

In-process inventory influences on the one hand the utilization and on the other hand also the lead-time. The conflict of objectives between low inventory and short lead-time on the one side and high utilization on the other side are known as the dilemma of production logistic (see Section 2.2) [Lödding 2005].

Inventory as a control variable for the CIP

The inventory is buffering process breakdowns and therefore it is covering disturbance susceptibility of processes. Figure 2 shows the different views on inventory: the historical view and the lean view coming from Japan.

The black peaks indicate various problems and the grey area shows the level of inventory. This visualization can be seen like a sea with rocks. If the level of water is high enough, all rocks are hidden, but a ship still can crash into one of these rocks. By lowering the water level (inventory level) the rocks (problems) appear.

The basic idea behind a CIP is to lower the inventory step by step until problems appear. Then the problems gets analyzed and fixed. Afterwards the inventory is lowered even further until new problems appear (and so on). This CIP leads to controlled processes. As a consequence, the quality is increasing, the production costs go down and also the lead-time decreases. Therefore, many Japanese companies use the level of inventory as an indicator of process control [Monden 1981].

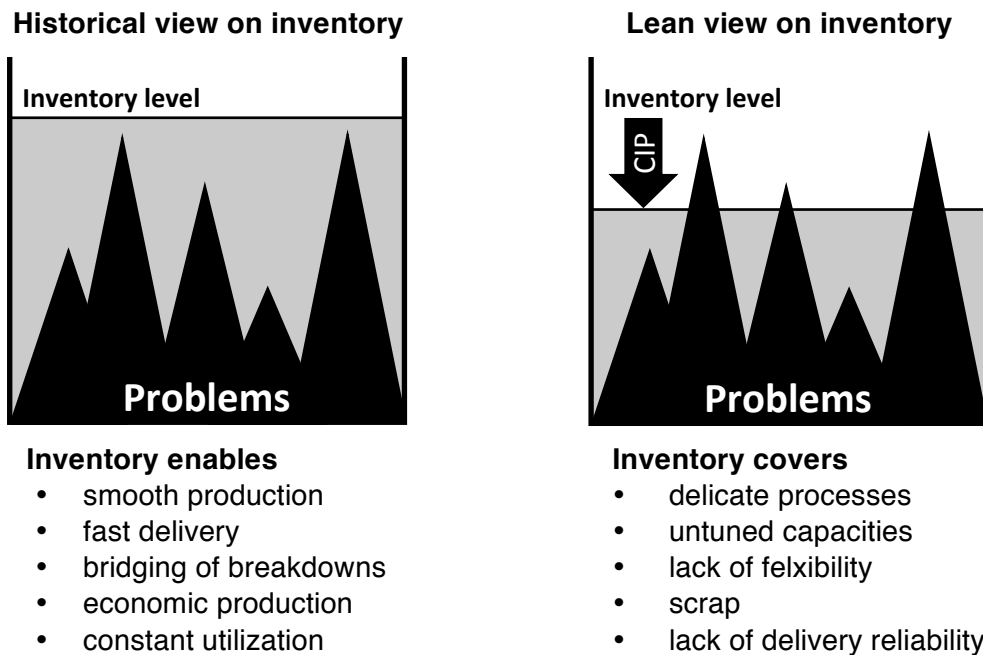


Figure 2: Different views on inventory [modified from Wiendahl 1997]

Furthermore, the inventory is influencing the timespan in which quality problems can be detected. Therefore, the timespan for detection is longer with higher inventory. Also the amount of scrap rises with higher inventory [Lödding 2005].

2.1.2.2 Performance and utilization

The physical power is defined as the ratio between work and time. This definition can be also adopted for the logistic performance [Wiendahl 1997]. The utilization is the ratio between the average and the maximum possible performance [Nyhuis 1999].

Traditionally the utilization is a very considered target value. Companies try to operate very expensive machines at full capacity to ensure the refinance. But after the investment, the investment costs are fixed and cannot be influenced by short-term planning decisions. Therefore, these sunk costs are not crucial to the production planning decisions. On the other hand, a high utilization generates high contribution margins and therefore high profit. Normally the utilization is limited to the customer demand. Even if the demand is higher than the possible output of a company then only bottleneck machines are fully utilized. And the overall output can only be increased if the capacity of the bottleneck machines gets increased [Lödding 2005].

2.1.2.3 Default costs

Default costs are those costs, which arise if the delivery of an order is too late (backorder). Some of these costs are directly measureable like penalty payments. But default costs also consider costs out of the fidelity damage [Lödding 2005].

2.2 Dilemma of the production logistic

The main challenge of the production logistic is that the improvement of one logistic target value leads to a decline in another target dimension. This target contradiction occurs in the operation planning and also in the disposition and is known in the literature as dilemma [Gutenberg 1951] or conflict of interests of the production logistic.

2.2.1 Operations planning dilemma

The operations planning dilemma occurs in the production planning. The main target of company is to finish all production orders in due time. A measure for that is the delivery date deviation or the delivery reliability. Especially on oscillating customer needs, this target is only achievable with unlimited production capacity. Because of the limited investment capabilities the production capacity is always limited. Furthermore, another target of the production logistic is to accomplish a high capacity utilization to be able to produce on low production costs. The realization of high capacity utilization on oscillating customer needs leads to oscillating lead-times. As the lead-time is directly influencing the delivery time this is again threatening the target of a high delivery reliability. Furthermore, also the inventory is influencing the lead-time and should be therefore minimized. As in Section 2.1.2.1 shown a low inventory has also other positive effects. These three main dimensions can be influenced through the availability and in the production planning through the lot size and the production scheduling (see Figure 3).

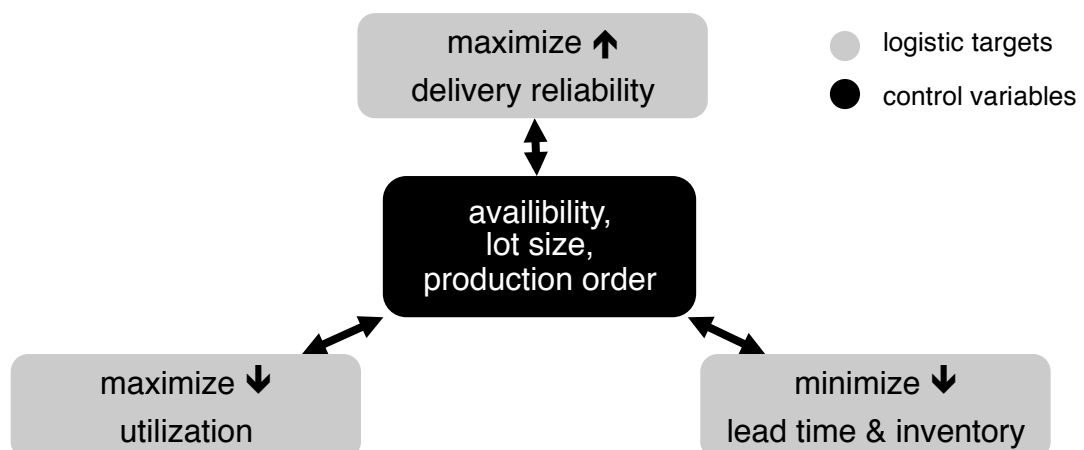


Figure 3: Operation planning dilemma [Erlach 2010]

2.2.2 Disposition dilemma

The disposition dilemma occurs in the procurement logistic with the disposition of raw, semi finished and finished products. In this case the target dimensions ability to supply, material costs and inventory are in conflict.

2.2.3 Target dilemma of the production

Erlach [2010] combines the operations planning and the disposition dilemma into the so-called target dilemma of the production with the three main targets: quality, velocity and economy (see Figure 4). This definition is quite similar to the definition of the aim of production logistic from Bauer [2011] (see Table 1).

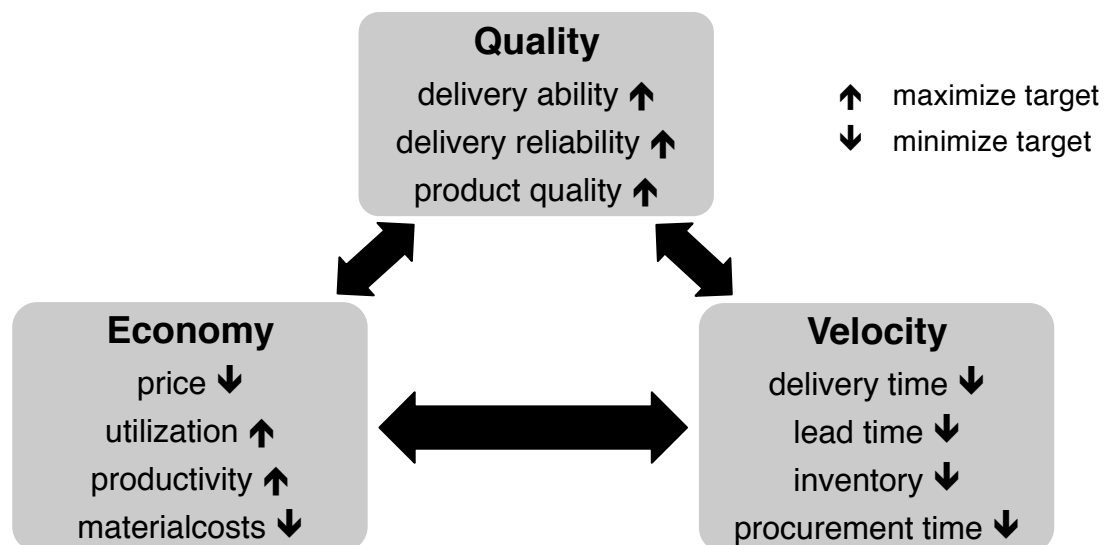


Figure 4: Target dilemma of the production [Erlach 2010]

Within quality there are on the one hand technological quality targets and on the other hand logistical quality targets. These logistical quality targets are the already discussed delivery reliability and also the delivery ability. The delivery ability is the ability to meet a customer need in terms of delivery time. The group velocity combines the delivery time, the lead-time, the amount of inventory and the procurement time. The economy group deals with the price, the utilization, the productivity and also the material costs.

3 Production planning

Production planning is the process of decision making for the production of goods. Furthermore, it is the definition making process of production under consideration of the availability of production factors. The tasks of production planning can be divided with respect of the planning horizon [Domschke et al. 1993].

Strategic planning considers a long-term perennial time horizon. Thereby, decisions about the product range and production location are made.

Tactic planning considers a period of several months to one year. Thereby, decisions about the equipment of a company in terms of machines and staff as well as cooperation's with suppliers are made.

Operative planning considers a period of one week or even shorter. Within the operative planning decisions about the scheduling of personal and the scheduling of production jobs are made. Furthermore, any kind of material and equipment requirement and preparation planning is made.

This thesis mainly concerns with the operative planning, especially the scheduling of production jobs. Therefore, in the next section the core tasks of production planning are explained in detail.

3.1 Core tasks of production planning

The core tasks of production planning are: production program planning (see Section 3.1.1), production requirement planning (see Section 3.1.2) and in-house production planning (see Section 3.1.3).

3.1.1 Production program planning

The core task of production program planning is to generate the production program for a company. This production program contains for each product and each planning period the required production amount.

Like it is shown in Table 3, the net primary requirements can be calculated out of the planned sales minus the amount of products, which are already on stock, plus the safety stock.

Planning period	1	2	3	4	5
Planned sales (Gross primary requirements)	20	25	22	24	26
- Inventory	30	20	20	20	20
+ Safety stock	20	20	20	20	20
Planned production (Net primary requirements)	10	25	22	24	26

Table 3: Production program planning [Lödding 2005]

On seasonal demand the company has to make the decision either to follow the demand or to level the production (see Figure 5).

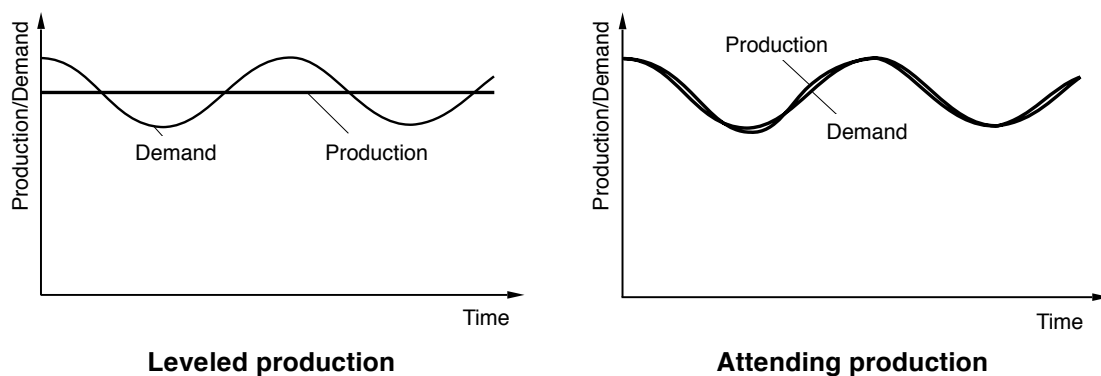


Figure 5: Production planning with seasonal demand [Lödding 2005]

Leveling of the production has the advantage of a constant utilization, but also the disadvantage of high inventory and therefore a higher level of bonded capital (see Section 2.1.2.1). The second strategy, attending the production, needs a high flexibility in workforce capacity. Furthermore, also the machine capacity has to be dimensioned higher. In practice normally a compromise between these two production strategies is made.

3.1.2 Production requirements planning

The task of production requirement planning is to transfer the net primary requirements into material and resources requirements. For that the dependent requirements get determined by breaking down the bill of material (BOM).

3.1.3 In-house production planning

The in-house production planning deals with the two main tasks of the production planning: lot sizing and production scheduling. A lot is the summarization of production jobs, which are produced in series on a machine. The lot sizing determines the optimum lot size with respect to costs. Afterwards production scheduling deals with the sequence (order) of the production lots [Schoner 2007].

3.1.3.1 Lot sizing

Lot sizing exists in the procurement as well as in in-house production. The main importance of lot size planning is the combination of different economic targets. Most lot sizing methods aim to minimize the lot size depending costs, which are particularly setup costs and inventory holding costs [Lödding 2005].

Setup costs: Normally a machine needs to get setup when a new version is produced. Through this setup process costs for staff or also material, which is used for the setup procedure, can occur. Furthermore, the setup is reducing the capacity of the machine and therefore additional opportunity costs in the high of the lost contribution margin arise. The setup costs decrease with the lot size.

Inventory holding costs are the costs for the inventory of finished goods and also semi finished products (see also Section 2.1.2.1)

The lot size planning was introduced first by Harris¹ under the name “Economic Order Quantity Model” (EOQ-model) and then by Andler². This method calculates the lot size by minimizing the sum of setup costs and inventory holding costs.

$$\text{setup costs} = \frac{D * C_S}{Q} \quad (7a)$$

$$\text{inventory holding costs} = \frac{C_H * Q}{2} \quad (7b)$$

$$\text{total costs} = \text{setup} + \text{inventory holding costs} = \frac{D * C_S}{Q} + \frac{C_H * Q}{2} \rightarrow \min! \quad (7c)$$

$$\rightarrow Q^* = \sqrt{\frac{2 * D * C_S}{C_H}} \quad (7d)$$

with Q lot size [#]
 Q^* optimum lot size [#]
 D annual demand quantity [#]
 C_S lot size independent costs (costs per setup) [\$]
 C_H lot size dependent costs (inventory holding costs) [\$/#]

The EOQ-model assumes a continuous demand, no capacity limits and a single stage production. That’s why nowadays often more complex models are used (see Section 4.2) [Rossi 2003].

All in all it is very difficult to combine all influences into such a calculation scheme. Therefore, the lot size is normally too big. Lödding [2005] gives three reasons for that:

1. As the EOQ-model is just taking costs into account, other positive effects of smaller lot sizes are not considered (see Section 2.1.2.1). The lot size is mainly affecting the lead-time in the production. The shorter the lead-time the shorter the delivery time gets and also the safety stock can be at a lower level.
2. Companies rate the setup costs to high. Many companies take for the cost calculation also, beside the staff costs, the depreciation of the machine into account. As the investment costs are sunk costs, just staff costs can be taken into account. Only if the company is producing on the capacity limit, the lost contribution margin during the setup process can be considered.
3. In my cases setup time can be reduced through organizational measures.

3.1.3.2 Production scheduling

The task of production scheduling is the time assignment of the production jobs to machines. Therefore, production scheduling is the planning of the order in which production jobs are machined. Additionally, even time points, when an operating

¹ Ford W. Harris 1913 in the USA

² Kurt Andler 1929 in Germany

procedure is fulfilled, get defined. The most important methods of production scheduling are the following [Lödding 2005]:

1. **Interactive production scheduling with control center:** an electronic control center visualizes the order of production jobs on all machines with Gantt charts¹. This control center supports the production planner by scheduling the production jobs manually [Dangelmaier & Aldinger 1986].
2. **Production scheduling with sequence rules:** Defined sequence rules are used to schedule the production jobs. These rules define the sequence in which production jobs are scheduled to the different machines. Such a method is only applicable in the scheduling of a simple production with a low number of different products.
3. **Optimizing algorithms:** Many scientific researches have the target to optimize the order of production jobs with an algorithm to achieve a certain objective. The operator of such an algorithm describes the objective and then the algorithm is calculating an order, which fits the best. There are different methods from the field of operations research, which can be applied on the production scheduling problem.

Section 4 gives an overview of such methods.

¹ A Gantt chart is a type of bar chart, developed by Henry Gantt (1861-1919)

4 Models and methods of production scheduling

4.1 Definitions

Makespan

The total production time it takes to produce a schedule is called makespan [Schoner 2007].

Job-Shop problem

The Job-Shop problem describes the sequence, also known as scheduling, problem for a certain amount of jobs. The production scheduling deals with the problem that all jobs have to be processed through a number of machines in a given technological order in a way that the makespan is the shortest [Schoner 2007].

Flow-Shop problem

If all jobs have to get processed in the same order through the same machines then the problem is known as Flow-Shop problem. Therefore, a Flow-Shop problem is a special case of a Job-shop problem [Schoner 2007].

Complexity

With the amount of production jobs also the amount of possible solution is increasing. The difficulty, to find the optimum solution within acceptable calculation time, is described in the complexity theory¹. An optimization problem is polynomial solvable, if there is a deterministic algorithm that finds for all possible input values with the problem size² n a solution within polynomial time. Meaning for a problem with n input data a solution can be found at least in n^k solution steps. Problems, which are only solvable in polynomial time with non-deterministic algorithms, are called \mathcal{NP} -hard³ [Domschke et al. 1993].

¹ The complexity theory describes referencing to decision problem if there is a solution or not. In contrast optimization problems search for the best solution. Nevertheless optimization problem can be transferred into decision problems by asking if there is a solution with an unknown optimum value.

² The problem size identifies the amount of data to be optimized.

³ \mathcal{NP} stands for non-deterministic polynomial

4.2 Advanced lot sizing models

The classical approach of the EOQ formula, which was shown in Section 3.1.3.1, has many restrictions like a continuous demand, no capacity limit and a single stage production.

Beside, this classical approach also more advanced methods exist. An also very basic method is the Wagner-Whitin Algorithm [Wagner & Whitin 1958], which takes a various demand for the product over a period of time into account. This problem is known as single-level uncapacitated lot sizing problem (SLULSP) [Rossi 2003].

The problem, which also considers a multistage, capacitated production process, is known as multi-level capacitated lot sizing problem (MLCLSP). For this problem several different model formulations and variants exist. Rossi [2003] gives in his dissertation a good overview of these formulations.

A formulation for the MLCLSP from Stadtler [1996] can be found in the Attachment 1.

4.3 Modeling of sequence problems

One key aspect in production scheduling is the modeling of the time period and how time points are represented within this model. The decision about the modeling of the sequence problem also determines the amount of variables and constraints in the optimization model.

For modeling of the time period two main models exist: models with discrete time points¹ and models with continuous consideration of the time period. In discrete time models processes can only start on certain intervals and therefore only these time points have to be considered in the planning. In continuous time models a variable defines the start point of a process [Schoner 2007].

The classical method is the description of the problem as a mixed integer linear programming problem (see Section 4.3.1). Another procedure is to describe the problem with methods of the graph theory (see Section 4.3.2). Furthermore, it is also possible to build up the problem in a discrete event simulation (see Section 4.3.3).

4.3.1 Mixed integer linear programming

By describing the problem as a mixed integer linear programming problem (MILP), the model is solved in a form like this [Schoner 2007].

$$\begin{aligned} \min \quad & c^T x & (8) \\ \text{with the constraints} \quad & Ax \geq b \\ & c \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m, x \in \mathbb{R}^n \end{aligned}$$

Normally the model also has a non-negative constraint: $x \geq 0$. The model is named linear model because the objective function and also the constraint functions are only having linear dependences. Mixed integer models have additionally for the decision variable the constraint that it has to be an integer $x_i \in \mathbb{N}$ or even binary $x_i \in \{0,1\}$.

By modeling the sequence problem with MILP two main properties have to be considered [Schoner 2007]:

- **Sequence:** all jobs must be machined in the given order
- **Exclusivity:** one machine cannot perform two jobs at the same time

There are different ways how to implement these two properties into a MILP problem. Three different formulations for the sequencing problem can be found in the following subsections.

¹ In discrete models the time horizon is divided into a finite amount of intervals.

4.3.1.1 Assignment problem

The assignment problem is solving the sequence problem by ordering the jobs in a position table. This position table has as much entries as jobs have to be scheduled. This means that an optimization problem with k jobs needs k^2 binary decision variables. Therefore, it is obvious that such a problem formulation leads to a high number of decision variables, because of the quadratic factor [Heipcke 2002].

A formulation for the Flow-Shop problem, using priority lists from Heipcke [2002], can be found in the Attachment 2.

4.3.1.2 Time point formulation

Another used formulation for the modeling of the sequence problem is with discrete time points. In an interval only one job can be processes on a machine. Therefore, the amount of intervals t defines the amount of decision variables. For each job t decision variables have to be defined. These decision variables decide if a job starts on the beginning of a new time interval [Schoner 2007].

On example for such a formulation is the State-Task-Network (STN). A STN consists of two types of nodes. The state node represents raw material, semi-finished and finished goods. The task node is used to describe the production process. Based on this network Kondili [1993] developed a MILP model for the scheduling problem.

4.3.1.3 Disjunctive formulation

The disjunctive formulation of the sequencing problem formulates the temporal alternative between two jobs. For the start time points $st(j_1)$ and $st(j_2)$ of two jobs j_1 and j_2 with the production duration t_{j_1} and t_{j_2} the formulation looks like this [Heipcke 2002]:

$$st(j_1) + t_{j_1} \leq st(j_2) \quad or \quad st(j_2) + t_{j_2} \leq st(j_1) \quad (9)$$

Heipcke [2002] also describes the translation of this formulation into a MLIP. For such an formulation with k jobs $\frac{k(k-1)}{2}$ decision variables and constrains are needed.

4.3.2 Graph theoretical modeling

Another way of modeling the sequence problem is using the graph theory. The big benefit of such a graph theoretical model is that the solution of the sequence problem is not just a starting order like in a MILP solution. The solution of a sequence problem can be represented in a network plan or graphically in a Gantt chart. Domschke [1993] gives a closer description of this procedure.

4.3.3 Discrete-event simulation

A simulation is the execution of an experiment in a model. Thereby, the model is an abstracted reproduction of a system, which should be reviewed. This means that the structure and the behavior of the system are described with a certain degree of accuracy within the model. Typically simulations are used to evaluate the performance of a system in a decision making process.

Simulation models can be classified in the following ways [Lothar et al. 2011]:

Static vs. dynamic

- Static: the system is reviewed on a certain point in time or the time has even no influence (e.g. Monte Carlo simulation¹)
- Dynamic: the system is reviewed in its behavior in terms of time (e.g. production models)

Deterministic vs. stochastic

- Deterministic: the system has no stochastic component (e.g. chemical reaction)
- Stochastic: the system behavior is influenced by random events (e.g. queuing models)

Continuous vs. discrete

- Continuous: system stats change continuous (e.g. differential equations)
- Discrete: system stats change at discrete time points (e.g. stock keeping system)

In discrete-event simulation (DES) only the points in time at which the state of the system changes are represented. That means that the system is modeled as a series of events, that is, instants in time when a state-change occurs [Robinson 2004].

Algorithm 1 shows the principle sequence of a DES. A DES consists of two main sections. First of all of the initializing section (see Line 1-5) and then the main loop (see Line 6-10) in which the simulation take place.

Within this loop the clock gets increased (see Line 7) and also the planned events get executed (see Line 8) every iteration until a defined ending condition is reached (see Line 6). Also various statistics (see Line 9) are updated for a later on visualization of the output.

¹ Monte Carlo simulation, the name given by John van Neumann and Stanisław M. Ulam, utilizes models of uncertainty where representation of time is unnecessary. Typical of Monte Carlo simulation is the approximation of a definite integral by circumscribing the region with a known geometric shape, then generating random points to estimate the area of the region through the proportion of points falling within the region boundaries [Nance, 1993].

Discrete-event simulationAlgorithm 1

```
1: Start
2:   Initialize ending condition to FALSE
3:   Initialize system state variables
4:   Initialize clock (simulation time = zero)
5:   Schedule an initial event
6:   Do loop While ending condition is FALSE:
7:     Set clock to next event time
8:     Do next event and remove from the Events List
9:     Update statistics
10:  End
```

The big benefit of such a DES is, like in a graph theoretical model, the output of the solution. Also hereby the output is not just a starting order like in a MILP solution. The solution of a sequence problem can be represented graphically in a Gantt chart and also further information about the simulation, like the utilization of a machine, can be represented.

4.4 Optimization methods

Nearly all scheduling problems are \mathcal{NP} -hard [Pinson 1995]. Only the special case of a two-machine Flow-Shop problem can be solved with the Johnson-Algorithm [Johnson 1954]. Furthermore, Akers [1956] has developed a procedure for minimizing the makespan in case of a Job-Shop problem. Therefore, instead of exact methods, algorithms with systematic or heuristic search strategy have to be used.

4.4.1 Systematic search

Systematic search methods improve the optimum solution of a scheduling problem by narrowing the solution space step by step. For a mathematic programming a formulation as MILP is necessary.

One of the most known systematic search methods is the Branch & Bound algorithm (B&B or BB). In this method the individual decision variables get fixed sequentially and then based on this new intermediate solution further partial solution (branch) get evaluated. If a solution is found, all other branches that already have a poorer solution stay unconsidered (bound). Vahrenkamp [2003] gives a more detailed description of the single steps of B&B.

In general linear programmed methods always find the optimum solution of an optimization problem. But typical these methods need a lot of computing power and computing time.

4.4.2 Heuristic search

Heuristic search methods don't need a problem description in the form of a MILP problem. Such optimization methods are used when a systematic search approach would need too much computing time and too high memory requirements. With heuristic search it is not guaranteed that the optimum solution is found. Therefore, such methods are also referred to approximation methods [Schoner 2007].

Heuristics for the scheduling problem normally use a two-stage practice. In the first step a construction heuristic generates a starting solution. In the second step an improvement heuristic modifies this solution. If this improvement heuristic just accepts better solutions then it is called iterative improvement. If also worse solutions are accepted, then the method is called meta heuristic [Brucker 1995].

The reason why meta heuristics also accept even worse solutions is visualized in Figure 6. This figure shows a graph with the objective function, which has a global minimum in point 8. The heuristic method calculates out of the starting solution the

solution 1. In the next step the improvement heuristic stochastically calculates the solution 2, and so on. As solution 4 is worse than solution 3 an iterative improvement heuristic method would not accept this solution and therefore probably this method won't find a better solution. As meta heuristic also accept to a defined degree worse solutions, the algorithm can find his way out of the local optimum in solution 3 and find the global optimum in solution 8.

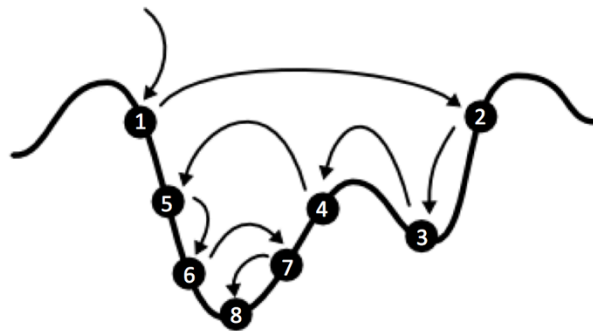


Figure 6: Solution steps of a meta heuristic method

4.4.3 Meta heuristic improvement methods

Meta heuristic improvement methods consist of two main parts. A permutation algorithm transforms an established solution into a new one. An acceptance algorithm decides if this new solution gets the new base for a further search. With meta heuristic improvement methods the new solution can be even accepted if it is worse than the previous one. Often the acceptance algorithm is recreated out of natural phenomena. An example for that is Simulated Annealing (SA), which emulated the thermodynamical behavior of crystals [Schoner 2007].

Meta heuristic improvement algorithms can be classified into methods, which are set based, and methods, which just consider one solution. Set based methods are Genetic Algorithms (GA), Evolution strategies (ES) and Tabu Search (TS).

Genetic Algorithms¹ and Evolution Strategies² always consider an amount of solutions (population). These solutions get modified and the best is selected with the principle of Darwin³ "survival of the fittest". Evolution strategies form their new solution through mutation of existing solutions. While Genetic Algorithms form new solutions through genetic recombining of several individual solutions, also called crossover [Schoner 2007].

¹ proposed by John H. Holland in 1975

² proposed by Ingo Rechenberg et al. in 1965

³ Charles R. Darwin (1809-1882) was an English naturalist

In contrast to the above set based algorithms Tabu Search¹ is modifying just one solution. But this method stored information about earlier solutions in a so-called Tabu list. These solutions are not considered for a further local search [Schoner 2007].

Meta heuristics, which work not set based, are: Simulated Annealing² (SA) and several modifications like Threshold Accepting³ (TA).

Because these methods are simple to implement, they are also often used for sequencing problems [Schoner 2007]. Schoner refers that Vaessens [1996] experienced for the Job-Shop problem a better performance of SA than TA. Also Käschel [1999] experienced in a comparison of several heuristic methods (SA, TA, and GA) a dominance of SA.

4.4.3.1 Simulated Annealing

The method of Simulated Annealing (SA) was developed independent by Kirkpatrick [1982] and Černý [1985] and is based on an algorithm, which simulates the physical behavior of solid objects proposed by Metropolis [1953].

The idea of SA comes from the metallurgy. A metal (crystal) should get into a stable and reliable state (= a as pure as possible crystal structure). For that the metal get heated and afterwards slowly and controlled annealed. During this annealing process the atoms get transferred from a high energy state to a lower one. SA is a heuristic method with a stochastic component. Starting from an existing solution the permutation algorithm generates a new solution. This new solution gets accepted with a certain probability. The acceptance probability p depends on the temperature T_i of the respective iteration step i [Schoner 2007].

Algorithm 2 shows the principle of the SA algorithm.

Simulated Annealing	Algorithm 2
1: Given: starting solution \mathcal{S} objective function $f(\mathcal{S})$ mutation rule op	
2: Define: iteration step $i = 0$ starting temperature T_0 break criteria X	
3: Return: best solution \mathcal{S}_{best}	

¹ proposed by Fred W. Glover in 1986

² proposed by Scott Kirkpatrick et al. in 1983

³ proposed by Gunter Dueck et al. in 1990

```

4: Do Loop While  $X \neq TRUE$ 
5:   Permutation of the existing solution  $\mathcal{S}' := op(\mathcal{S})$ 
6:   Accepting decision: if  $f(\mathcal{S}') \leq f(\mathcal{S})$ 
       • then  $\mathcal{S}_{best} := \mathcal{S}'$  and  $\mathcal{S} := \mathcal{S}'$ 
       • else if  $p < \exp\left(\frac{f(\mathcal{S}')-f(\mathcal{S})}{T_i}\right)$  for a random number  $p \in [0,1]$ 
           o then  $\mathcal{S}'$  is new starting solution  $\mathcal{S} := \mathcal{S}'$ 
7:   Set:  $i = i + 1$  and  $T_i = \vartheta(T_{i-1})$  and determine break criteria
        $X := X(T_i, i)$ 
8: End

```

The iteration starts in line 5 with the permutation of the existing or starting solution \mathcal{S} into \mathcal{S}' with an mutation rule $op(\mathcal{S})$. This permutation rule modifies \mathcal{S} into a neighborhood solution \mathcal{S}' . There are several strategies to find a suitable neighborhood. The easiest one is just a two variable exchange. This would mean for the scheduling problem that two jobs in the production plan get exchanged. In the next step (line 6) the accepting rule decides whether the new solution gets accepted or not. The so-called Boltzmann probability distribution

$$P(\text{accepting } \mathcal{S}') := \begin{cases} 1 & \text{if } f(\mathcal{S}') \leq f(\mathcal{S}) \\ \exp\left(\frac{f(\mathcal{S}') - f(\mathcal{S})}{T_i}\right) & \text{else} \end{cases} \quad (10)$$

is based on the idea of annealing liquid metal into a solid state. At high temperatures T_i the probability to accept a deterioration of the solution is higher. Furthermore, the level of acceptance is also depending on the degree of the deterioration $f(\mathcal{S}') - f(\mathcal{S})$.

Another essential criteria for the solution quality is the selection of the cooling scheme. Therefore, the cooling behavior $T_i = \vartheta(T_{i-1})$ and also the starting temperature T_0 have to be chosen wisely. The starting temperature can be calculated for a starting acceptance probability p with a maximum difference of the objective function $\Delta f_{max} = \max\{|f(op(\mathcal{S}) - f(\mathcal{S})|\}$ with

$$T_0 := -\frac{\Delta f_{max}}{\ln(p)}. \quad (11)$$

For the cooling behaviors various functions are possible like a multiplicative decrease.

$$T_i = \alpha * (T_{i-1}) \quad \text{with } \alpha \in [0,9; 0,99]. \quad (12)$$

The break criteria $X := X(T_i, i)$ can be either a fixed amount of total iterations or is a fixed amount of iteration since the last improvement [Schoner 2007].

4.4.3.2 Threshold Accepting

Based on SA Dueck & Scheuer [1990] developed Threshold Accepting (TA). The procedure of TA is similar to SA. Only the accepting rule is different (see Algorithm 3).

Threshold Accepting (changes to SA – Algorithm 2)	Algorithm 3
<hr/>	
1: Define: threshold value T_i	
2: Change: Accepting decision of Algorithm 2 with line 3	
3: Accepting decision: if $f(\mathcal{S}') \leq f(\mathcal{S})$	
• then $\mathcal{S}_{best} := \mathcal{S}'$ and $\mathcal{S} := \mathcal{S}'$	
• else if $f(\mathcal{S}') \leq f(\mathcal{S}) + T_i$ then \mathcal{S}' is new starting solution	
$\mathcal{S} := \mathcal{S}'$	

Instead of the stochastic accepting rule of SA, TA is accepting all worse solutions, which doesn't cross a certain threshold value T_i . Therefore, the method of SA for decreasing the temperature $T_i = \vartheta(T_{i-1})$ is now decreasing the size of the threshold value [Schoner 2007].

4.5 Optimization software and tools

4.5.1 Software for systematic search

For the implementation of the scheduling problem as a MILP problem several mathematical programming tools are available. A solver independent modeling language is AMPL, which was developed by Fourer [1990]. AMPL stands for “a mathematical programming language” and is a comprehensive and powerful algebraic modeling language for linear and nonlinear optimization problems, in discrete or continuous variables [AMPL 2012].

Systematic search methods need a problem description in MILP. Such a description needs a high degree of precision in the mathematical formulation. Already small changes within the production process can because extensive changes within the MILP model.

4.5.2 Software for heuristic search

Heuristic methods also work with graph theoretical model descriptions and discrete-event simulations. For solving the problem with a simulation based optimization using meta heuristic also several tools exist. One simulation tool, which is specialized in production simulation, is Plant Simulation from Siemens. Plant Simulation is a DES tool that helps to create digital models of logistic systems, such as production, so that system’s characteristics can be explored and the performance can be optimize [Siemens Plant Simulation 2012].

Plant Simulation offers an optimization module based on GA to optimize sequence, selection and set allocation problems. The *GAOptimization* module is the central control model for optimizations in Plant Simulation. *GASequence* is used to implement scheduling problems. With *GASelection* selection problems can be described. Selection problems have the task to select out of a set a certain number of elements. The module *GASetAllocation* is used to define allocation problems. Allocation problems have the task to assign each element out of a set to another element out of a set. The module *GAWizard* helps as an assistant to define such optimization problems within Plant Simulation [Völker & Schmidt 2010].

Figure 7 shows a simple scheduling problem in Plant Simulation. The basic simulation model consists of a *Source*, 2 machines (*Machine1* and *Machine2*) with a stock (*PufferStock*) in between and a *Drain*. A *DeliveryTable* is linked to the source and defines the inflow into the production. The *SetupMatrix* table is linked to the machines and provides information about the switch over time in case of a product type change.

The optimization problem is defined with the *GAWizard* module with the task to minimize the simulation time, meaning the total production time (makespan).

Völker & Schmidt [2010] implemented several standard scheduling problems in Plant Simulation and attest a good performance.

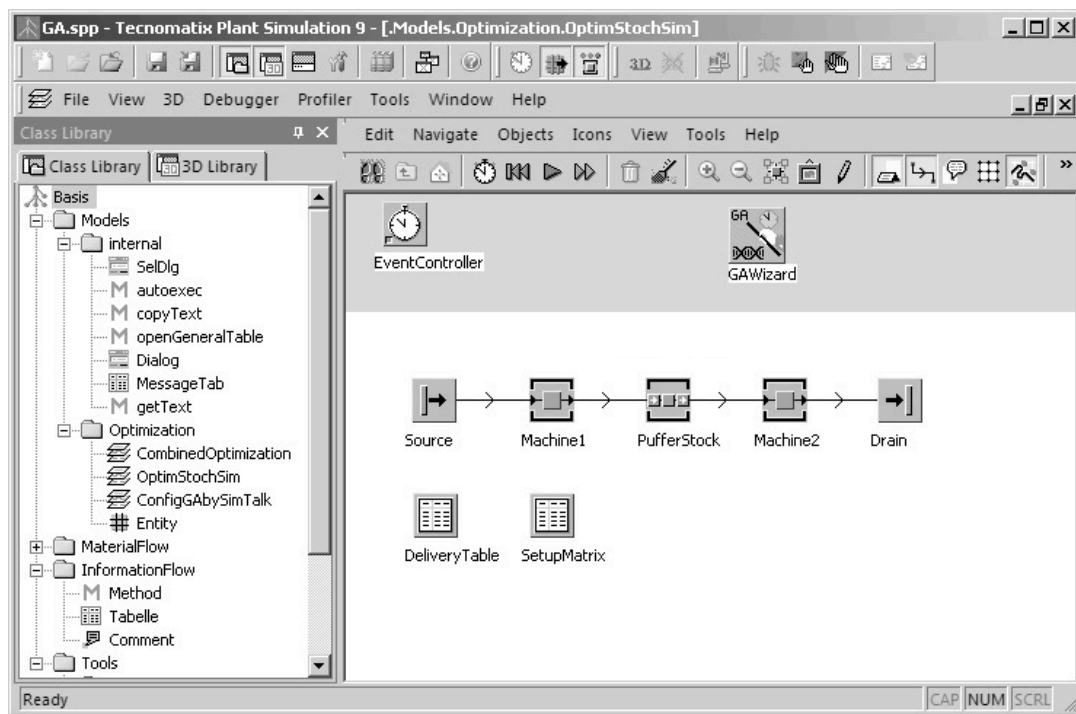


Figure 7: Plant Simulation with an optimization problem

4.5.3 Self made software

After a close consideration of several implementation possibilities I decided to use a discrete-event simulation model for the solution of the scheduling problem in this thesis. The big advantage of a simulation based optimization is that the outcome of the program is not just a starting order like in a MILP solution. Furthermore, a simulation model gets build up with modules. Therefore, the production process can be build up with such modules and later on the process can be modified much easier than in a MILP implementation.

Plant Simulation already offers, beside a lot of other features, with its *GASequence* module a state of the art software solution for scheduling problems. However, this program is very expensive and in principle the logic behind a simple discrete event simulation software can be programmed also very easy. Also the optimization algorithm SA can be implemented easily. Furthermore, the company TDK wanted an Microsoft Excel solution for this problem because they already have experience in Visual Basic for Application (VBA) of Excel.

5 Production process of multilayer varistors

5.1 Product description

Ceramic transient voltage suppressors (CTVS) are voltage-dependent resistors with a symmetrical V/I characteristic curve whose resistance decreases with increasing voltage (see Figure 8). Connected in parallel with the electronic device or circuit that is to be guarded, CTVS form a low resistance shunt when voltage increases above a CTVS type-specific threshold value and thus prevent any further rise in the transient overvoltage. [EPCOS 2010]

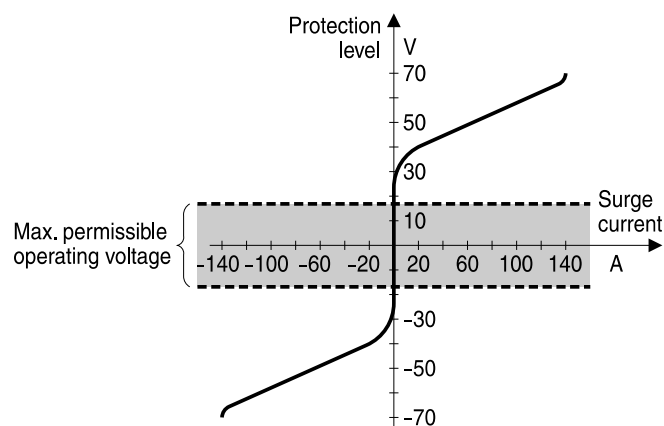


Figure 8: Protection level of the CTVS

Sintering zinc oxide together with other metal oxide additives produces a polycrystalline ceramic whose grain boundary resistance exhibits a non-linear dependence on voltage. This phenomenon is called the varistor effect.

The body of a multilayer CTVS, also known as multilayer varistor (MLV), consists of a stack of intercalating ceramic/electrode layers (see Figure 9). The thickness of the ceramic layers affects the protection level. The internal electrodes are connected to external electrodes [EPCOS 2010].

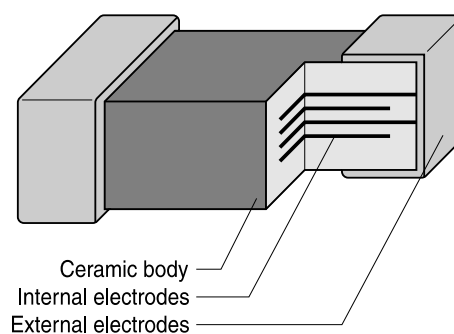


Figure 9: Internal construction of a multilayer chip component

5.2 Production process¹

The MLV production process is divided into two stages: the frontend processes and the backend processes. All the frontend processes take place in the production side of TDK in Deutschlandsberg. Further backend processes are performed in Croatia at a production side in Kutina (see Table 4). As this diploma thesis deals with the production scheduling in a part of the frontend production, the main frontend production processes are described in detail in the next sections.

Production Processes of multilayer varistors				
Frontend processes (Deutschlandsberg)	1.	Powder production		
	2.	Tape production		
	3.	Assembler		
	4.	Press		
	5.	Cutter		
	6.	Side printing		
	7.	Preparation for thermal processes		
	8.	Binder burn out		
	9.	Sintering		
		Backend processes (Kutina, Croatia)	10.	Tumbling
			11.	Passivation
			12.	Metallization
			13.	Firing
			14.	Plating
			15.	100% Sorting
			16.	Taping
			17.	Shipping

Table 4: Main production processes of the MLV production

5.2.1 Overview

The main processes of the MLV production in the frontend are the powder production, the tape production process, the assembler process, the pressing process, the cutting process, and finally the temperature processes with the preparation for the thermal processes, the binder burn out and ultimately the sintering process. Some types of the MLV also have to pass through the side printing process. Furthermore, also some side processes take place. The frontend production is running twenty-four-seven in 21 shifts.

In the tape production a thin ceramic tape gets manufactured. During the next step, the so-called assembler process, this tape gets printed with a conductive paste in a screen-printing process. Hereby the production sequence is very crucial because at a changeover of the screen some grams of the very expensive conductive paste gets lost. Furthermore, the several printed and unprinted layers of the tape get stacked during this process. After a process related lay time of 12 hours the stacked foils, also called stacks, get pressed together. Then after some side processes the final chip get

¹ All information, figures and data of this section are from internal TDK documents.

cut out of the stack. Depending on the case size of the chip a certain amount of chips can be cut out of a stack. Afterwards the chips are transferred to the temperature processes except a special automotive type of the MLVs, which have to be passed through another side printing process first. The thermal processes start with a preparation process in which the chips get loaded on special racks, also called setters. In the next production step the binder gets burned out. And finally the chips get sintered. As the sintering temperature is very critical for the electronic behavior of the MLVs, the frontend production consists of several furnaces, which can run on certain temperatures.

Figure 10 shows the product improvement during the production process, starting at the assembler with stacked and printed foils to the final sintered chip after the thermal processes.

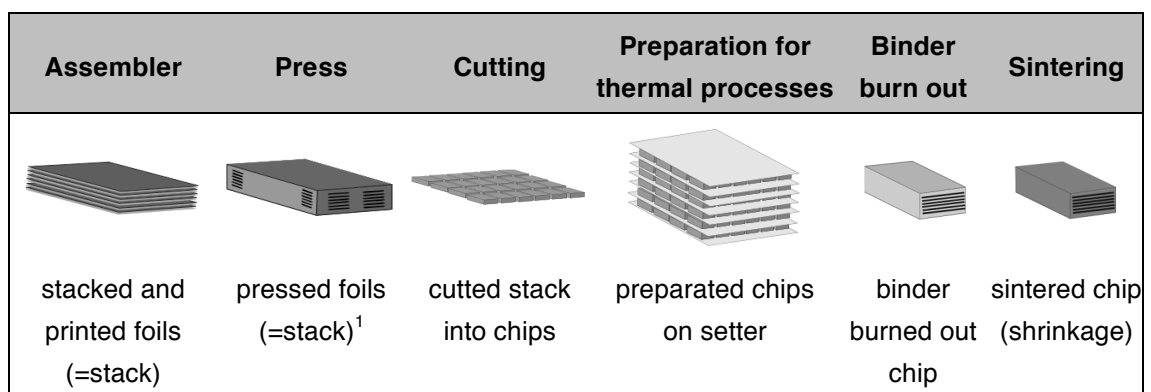


Figure 10: Product improvement during the production

After the shipment to the backend, the following production processes are performed: tumbling, passivation, metallization, termination firing, plating, 100% measuring & sorting and finally taping.

The production processes, beginning at the assembler to the sintering, are the area in which the production scheduling should be optimized. The production scheduling problem mainly arise because of the different process durations for different products. Approximately 5 billion chips in around 900 different types get produced annually at the production side in Deutschlandsberg. The high number of variances combines different case sizes and also different electric behavior.

¹ with cut-marks exposed

5.2.2 Main frontend processes

5.2.2.1 Assembler

In the assembler process a thin (around 50 μ m thick) ceramic foil gets cut into around 150x150mm pieces and picked up on a carrier. In the next step the foil gets printed with a conductive paste like Argent-Palladium in a screen-printing process. Afterwards the paste gets hardened and finally the printed foils get stacked.

In principle two different production technologies exist in the MLV frontend: the L90 technology and the L21 technology (see also Section 5.3). In the L90 technology the screen printing and stacking is combined in one machine like shown in Figure 11.

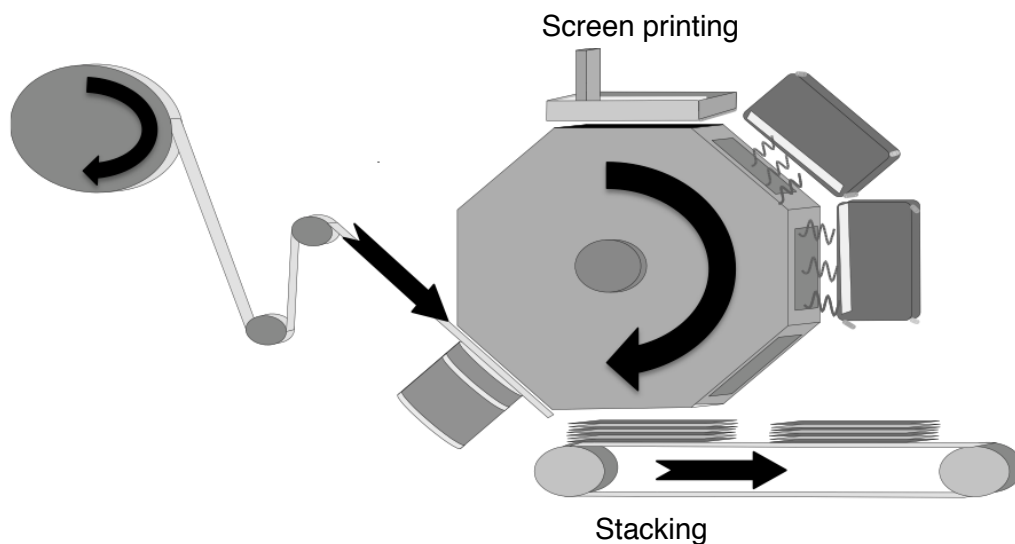


Figure 11: Assembler (L90)

In the L21 technology the stacking and screen printing is done in separate machines and furthermore another machine cuts cover foils for each stack.

The amount of foils a stack has depends on the electric behavior of the final MLV and can vary a lot (see Figure 12). The process duration can be calculated out of the amount of foils the MLV has, times the cycle time for one foil on the assembler.

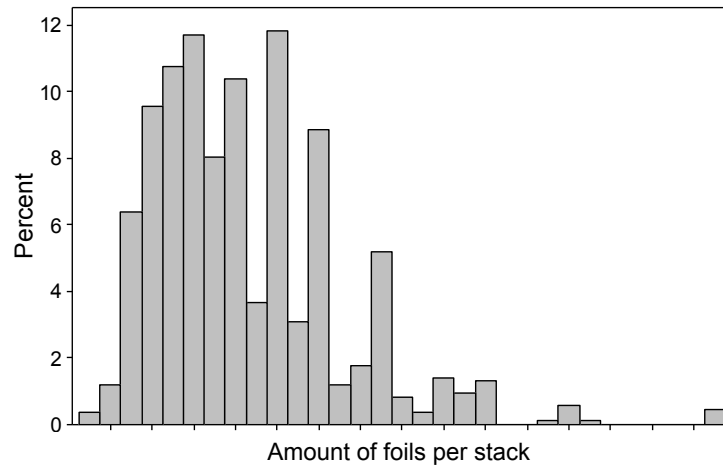


Figure 12: Histogram of the amount of foils per stack of all different MLV products

Another issue within the assembler process is the amount of setups. Every time the screen of the screen-printing device gets changed some of the expensive Argent-Palladium paste gets thrown away. Therefore, it is important to have a sequence in which all the products, which need the same screen, are in a row.

5.2.2.2 Press

After a process related lay time of 12 hours the stacked foils get pressed together. For that the stacks get build up on a carrier and get pressed together (see Figure 13). The principle of the pressing process is the same for L21 and also L90. The process duration is for each stack the same and is equal to the cycle time for one stack.

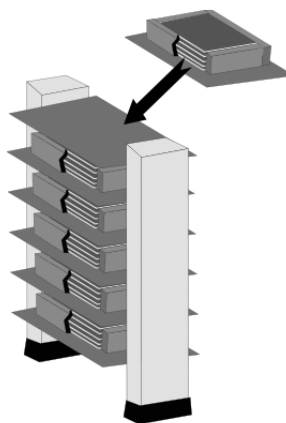


Figure 13: Press

5.2.2.3 Cutter

The cutting process is different for L21 and L90. In the L90 technology the edges of each stack get cut to expose the cut marks which are needed afterwards in the cutter process. Furthermore, an adhesive foil gets glued on each stack. For that two different technologies exist: an adhesive foil, which can be used just once, and an adhesive foil, which can be used for several times. For these two technologies also two different types of machines exist. These cutting machines are always combined into a pair, which is operated by one machine operator. The MLV frontend production has one cutting machine pair for each technology.

In the L21 technology the stack cutting for the exposure of the cut marks and also the adhesive foil is not needed. The stacks can be cut directly on a cutter.

The cutter cuts the stack with a knife into the single chips. (see Figure 14). The process duration depends on the amount of cuts and therefore on the case size of a chip.

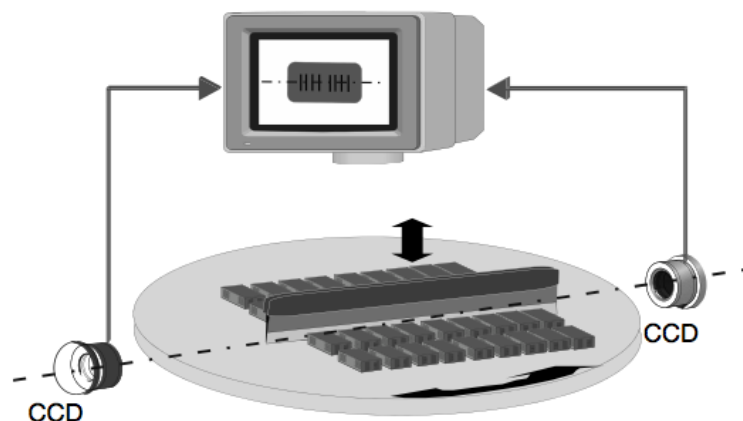


Figure 14: Cutting (L90)

For a MLV with the biggest case size 2220 only 200 chips can be cut out of a stack whereas for a MLV with the smallest case size 0201 about 76.000 chips can be cut out of a stack (see Table 5).

Case size	2220	1812	1210	1206	0805	0603	0402	0201
Chips per stack	200#	416#	825#	1.350#	2.829#	5.616#	14.529#	76.014#

Table 5: Overview of case sizes of the different MLV products

5.2.2.4 Side printing

A special type of MLV, a so-called CT type for the automotive industry, has to be processed through the side printing process. In this process the chips get printed with ceramic paste on 2 of the 4 sides, which are cut out of the stack. For that first of all, the chips get tempered, tumbled and washed. Then the chips get build up, either manually or automatic, in so-called setters. Afterwards the cut sides of the chips in this setter get printed in a screen printing process with ceramic paste.

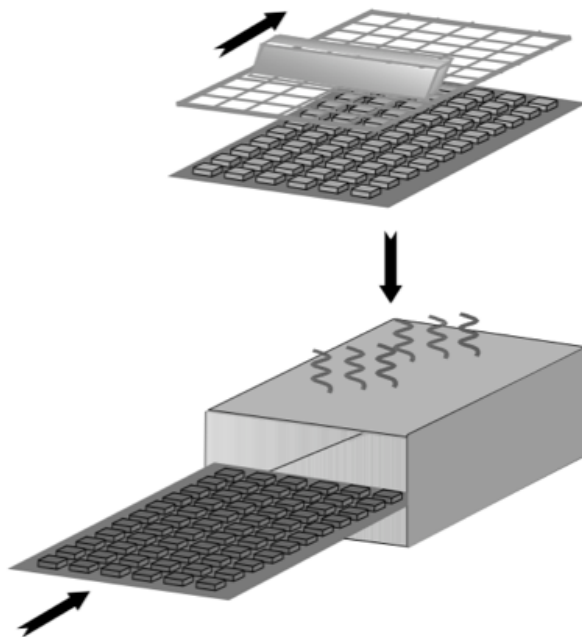


Figure 15: Side printing

5.2.2.5 Preparation for the thermal processes

Before the chips can be handled through the thermal processes, the chips have to be build up on special setters (see Figure 16). This can be either done manually or automatic. All MLVs which have been side printed must be build up manually.

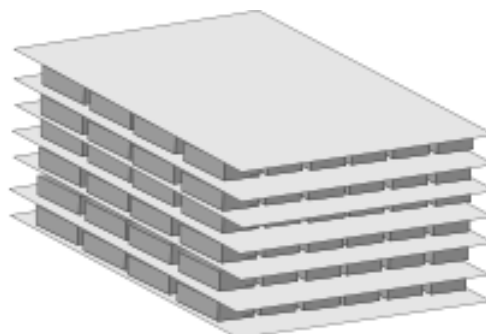


Figure 16: Setter with chips for the thermal processes

5.2.2.6 Binder burn out

In the first thermal process the binder gets burned out at a temperature of around 400°C for 36 hours. For that the frontend production owns 20 binder burn out furnaces which can take a loading of 23 kg each.

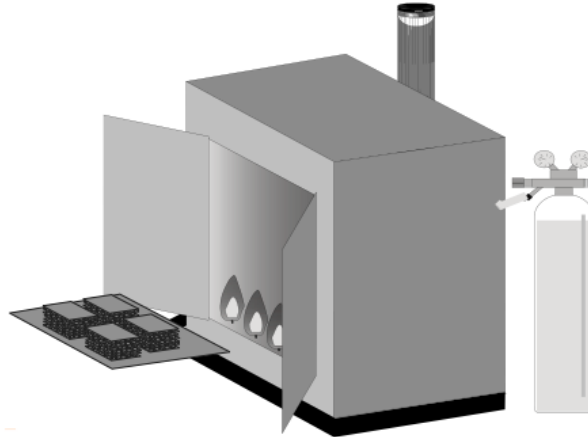


Figure 17: Binder burn out

5.2.2.7 Sintering

Finally, before the chips get shipped to the backend production, the chips get sintered at a high temperature around 1000°C. This sintering temperature influences the electric behavior. Therefore, the parts have various sintering temperatures. In principle the frontend production consists of 2 different types of furnaces: pusher type furnaces and rotary hearth furnace (see Figure 18).

Most of the parts are processed through the pusher type furnaces, which run 24 hours a day. At a pusher type furnace a changeover to another the temperature takes about 8 hours. The rotary hearth furnaces can be run up individually.

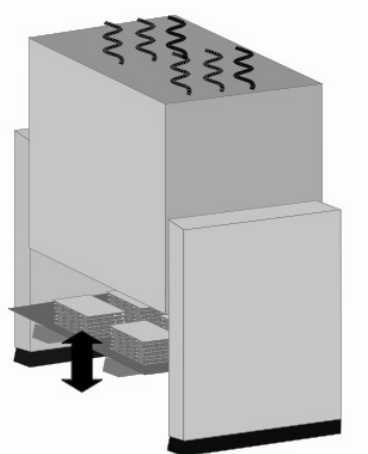


Figure 18: Sintering (Rotary hearth furnace)

These furnaces are limited to a maximum of 3 different temperatures (see Table 6). Some types of the MLVs have to run a test sintering procedure to find the right sintering temperature. In general this sintering temperature is known and most of the MLVs can be sintered directly.

	Furnace name	
Pusher type furnace	D6	2 Temperatures
	D7	3 Temperatures
Rotary hearth furnace	DH1	3 Temperatures
	DH2	3 Temperatures
	DH43	2 Temperatures
	DH14	1 Temperature
	DH36	1 Temperature
	DH8	2 Temperatures
	DH7	2 Temperatures
	DH42	2 Temperatures
	DH4	2 Temperatures
	DH13	1 Temperature

Table 6: Furnace temperature assignment

5.3 Summary of the frontend production processes

Figure 19 shows an overview of the MLV frontend production process separated into the main areas: L90 and L21 clean room and cutter area, side printing and the thermal area. Only about 5% of all the stacks run the L21 line. The main production flow is in the L90 technology.

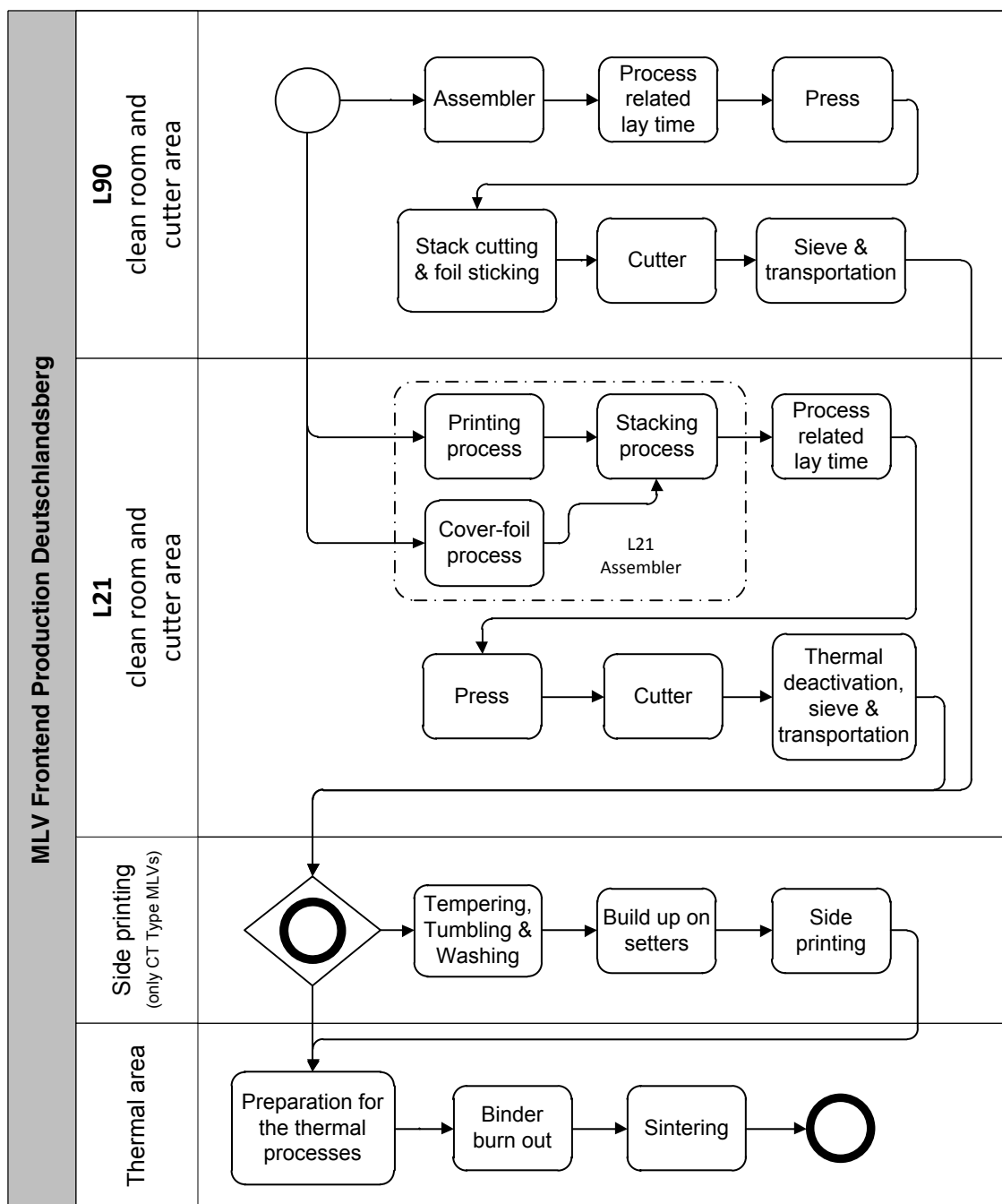


Figure 19: Overview of the production process

5.4 Simplification of the frontend production processes

Figure 20 shows the simplified process flow of the MLV frontend production beginning at the assembler process to the sintering process. In this figure all the main processes are marked in grey. As only 5% of the stacks run the L21 technology in the simplified production model only the L90 technology will be considered. Therefore, in this process overview the assembler, the pressing and the cutting process are L90 technology processes. Furthermore, the side printing process is simplified to a single process with an average process duration.

In the thermal area the bottlenecks are the pusher type furnace. About 80% of all stacks are sintered in one of the two furnaces of this type. Therefore, only this type of furnace is considered in the simplified model.

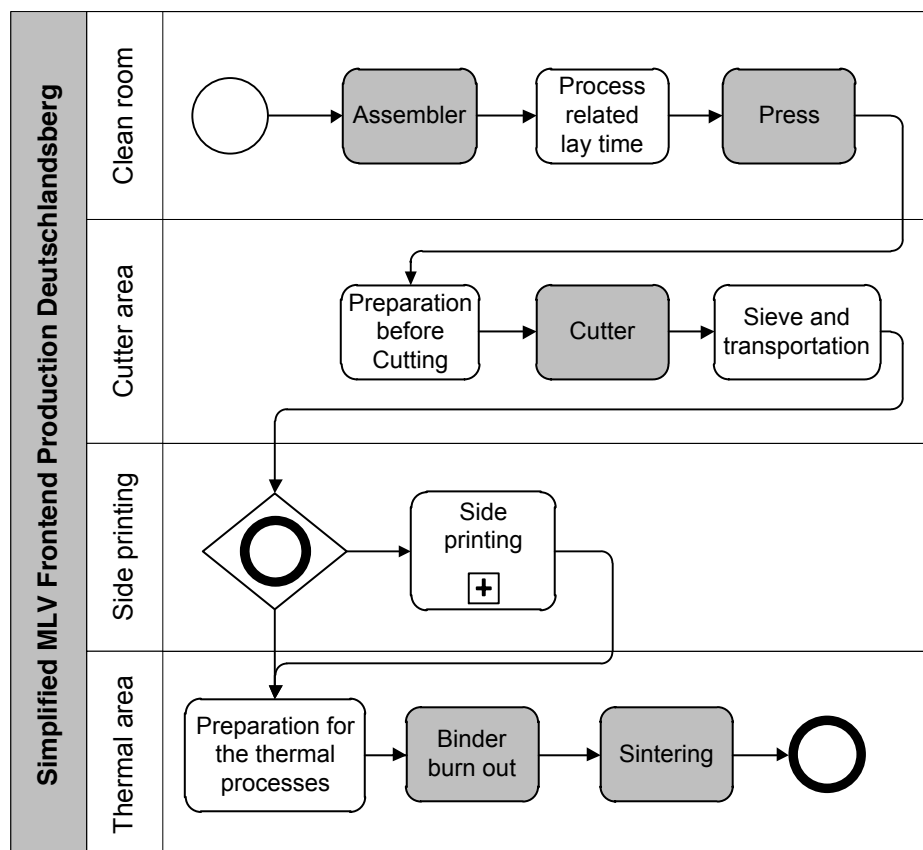


Figure 20: Overview of the simplified production process

The Table 7 shows the number of machines and the process duration per stack for the main processes. Beside, this process durations also setup times have to be considered at the assembler and also at the pusher type furnace. At the assembler the setup times for a screen changeover and also for a paste changeover are considered. Furthermore,

at the pusher type furnace the temperature changeover, that takes around 8 hours, is considered in the simplified model.

Process name	Number of machines	Process duration per stack
Assembler	2 assembler (setup considered)	<i>cycle time per foil * foils</i>
Press	1 press	<i>cycle time per stack</i>
Cutter	2 cutter pairs but only 1 operator → only 1 cutter pair in use at the same time	<i>cycle time per cut * cuts + handling time</i>
Binder burn out	20 binder burn out furnaces	<i>fixed cycle time</i>
Sintering	2 pusher type furnace (setup considered)	<i>fixed cycle time</i>

Table 7: Process details of main frontend processes

6 Production scheduling tool

The aim of the production scheduling tool is to find a detailed sequence for all production jobs with respect to a low inventory (WIP). The production scheduling tool also has to consider the special setup situation at the assembler.

The planning horizon is one week with a production demand of around 90 different jobs. Furthermore the optimization is done offline. That means that the production scheduling tool has no interaction with the real production system. The optimization tool is using a predefined model and the optimization process is done once for the production demand of the upcoming week. The approach of the production scheduling tool is a simulation based meta heuristic optimization. The simulation model is using the simplified production process from Section 5.4. The production scheduling tool is developed in Microsoft Excel 2003 Visual Basic for Application (VBA).

In the next section a general overview of the principle of a simulation based optimization is given (see Section 6.1). Section 6.2 explains the implementation of the discrete event simulation used in the production scheduling tool. Furthermore, Section 6.3 explains the implementation of the optimization algorithm. In Section 6.4 the objective value for this scheduling problem gets deducted. The graphical user interface (GUI) of the production scheduling tool gets explained in Section 6.5. Finally, also the testing results are shown in Section 6.6.

6.1 Simulation based optimization

The production scheduling tool is using a simulation based optimization approach. For that the tool consists in principle of two main modules: the simulation module and the optimization module (see Figure 21). The simulation module is a discrete event simulation model of the simplified production process (see Section 4.3.3). This simulation module is connected to an optimization module. The optimization module uses a meta heuristic improvement method and consists of an acceptance algorithm and a permutation algorithm. Since Simulated Annealing (SA) is simple to implement and also has a good performance in scheduling problems, this method is used in the optimization module (see Section 4.4.3.1).

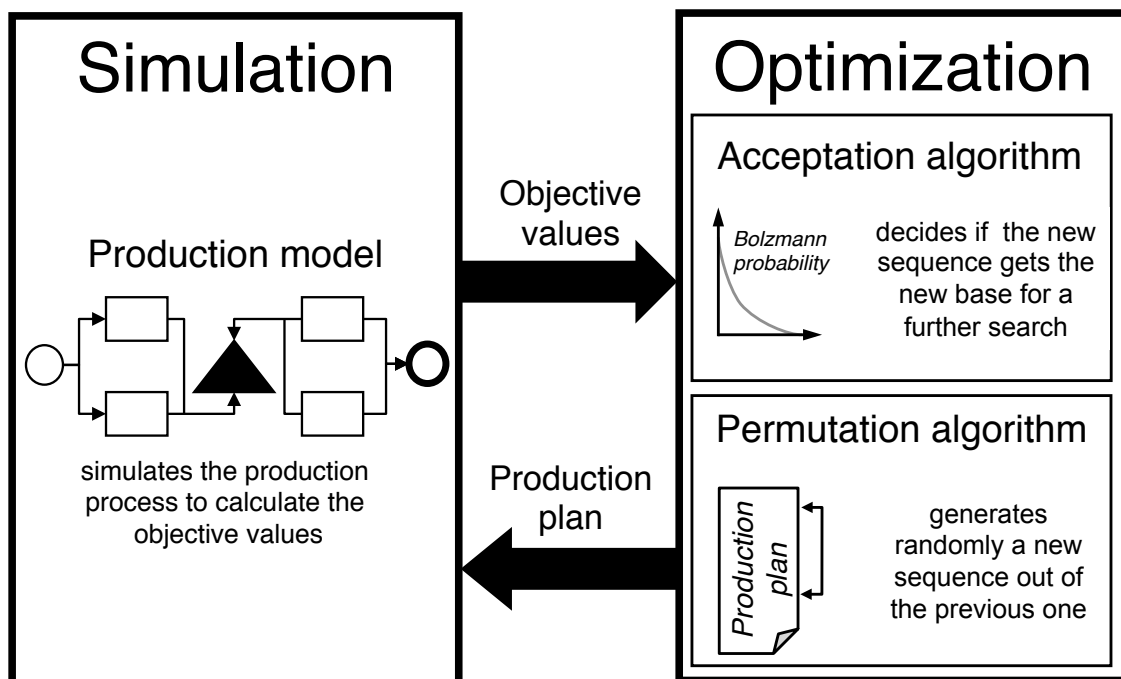


Figure 21: Architecture of the production scheduling tool

In principle the production scheduling tool is working like this: First of all the permutation algorithm is generating a new sequence out of a previous production plan. Then this production plan gets handed over to the simulation module. Within the simulation model this production plan gets simulated and several objective values are calculated. Then these objective values get passed again to the optimization model. Within the meta heuristic acceptance algorithm the decision, whether this sequence will be the base for a further search or not is made. This acceptance algorithm is based on the SA algorithm with the Boltzmann probability (see Section 4.4.3.1 and Equation (10)).

6.2 Simulation

The production model is a discrete-event simulation model of the simplified production process. The production process can be modeled with the core elements: *Machine*¹ and *Queue*.

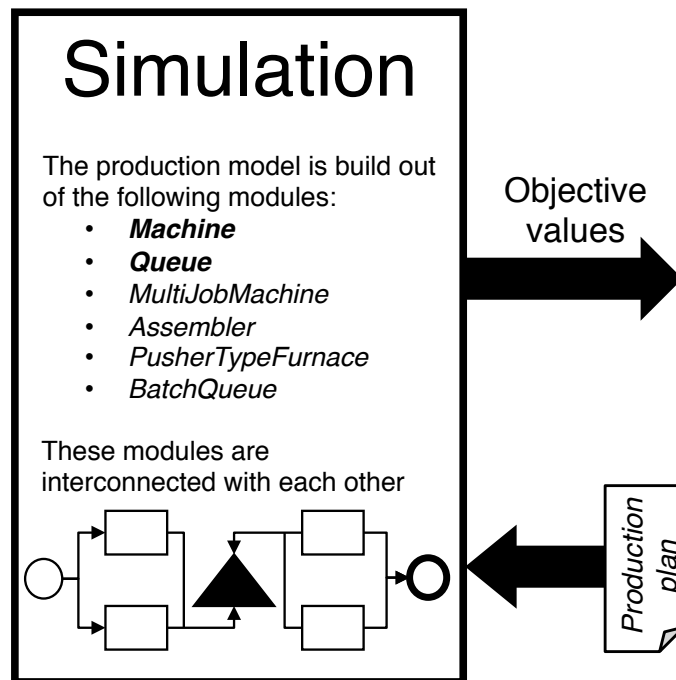


Figure 22: Overview of the simulation module

Furthermore, for some special processes, also modifications of these core elements exist. A *MultiJobMachine* can have more than one job in process at the same time. This module is used to simulate the simplified side printing process, for example. The *Assembler* module is a modification of the *Machine* module with additional functionality. It is considering the setup procedures and also breaks because of staff availability at the assembler. The *PusherTypeFurnace* module simulates the continuous production behavior of a pusher type furnace. All this modules are interconnected with each other to model the simplified production process. The input into the simulation module is a production plan with a given sequence. The outputs of the module are objective values like the makespan or the total WIP (see Section 6.4)

¹ Italicized words are proper names, which are also used in the VBA programming code.

6.2.1 Core simulation modules

The simulation model consists of the two core simulation modules: *Machine* and *Queue*. The basic element within the simulation is a job with a job number and a lot size, which is the stack quantity.

Figure 23 shows the principle of the simulation model in a simple example. Hereby the model consists of two *Machines* (rectangle) with a *Queue* (triangle) in between. Furthermore, also the *FinalStorage* and the *ProductionPlan* are modeled as a *Queue*. A *Queue* is modeled with the first in first out principle using the class *ClsQueue*. Such a queue can store several jobs and can release the first job on demand. The machines are using the class *ClsMachine* with the functionality to load and unload one single job. When a job is loaded to a machine, next to the information about the job number and the lot size, also the machining time is handed over to this module.

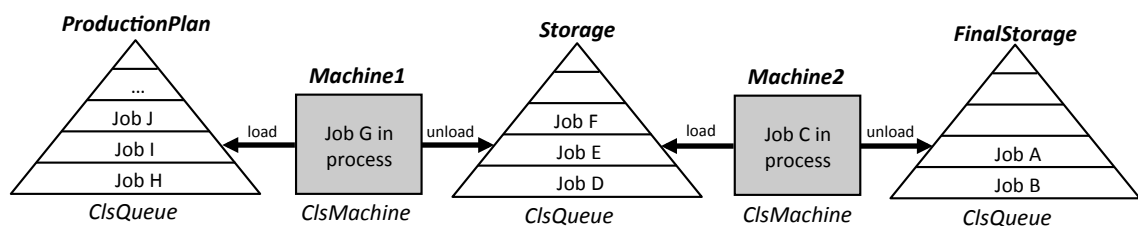


Figure 23: Example of the core simulation modules

When the simulation starts, all jobs are stored in the queue *ProductionPlan*. As *Machine1* is not loaded with a job, *Machine1* demands a job of the queue *ProductionPlan*. *Machine2* does the same but in this case the queue *Storage* is empty. So only *Machine1* gets loaded with the first job out of the queue *ProductionPlan*.

During the run of the simulation the machines are decreasing the machining time of the loaded job every time step of the simulation until the job is finished. Then the machine unloads the job to the next queue.

The simulation runs until the queues *ProductionPlan* and *Storage* are empty and furthermore also the machines are unloaded. Then the simulation returns the collected objective values.

In the next subsections the algorithm of the two core simulation models are described in detail.

6.2.1.1 Class *ClsQueue*

Algorithm 4 shows the class *ClsQueue* in VBA code. The class *ClsQueue* can store several jobs of the class *ClsJob* in the array *mJob()* (see Line 1) and release them with

the first in first out principle. Furthermore, the class stores in the variable *mQueueLength* (see Line 2) its actual length and in the variable *mQueueName* (see Line 3) the name of the queue.

VBA Class: ClsQueue	Algorithm 4
1: Dim mJob(NUMBEROFJOBS) As ClsJob	
2: Dim mQueueLength As Integer	
3: Dim mQueueName As String	
4: Sub EnQueue(Job As ClsJob)	
5: mQueueLength = mQueueLength + 1	
6: Set mJob(mQueueLength - 1) = Job	
7: End Sub	
8: Function DeQueue() As ClsJob	
9: Dim i As Integer	
10: Set DeQueue = mJob(0)	
11: For i = 0 To mQueueLength - 2	
12: Set mJob(i) = mJob(i + 1)	
13: Next i	
14: Set mJob(i).JobNumber = Nothing	
15: mQueueLength = mQueueLength - 1	
16: End Function	
17: Function WorkInQueue() As Integer	
18: Dim i As Integer	
19: WorkInQueue = 0	
20: For i = 0 To mQueueLength	
21: WorkInQueue = WorkInQueue + mJob(i).StackQuantity	
22: Next i	
23: End Function	
24: Function IsNotEmpty() As Boolean	
25: If QueueLength = 0 Then	
26: IsNotEmpty = False	
27: Else	
28: IsNotEmpty = True	
29: End If	
30: End Function	

The class *ClsQueue* has two main functions: *EnQueue* and *DeQueue* (see Figure 24). The sub *EnQueue* (see Line 4) can store a new job on the last position of the queue.

The function *DeQueue* (see Line 8) returns the first job and swaps all other jobs for one position.

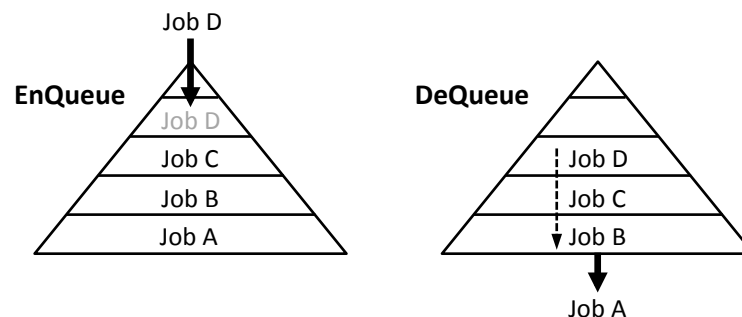


Figure 24: *ClsQueue* functions: *EnQueue* and *DeQueue*

Furthermore, the class *ClsQueue* has the function *WorkInQueue* (see Line 17) which returns the actual amount of stacks of all jobs which are in the queue, also called work in queue (WIQ). The function *IsNotEmpty* (see Line 24) returns in a logic variable if there are jobs in the queue or not.

6.2.1.2 Class *ClsMachine*

The Algorithm 5 shows the class *ClsMachine* in VBA code. The class *ClsMachine* stores the actual production progress of a machine. For that the class contains of the variable *mJob* (see Line 1), which stores the job name and stack quantity and the variable *mProcessProgress* (see Line 2), which stores the actual progress of the job that is machined. Furthermore, in the variable *mMachine* (see Line 3) the type of the machine is stored.

VBA Class: *ClsMachine*

Algorithm 5

```

1: Dim mJob As ClsJob
2: Dim mProcessProgress As Integer
3: Dim mMachine As ClsMachine
4: Function UnloadProcess() As ClsQueueEntity
5:     Set UnloadProcess = mJob
6:     mJob.JobNumber = ""
7:     mProcessProgress = 0
8: End Function
9: Sub LoadProcess(Job As ClsJob, ProcessDuration As Integer)
10:    Set mJob = Job
11:    mProcessProgress = ProcessDuration
12: End Sub
13: Function ProcessLoaded() As Boolean

```

```

14:     If mJob.JobNumber = "" Then
15:         ProcessLoaded = False
16:     Else
17:         ProcessLoaded = True
18:     End If
19: End Function
20: Function InProcess() As Boolean
21:     If mProcessProgress > 0 Then
22:         InProcess = True
23:     Else
24:         InProcess = False
25:     End If
26: End Function
27: Sub ProgressStep()
28:     If InProcess Then
29:         mProcessProgress = mProcessProgress - SIMULATIONSTEP
30:     End If
31: End Sub

```

The sub *LoadProcess* (see Line 9) can load a new job into the machine. On the other hand, the function *UnloadProcess* (see Line 4) unloads a job from the machine. The functions *ProcessLoaded* (see Line 13) and *InProcess* (see Line 20) return the actual state of the machine. The sub *ProgressStep* (see Line 27) can decrease the production time of the job in the machine that is left.

6.2.2 Discrete-event simulation

With the core modules, explained in Section 6.2.1, the simulation model can be build up. The principle of the discrete-event simulation gets explained by using a simple example with two machines shown in Figure 25 (see also Section 6.2.1).

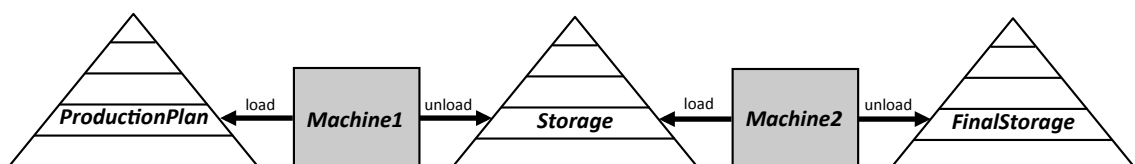


Figure 25: Example for the discrete-event simulation

Algorithm 6 shows the VBA code of the simulation module. The *Simulation* is a function, which returns the objective value in the class *ClsObjectivValue* and gets the

ProductionPlan as an input (see Line 1). The variables *ActualWIP* (see Line 3) and *TotalWIP* (see Line 4) are used to calculate the *TotalWIP*. The variable *SimulationTime* (see Line 6) is used as the clock for the simulation loop. Furthermore, the queues and the machines get declared using the classes *ClsQueue* and *ClsMachine* (see Line 8-15).

VBA Function: Simulation**Algorithm 6**

```

1: Function Simulation(ProductionPlan As ClsQueue,
                        SimulationStep As Integer) As
                        ClsObjectivValue
2:     Set Simulation = New ClsObjectivValue
3:     Dim ActualWIP As Integer
4:     Dim TotalWIP As Long
5:     TotalWIP=0
6:     Dim SimulationTime as Long
7:     SimulationTime = 0
8:     Dim Machine1 As ClsMachine
9:     Set Machine1 = New ClsMachine
10:    Dim Storage As ClsQueue
11:    Set Storage = New ClsQueue
12:    Dim Machine2 As ClsMachine
13:    Set Machine2 = New ClsMachine
14:    Dim StorageFinal As ClsQueue
15:    Set StorageFinal = New ClsQueue
16:    Do
17:        ActualWIP = 0
18:        ActualWIP = ActualWIP + RunMachine(ProductionPlan,
                                                Machine1, Storage)
19:        ActualWIP = ActualWIP + RunMachine(Storage, Machine2,
                                                StorageFinal)
20:        ActualWIP = ActualWIP + Storage.WorkInQueue
21:        TotalWIP = TotalWIP + ActualWIP
22:        SimulationTime = SimulationTime + SimulationStep
23:    Loop Until Not Machine1.ProcessLoaded
                    And Not Machine2.ProcessLoaded
                    And Not Storage.IsNotEmpty
                    And Not ProductionPlan.IsNotEmpty
24:    Simulation.Makespan = SimulationTime
25:    Simulation.TotalWIP = TotalWIP * SimulationStep
26: End Function

```

The main part of the function *Simulation* is the *Do Until* loop (see Line 16-23). In this loop the simulation model is build up with the *RunMachine* function (see also Algorithm 7), which defines for a machine the queue before the machine and the queue after the machines. The machine loads the jobs from the queue before the machine and unloads the jobs to the queue after the machine. For example, *Machine1* loads the jobs from the queue *ProductionPlan* and unloads the jobs to the queue *Storage* (see Line 18). Furthermore, the *RunMachine* function returns the WIP of the machine. This WIP gets stored for each *SimulationTime* in the variable *ActualWIP* and get summarized in the variable *TotalWIP* (see Line 21) to calculate the total WIP objective function. In Line 22 the *SimulationTime* gets increased by the variable *SimulationStep* which is an integer number greater than 1. The simulation loop runs until the machines are unloaded and also the *ProductionPlan* and the *Storage* queue are empty (see Line 23). Then the objective values makespan and total WIP get returned (see Line 24-25).

Algorithm 7 shows the VBA code of the function *RunMachine*, which is used within the simulation module.

VBA Function: RunMachine
Algorithm 7

```

1: Function RunMachine (QueueBeforeMachine As ClsQueue,
                        Machine As ClsSimMachine,
                        QueueAfterMachine As ClsQueue) As Integer
2:   Dim QueueEntity As ClsQueueEntity
3:   RunMachine = 0
4:   If Not Machine.InProcess Then
5:     If Machine.ProcessLoaded Then
6:       Set QueueEntity = Machine.UnloadProcess
7:       Call QueueAfterMachine.Enqueue (QueueEntity)
8:     End if
9:     If QueueBeforeMachine.IsNotEmpty Then
10:      Set QueueEntity = QueueBeforeMachine.DeQueue,
11:      Call Machine.LoadProcess (QueueEntity,
                                ProductList.GetProcessData (QueueEntity.JobNumber,
                                                                Machine.Machine.ProcessName).ProcessTime)
12:      RunMachine = Machine.StackQuantity
13:     End If
14:   Else
15:     RunMachine = Machine.StackQuantity
16:   End If
17:   Call Machine.ProgressStep
18: End Function

```

The function *RunMachine* has three main tasks. First of all it defines the queues before and after a machine, second it handles the loading and unloading of jobs of the machine and third it returns the actual WIP of the machine.

At the start the function checks if the machine is in process (see Line 4). If the machine is not in process then the function checks if the machine is loaded with a job (see Line 5). When a machine is loaded but not in process this means that the loaded job is finished. Therefore, the job gets unloaded from the machine (see Line 6) and gets loaded in the queue after the machine (see Line 7).

If the machine is not in process and the queue before the machine is not empty (see Line 9) then the machine gets loaded. For that a job gets unloaded from the queue before the machine (see Line 10) and loaded to the machine (see Line 10). To load a process to the machine also the machining time of the job on the machine is needed. For that this data is loaded from the *ProductList* with the function *GetProcessData*.

Finally, the function *RunMachine* also calls the function *ProgressStep* (see Line 17) of the machine to simulate a time step.

6.2.3 Implemented simulation model

In Section 6.2.2 the principle of a DES model with the basic elements machine and queue gets described. The same principle can be applied by implementing such a DES on the simplified MLV frontend production process (see Figure 20).

The simulation model of the simplified MLV frontend production process, described in Section 5.4, is shown in Figure 26. This simulation model is using deterministic machining and setup times and is furthermore not considering any machine disturbances.

The simulation model has two different modes: an optimization mode and an output mode. The optimization mode is used within the optimization algorithm to decrease the computing time. This mode does not simulate processes, which have the same machining time for all products. For example, the lay time is for all products the same. Therefore, the computing time for one simulation can get decrease by leaving such processes out. Beside, the lay time also the sieve and transportation process and the binder burn out is not simulated in this mode. The output mode simulates all machines and generates further data used for the outputs shown in Section 6.5.2. Table 9 shows the difference in the computing time of the two simulation modes.

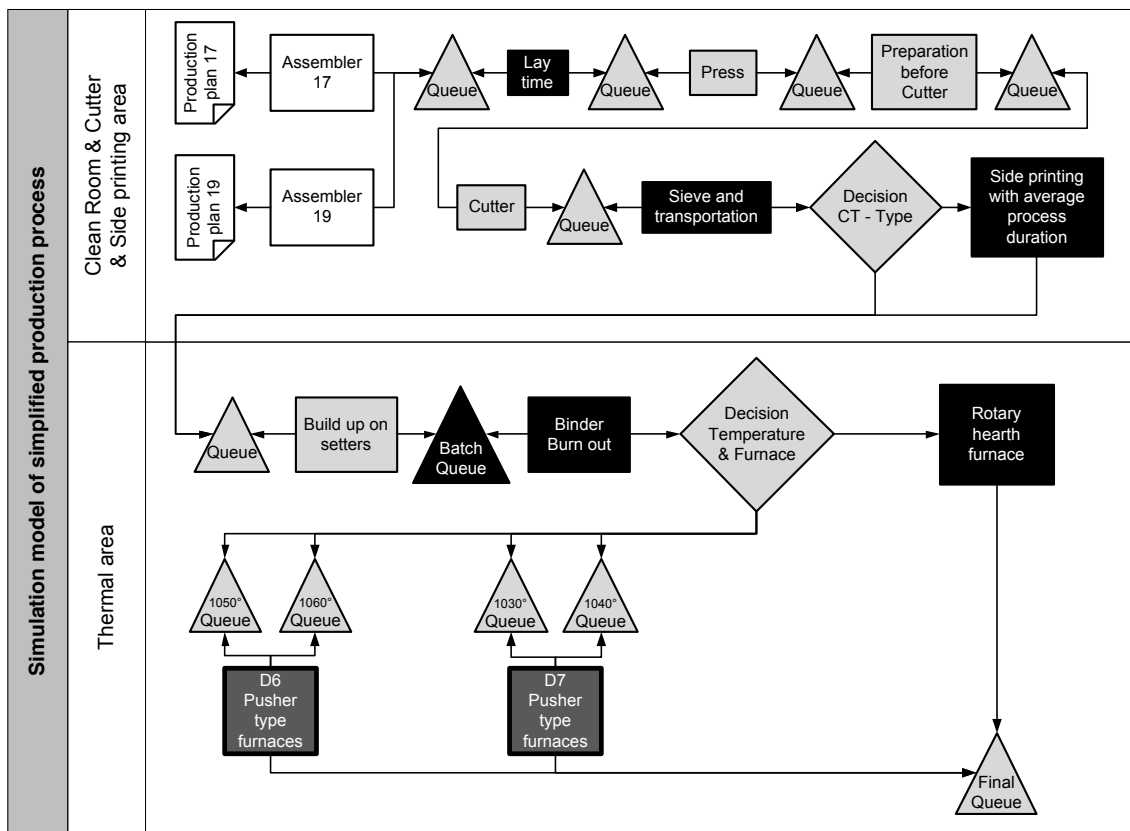


Figure 26: Simulation model of the simplified production process (see Figure 20)

The simulation model shown in Figure 26 simulates the simplified production process using the simulation modules described in Table 8.

Simulation module	Description
Machine	Basic machine module described in Section 6.2.1.2
Assembler	Modification of the machine module with the additional functionality to considering the setup procedures and also breaks because of personal availability.
Multi job machine	Modification of the machine module, which can process more than one job at once.
Pusher type furnaces	Modification of the machine module with the additional functionality to model the continuous production behavior of a pusher type furnace.

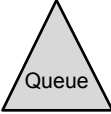

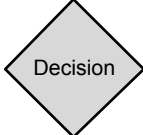
	Basic Queue module described in Section 6.2.1.1
	Modification of the Queue module with the limitation that jobs can only be released when a certain batch size is reached.
	The decision module makes assignments based on various product data.

Table 8: Description of all simulation modules

Furthermore, the increase of the simulation clock and thereby the amount of discrete time points within the simulation can be adjusted with the variable *SimulationStep*. This affects the accuracy of the objective values and also the computing time. Table 9 shows the simulation results for different values of the variable *SimulationStep* in calendar week 5 of 2012. In this week the production plan had 110 job and 27 setup clusters.

	<i>SimulationStep</i> (see Algorithm 6 Line 22)			
	1 min	2 min	3 min	4 min
Output mode	82 sec	42 sec	29 sec	21 sec
Optimization mode	7 sec	4 sec	2,5 sec	1,5 sec
Makespan	20149 min	20438 min	20499 min	20560 min
Total WIP	114295985	113744320	113812485	113975860
Failure Makespan	-	1,43 %	1,74 %	2,04 %
Failure Total WIP	-	- 0,48 %	- 0,42 %	- 0,28 %

Table 9: Influence of the variable *SimulationStep* on the computing time and the objective values

This simulation was done on an Intel Core 2 Duo P8400 CPU in Excel 2003. Unfortunately Excel 2003 VBA does not have multiprocessor support so this simulation was running on one of the two CPU cores at 2,26 GHz.

6.3 Optimization

The optimization model is using the meta heuristic improvement method Simulated Annealing (see Section 4.4.3.1) and consists of an acceptance algorithm and a permutation algorithm.

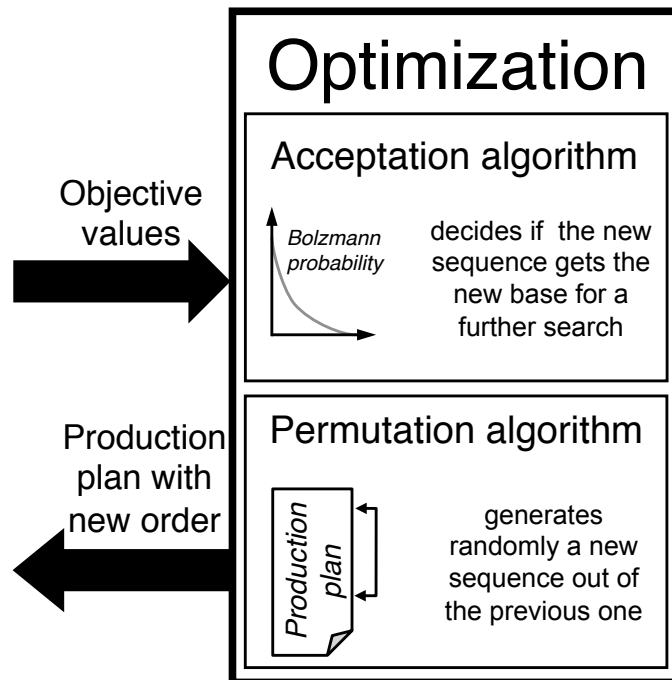


Figure 27: Overview of the optimization module

The input into the optimization module is an objective value for a certain production plan order, calculated from the simulation. The output of the module is a production plan which order is modified randomly out of the previous production plan.

6.3.1 Acceptance algorithm

The acceptance algorithm decides if the new order of the production plan gets the new basis for a further search. The theoretical mathematical description can be found in Algorithm 2 in Section 4.4.3.1. The Algorithm 8 shows the implementation in VBA code.

VBA Sub: SimulatedAnnealing

Algorithm 8

```

1: Sub SimulatedAnnealing(AmountSteps as Integer)
2:     Dim OldObjectiveValue as ClsObjectiveValue
3:     Dim ObjectiveValue as ClsObjectiveValue

```

```
4:      Dim ProductionPlan as ClsQueue
5:      Dim OldProductionPlan as ClsQueue
6:      Dim StartTemperature as Double
7:      Dim DeclineFactor as Double
8:      Dim Temperatur as Double
9:      Dim RandomNumber as Double
10:     Dim BoltzmannProbability as Double
11:     Set ProductionPlan = ReadProductionPlan
12:     Set OldObjectiveValue = Simulation(ProductionPlan)
13:     StartTemperature = -OldObjectiveValue.Rated * 0.005/Ln(0.7)
14:     DeclineFactor = (1 / 500) ^ (1 / AmountSteps)
15:     Temperature = StartTemperature
16:     For i = 1 To AmountSteps
17:         Call StoreAndSwitchPosition(ProductionPlan,
                                     OldProductionPlan)
18:         Set ObjectiveValue = Simulation(ProductionPlan)
19:         If ObjectiveValue.Rated > OldObjectiveValue.Rated Then
20:             Randomize
21:             RandomNumber = Rnd()
22:             BoltzmannProbability = Exp(-(ObjectiveValue.Rated -
                                         OldObjectiveValue.Rated) /
                                         Temperature)
23:             If RandomNumber > BoltzmannProbability Then
24:                 Set ProductionPlan = OldProductionPlan
25:                 Set ObjectiveValue = OldObjectiveValue
26:             Else
27:                 Set OldObjectiveValue = ObjectiveValue
28:             End if
29:         Else
30:             Set OldObjectiveValue = ObjectiveValue
31:         End if
32:         Temperature = Temperature * DeclineFactor
33:     Next i
34: End Sub
```

The optimization starts in Line 12 with an initial simulation of a production plan. The return values of the simulation are stored in the variable *OldObjectiveValue*. Out of this first simulation the start temperature gets calculated with the Equation 11 (see Line 13). Also the decline factor of the temperature in Equation 12 gets calculated (see Line 14). The constant values for these two calculations are determined out of experience and

deliver good optimization results. With the *for* loop in Line 14 the iteration start until a certain amount of iterations (variable *AmountSteps*) is reached. The first step within the loop is the execution of the function *StoreAndSwitchPosition*. This function represents the permutation algorithm, which will be explained in detail in the Section 6.3.2. In principle this function stores the previous production plan in the queue *OldProductionPlan* and randomly switches the position of 2 jobs in the queue *ProductionPlan*. Then the production plan with the new order gets simulated (see Line 18). The *if* query in Line 19 checks if the objective value of the new order is better than the previous order.

If the objective value of the new order is worse than the previous one, the new order should be accepted with the Boltzmann probability (see Equation 10). For that the probability is calculated in Line 22 and compared with a random number. If the random number is bigger than the Boltzmann probability the new order get quashed and the previous production plan and also the objective value get restored (see Line 24-25). When the random number is smaller than the Boltzmann probability the new order get accepted and is the base for a further search (see Line 27).

If the objective value of the new order is better than the previous one, the new order gets accepted instantly (see Line 30).

Finally, a certain decline factor decreases the temperature after each iteration step. (see Line 32).

6.3.2 Permutation algorithm

The task of the permutation algorithm is to generate a new job sequence out of a given one. For that the permutation rule modifies the production plan into a neighbor ship solution. There are several strategies to find a suitable neighbor ship. The easiest one is just a two variable exchange, meaning two jobs get exchanged. With some modifications, this strategy is also used in the production scheduling tool.

The reason for these modifications is the setup situation at the assembler (see Section 5.2.2.1). Because of the high setup costs at the assembler, all jobs that need the same screen and paste should be produced at once. Therefore, all jobs that need the same type of screen and paste in the screen printing process are combined in one setup cluster. Furthermore, two production plans are necessary because the production consists of two assemblers. Figure 28 shows in an example two production plans with 12 jobs each.

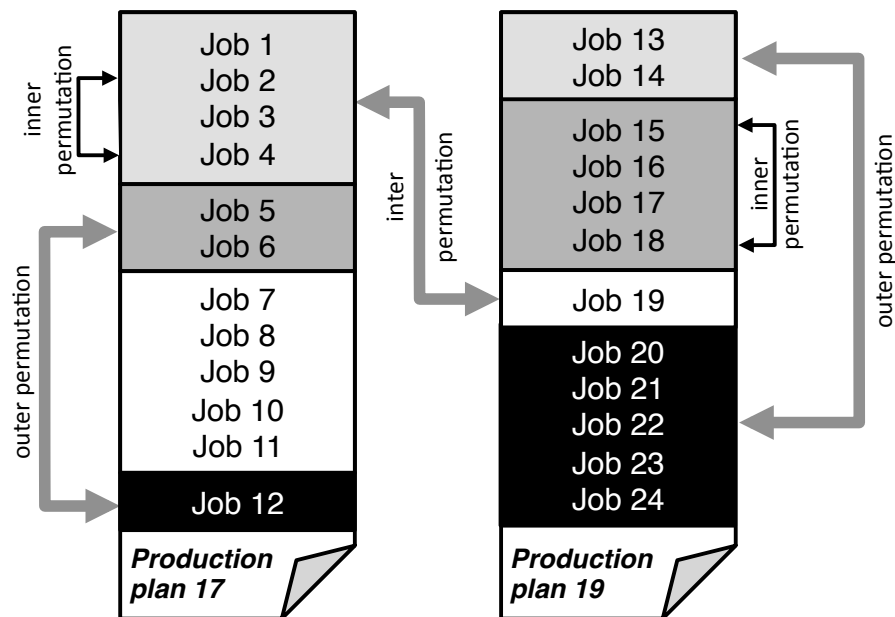


Figure 28: Example of the inner-, inter- and outer permutation

The underlying gray scale pictures a setup cluster. For example in the production plan 17 on the left side, the jobs 7 – 11 are in one cluster. The permutation algorithm has three different strategies: the inner-, the inter- and the outer permutation.

The inner permutation randomly switches jobs within a setup cluster. The outer permutation switches one cluster with another one within the production plan. Furthermore, the inter permutation switches one cluster from a production plan to another production plan. The permutation algorithm randomly chooses one of these permutation strategies in every iteration. Thereby, the algorithm generates two new production plans.

With this approach also the search space¹ get narrowed. For example, a typical production plan has about 90 jobs. This leads to $90! \approx 10^{138}$ possibilities of arranging the jobs. By clustering the 90 jobs into 30 setup cluster the search space has just $30! \approx 10^{32}$ possibilities², which is a dramatic decrease in the amount of possibilities. Therefore, this method leads to a faster optimization by having the same or even, because of lack of computing time, a better solution.

¹ The search space of an optimization problem is the set in which the optimization algorithm searches for an optimum solution.

² The search space of 10^{32} possibilities only considers the outer and the inter permutation.

6.4 Objective value

In the literature many scheduling problems deal with the minimization of the makespan. However, there are more variables, which can be taken into account. This is shown in the simple example below.

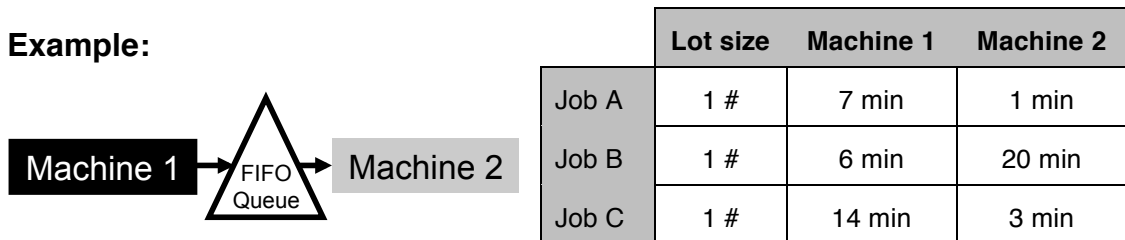


Figure 29: Production process

Table 10: Production plan

In this example the production process consists of two machines with a first in first out (FIFO) storage in between (see Figure 29). Furthermore, the production plan has three jobs: A, B and C. These jobs have all a lot size of one piece and they have various machining times (see Table 10).

One main objective value is the makespan. This is the duration it takes to produce all three Jobs. Figure 30 visualizes in a Gantt chart the interconnections of the production plan with the order ABC. The production starts with job A at machine 1 (black). After the machining time of 7 minutes job A is finished at machine 1 and transferred to machine 2 (grey). Now job B can be machined on machine 1. Furthermore, job A is now machined on machine 2 for 1 min. The situation for job C is a little bit different. Job C starts at machine 1 after job B is finished on this machine. After 14 minutes machining time of job C at machine 1 the job C still has to wait for 6 minutes until job B is also finished at machine 2.

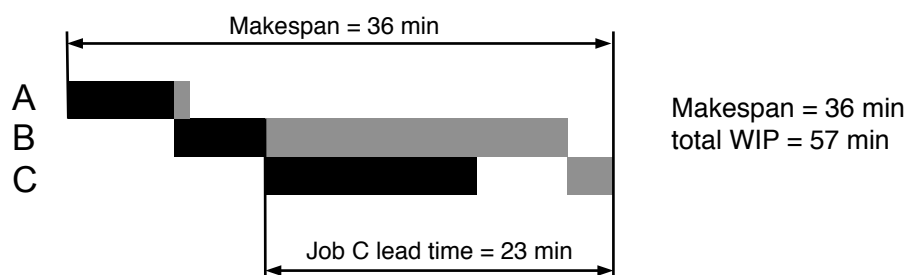


Figure 30: Gantt chart of order ABC

An indicator for this in production inventory is the total WIP. The total WIP can be calculated with the sum of each job lead-time multiplied with the lot size.

$$total\ WIP = \sum_{j=1}^{jobs} lead\ time_j * lot\ size_j \tag{13}$$

In this example the minimum total WIP is 51 minutes, which is the sum of all machining times. If the lead-time of a job gets higher than the sum of its machining times, because the job has to wait until the next machine is available, also the total WIP increases. The lot size indicates the value of a job that means a job with a higher lot size has a higher value. Therefore, the lot size is multiplied with the lead-time in the total WIP calculation.

Beside, the sequence ABC also 5 other sequences exist. All six possible sequences with their objective values are shown in Figure 31.

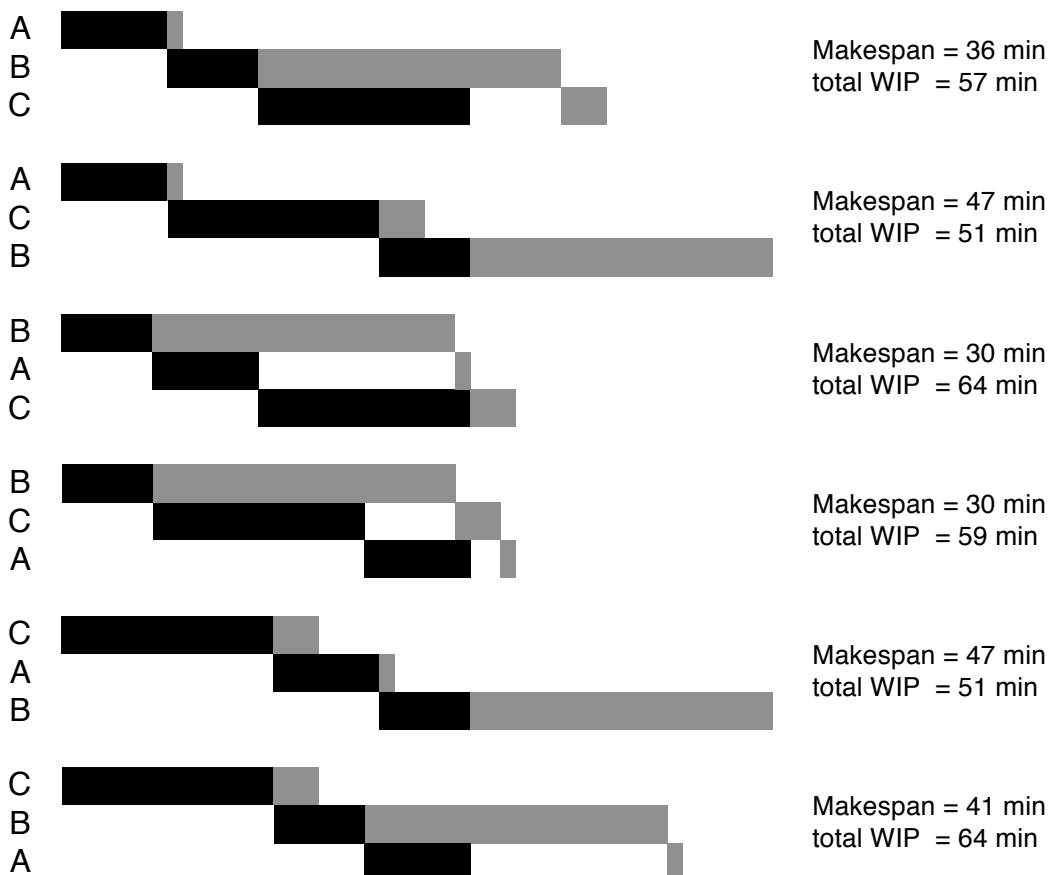


Figure 31: All six possible sequences

The order ACB and CAB, have the lowest total WIP but also the highest makespan. BAC as well as BCA have the lowest makespan and therefore the utilization of machine 2 (grey) is also high. However, the combination BCA has a lower total WIP than BAC

because the waiting time of the jobs in the storage is with 8 minutes in BCA a bit lower than 13 minutes in BAC. Therefore, the combination BCA is the preferred order in this example.

As in this example is shown, beside the makespan, also the total WIP is an indicator for a good scheduled production plan. To make it possible that several indicators get combined into one objective value the indicators have to get scaled and rated. To scale an indicator, its lowest possible value or a reference value divides the indicator. Afterwards the scaled indicator can get rated with a factor.

$$f(S) = \sum_{i=1}^{indicators} \frac{x_i}{x_{i\ min}} * k_{x_i} \quad (14)$$

with $f(S)$ objective value
 x_i value of indicator i
 $x_{i\ min}$ lowest possible value of indicator i
 k_{x_i} factor for rating of x_i

For the previous example this would lead to an objective function shown below which should be minimized.

$$f(S) = \frac{Makespan}{Makespan_{min}} * k_{Makespan} + \frac{total\ WIP}{total\ WIP_{min}} * k_{total\ WIP} \quad (15)$$

6.4.1 Implemented objective value

For the production scheduling tool following objective value is implemented.

$$f(S) = \frac{MakespanAssembler}{MakespanAssembler_{min}} * k_{MakespanAssembler} \quad (16a)$$

$$+ \frac{MakespanCutter}{MakespanCutter_{min}} * k_{MakespanCutter} \quad (16b)$$

$$+ \frac{total\ WIP}{total\ WIP_{min}} * k_{total\ WIP} \quad (16c)$$

$$+ \frac{total\ WIQ}{total\ WIQ_{min}} * k_{total\ WIQ} \quad (16d)$$

The first part of the objective value is minimizing the makespan of the assembler (see Equation 16a). This indicator is needed to ensure that both assemblers are well utilized. As the inter permutation algorithm is switching setups clusters also between the two assemblers it can happen that total production time of each assembler is various.

The objectives in Equation 16c and Equation 16d are used to minimize the inventory. On the one hand with the total WIP, which also includes in process inventory and on

the other hand with the total WIQ, which is just inventory waiting in storage. Without the makespan assembler objective the optimization algorithm would shift several jobs to one of the two assemblers to fulfill this inventory objective.

Furthermore, the implemented objective value consists of the objective to minimize the makespan of the cutter. Until the cutter more or less all jobs have equal production times. After this process all CT type MLVs have to be processes through the side printing process. This leads to the fact that some jobs have a much higher total lead-time (see Table 11).

	lead-time until cutter	total lead-time
average job non CT type	30 hours	120 hours
average job CT type	30 hours	200 hours

Table 11: Comparison of average lead-times

In a classical makespan optimization, meaning optimizing the total production time, the algorithm would try to schedule all CT type in the beginning to come to a short total makespan. As such an approach is undesirable only the makespan until the cutter gets minimized (see Equation 16b).

All the single indicators in the objective value get scaled and rated like it is shown in Equation 14.

6.5 Graphical user interface

6.5.1 Input

The input into the production scheduling tool is an Excel table with the weekly production jobs and their lot size. Further this table contains information about related product data and the cluster assignment (see Table 12). The jobs in this list are ordered in a way that the jobs, which need the same screen and paste, are in a row and a setup cluster number gets assigned manually. Out of the product data the machining times for each process can be calculated. Also pause times of the assembler can be set. In the Attachment 3 a screenshot of the input of the production scheduling tool can be found.

Inputs into the production scheduling tool			
Job number	Lot size (Amount of stacks)	Product data (case size, type,...)	Machining times for all processes
			Setup Cluster

Table 12: Inputs into the production scheduling tool

6.5.2 Output

Gantt Chart

Beside, the start order of the jobs, the production scheduling tool also visualizes the production in a Gantt chart. Figure 32 shows a cutout of such a chart. On the vertical axis several production jobs are listed. The horizontal axis indicates the timeline. The bars in the chart visualize the machining and storage times for each job. In this figure the grey bars visualize a machine and the black once a storage queue.

In Attachment 4 and 5 a screenshot of the Gantt chart of the production scheduling tool can be found. This screenshot shows the Gantt chart in more detail. Several different production processes are visualized in various colors. Therefore, this Gantt charts helps the operator of the production scheduling tool to check the calculated sequence visually.

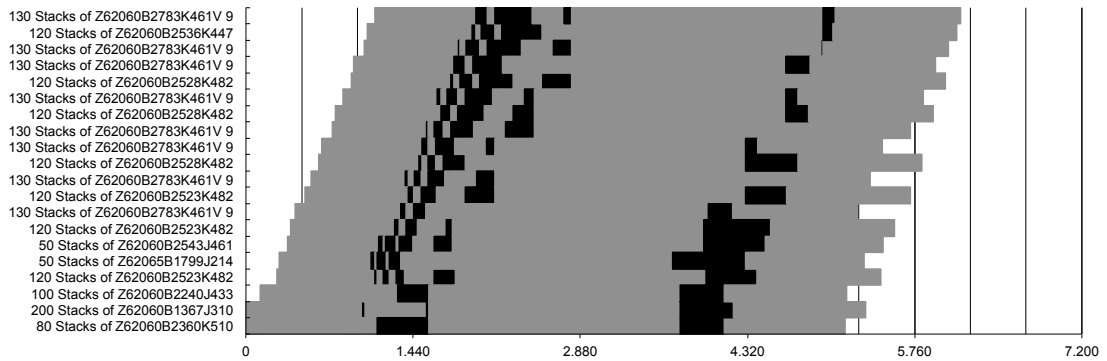


Figure 32: Cutout of the Gantt chart

Side printing process

Furthermore, the side printing process gets visualized in detail. Figure 33 shows the inflow of stacks into the side printing process of different case sizes over the time.

This chart allows the operator of the production scheduling tool to verify if the calculated sequence will lead to problems in the side printing area, which is not simulated in detail in the scheduling tool. In the Attachment 6 a screenshot of the visualization of the side printing process can be found.

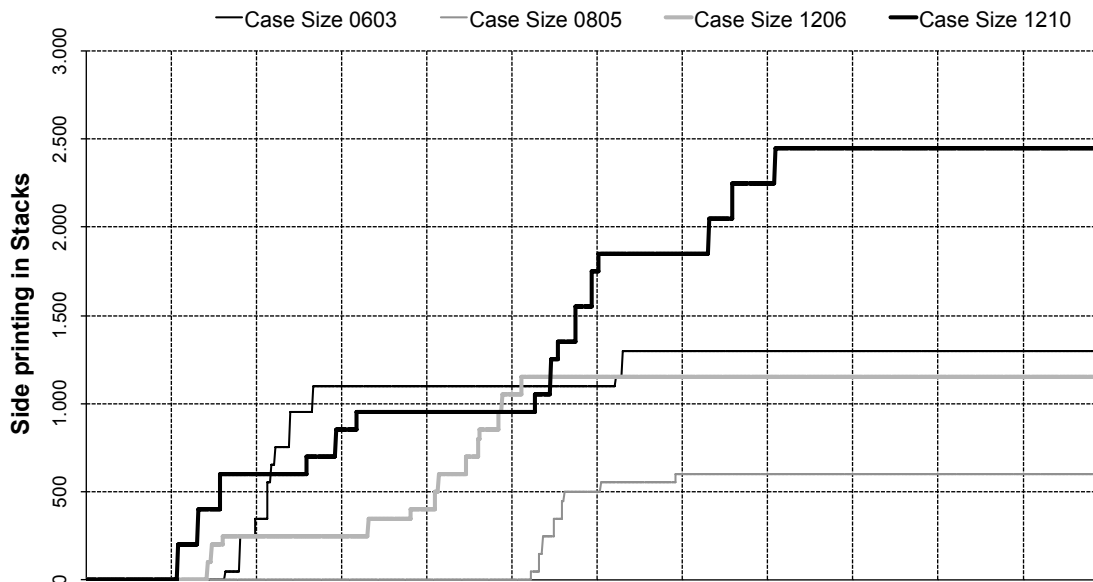


Figure 33: Visualization of the side printing process

Pusher type furnaces

Also the pusher type furnaces get visualized. Figure 34 shows the production of the D6 pusher type furnace for a weekly production plan. The thick grey line indicates the WIP in the furnace in stacks. The thin black and grey lines visualize the work in queue (WIQ) in the storage before the furnace for different temperatures. And also the actual furnace temperature is visualized with the thick black line. This chart mainly shows the operator the utilization of the furnace and visualizes the inventories before the furnace. In the Attachment 7 a screenshot of the visualization of the pusher type furnaces can be found.

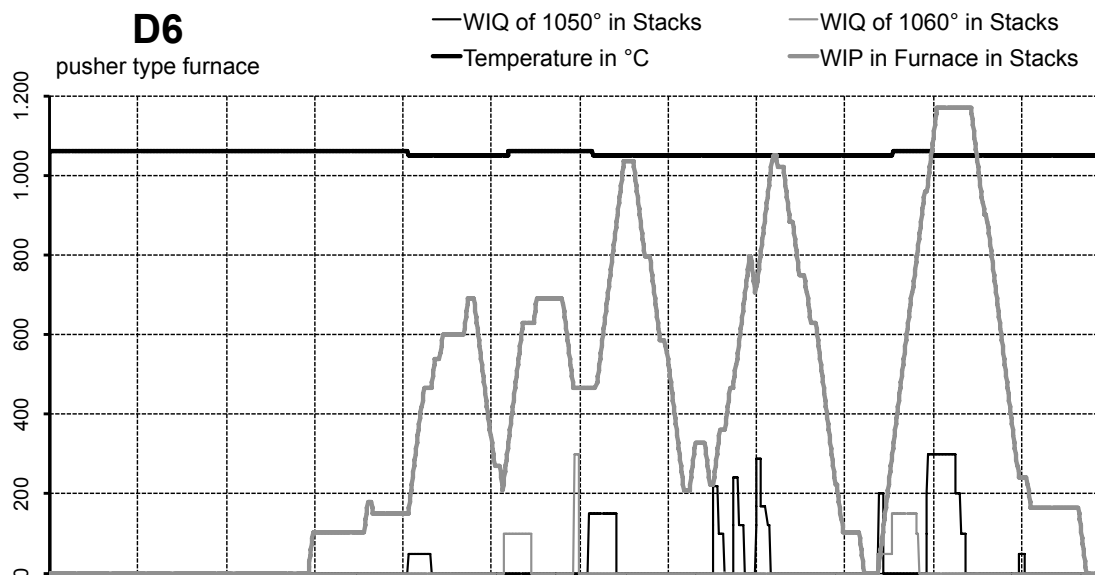


Figure 34: Visualization of the pusher type furnace

Output of the frontend production

Also the output of the frontend production gets visualized. Figure 35 shows the production outflow of stacks and also single chips. This chart allows the operator of the production scheduling tool to verify if the calculated sequence will lead to delivery problems of the backend. In the Attachment 8 a screenshot of the visualization of the output of the frontend production can be found.

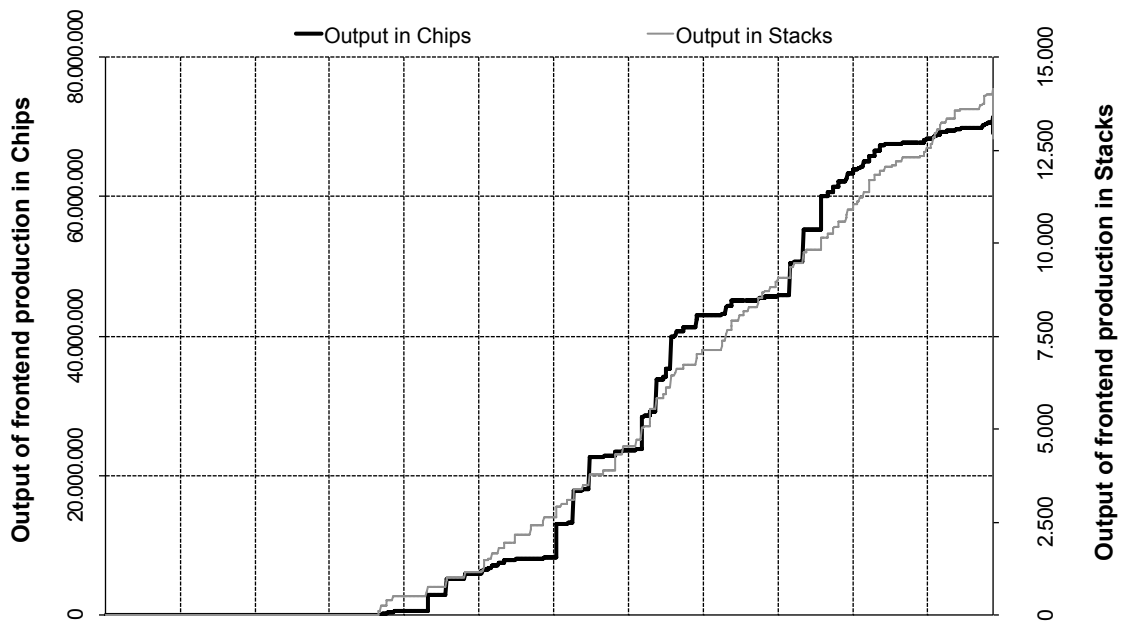


Figure 35: Visualization of the output of the production

Chronological starting sequence

Finally, the optimized starting sequence gets issued in a table (see Table 13). This table contains for each job an assignment to one of the two assemblers and also the optimized sequence for all jobs.

Chronological starting sequence				
Assembler number	Job number	Lot size (Amount of stacks)	Product data (case size, type,...)	Day and shift

Table 13: Chronological starting sequence

6.6 Testing results

The production scheduling tool was tested during the implementation for several times to find the right objective values as well as a suitable permutation algorithm. Hereby it turned out that the implementation of the permutation algorithm using setup cluster is very helpful to reduce the search space and thereby come to results faster. Beside that, a key aspect for good optimization results is an accurate simulation model. Because of the limitations of Microsoft Excel it was only possible to implement a simplified model of the production process, which uses deterministic process times. But still this model delivers an accurate replication of the real production process.

Qualitative benefits of the production scheduling tools are the standardization and objectification of the production scheduling. Through the software tool it can be clearly defined which objectives should be optimized and furthermore the influence of these targets can also be rated with factors. However, the optimization is using a heuristic approach and that means that the optimized production sequence does not have to be the same if several optimization runs are compared. Therefore, the visual outputs of the production scheduling tool with the Gantt chart and the further visualizations of certain important areas should help the operator to verify the optimization result.

Beside that, also quantitative improvements could be achieved (see Table 14).

		CW06 2012	CW07 2012 ¹	CW08 2012
Number of jobs		80	91	76
Actual sequence	WIP [min * Stacks]	93918465	106615455	91025160
	WIQ [min * Stacks]	12884970	13436820	16849890
	Assembler setups	31	36	34
Optimized sequence	WIP [min * Stacks]	82923345	96173550	84298455
	WIQ [min * Stacks]	5589300	7572270	6511650
	Assembler setups	29	28	26
Reduction of WIP [%]		12%	10%	7%
Reduction of WIQ [%]		56%	43%	61%

Table 14: Examples of the testing results of the production scheduling tool²

¹ Detailed results of this week can be found in the attachment.

² Simulation settings: *SimulationStep* = 3

Objective function settings: $k_{MakespanAssembler} = 1$, $k_{MakespanCutter} = 1$, $k_{totalWIP} = 1$, $k_{totalWIQ} = 2$
Average optimization time = 45min with 500 iterations

Through the use of the production scheduling tool the work in process (WIP) could be reduced by around 10% at planning level. Furthermore, also the work in queue (WIQ) could be reduced by in average above 50%. The reduction emerges just through the optimized sequence and gets calculated by a comparison with the actual production sequence.

However, the optimized sequence is still not the best possible schedule for the whole production chain, because the simulation model of the production scheduling tool is only simulating the production processes in the frontend. Thereby, two main issues are not covered in the simulation model: First of all the product mix in the side printing process and second the output of the frontend production, which should be continues to avoid stockpiling in the backend production.

One possible solution to cover also these issues would be the implementation of a storage queue as a flow regulator (see Figure 36).

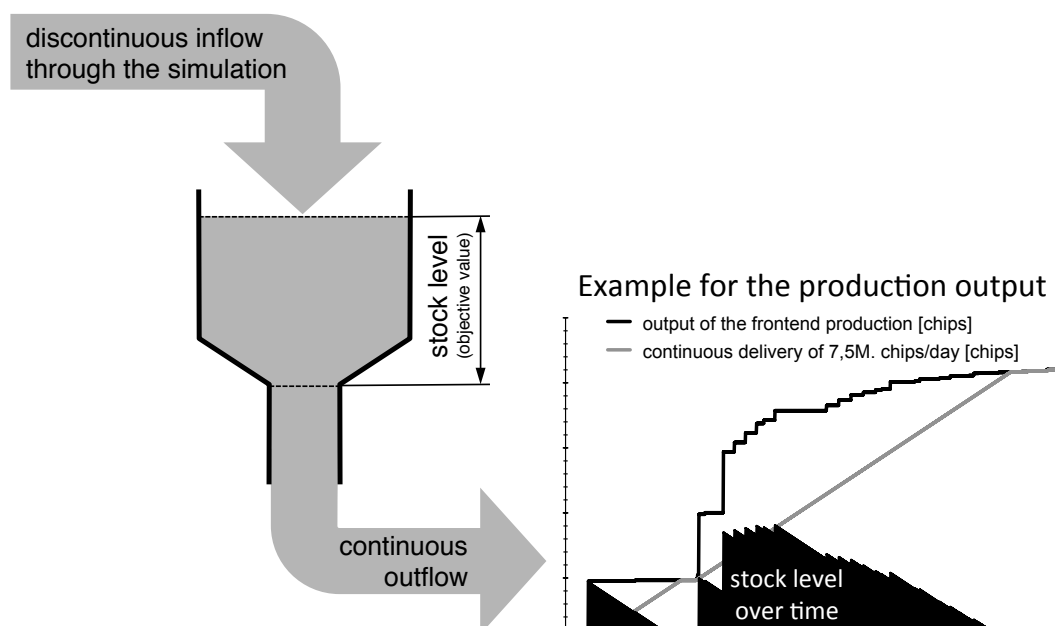


Figure 36: Possibility of simulating the side print process and the production output

Such a storage queue could be implemented using information from the simulation like the production output. This discontinues flow could be passed into a storage, which has a continuous outflow. The differences between the inflow and the outflow can be covered in the stock level, which can be measured over the time. Thereby, also the side printing process and also the output of the frontend production could be implemented into the objective function.

Furthermore, the optimization is working offline, which means that the production scheduling tool has no information about the actual situation of the production system. Section 7 gives a perspective for an online production scheduling tool.

7 Perspective

The production scheduling tool is using a classical offline optimization. This means that the production scheduling tool has no interaction with the real production system. The optimization tool is using a predefined model and the optimization process is done once for the production demand of the upcoming week.

Beside, this classical offline approach, an online optimization is also possible. Thereby, the optimization tool has the actual production status as an input and the tool does not need that much knowledge about future events. In online production scheduling the optimization tool is integrated into the production system like a closed loop control and is running all the time [Heib & Nickel 2011].

7.1 Online production scheduling

Figure 37 shows the architecture of an online production scheduling system.

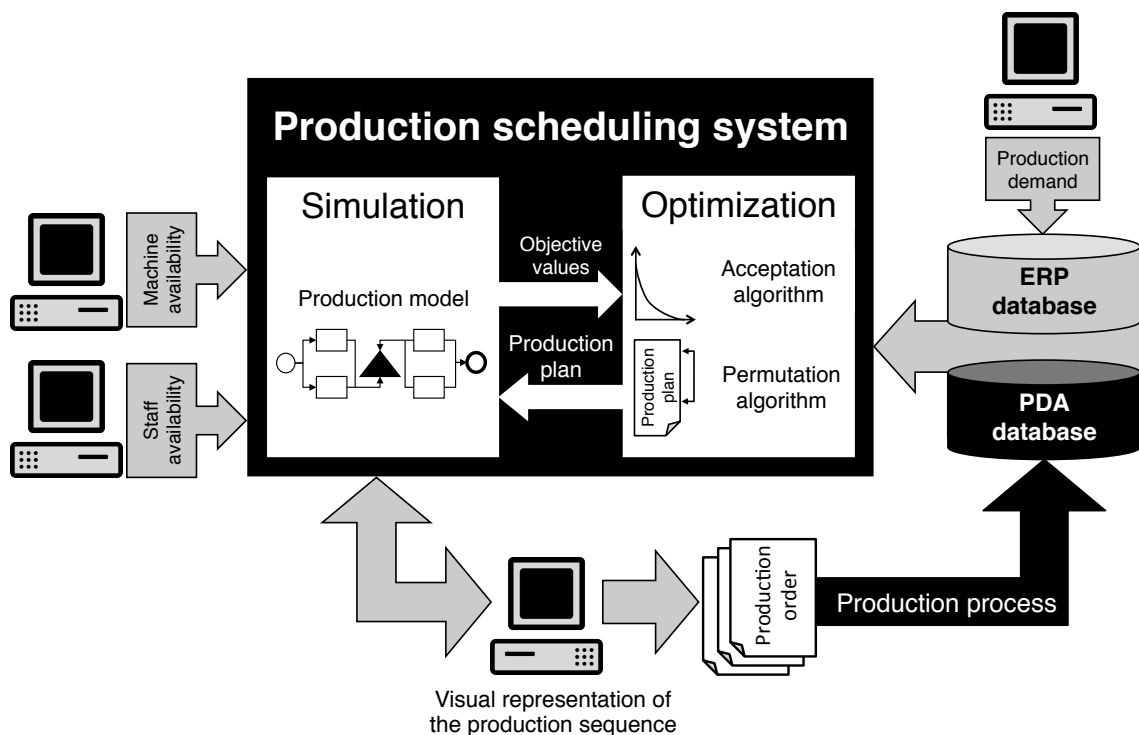


Figure 37: Architecture of an online production scheduling system

Thereby, the optimization tool is integrated into the IT landscape of the production system. The main inputs come from the Enterprise Resource Planning (ERP) database with the production demand and the associated process times. Furthermore, the optimization tool has information about the actual situation of the production through

the Production Data Acquisition (PDA) database. This information is needed to close the control loop.

Additionally, the availability of the staff and machines can be set online and thereby the production model within the simulation can be adjusted to the actual situation.

The output of the online production scheduling system is a visual representation of the production sequence like in an offline system. Hereby the operator can make adjustments, which lead to further optimizations. A typical adjustment would be to prioritize a rush order. Finally, the operator can also release production orders. This production orders will give new feedback over the PDA system into the production scheduling system.

Heib & Nickel [2011] attested an online production scheduling system better performance than on offline system. Hereby it should be marked out that an online production scheduling system needs less computing time than an offline system. The reason for that is that in online optimization the scheduling problem is split up into smaller sub problems with a shorter planning horizon.

7.2 Case Studies

This section gives an overview of selected case studies from the book “Simulation und Optimierung in Produktion und Logistik” from März [2011].

7.2.1 Simulation based optimization in semiconductor production

This case from Klemmt [2011] gives an insight into a scheduling tool of a semiconductor manufacturer

Initial situation

The aim of this case study was to develop a simulation based planning and optimization system for a semiconductor backend production located in Dresden. Therby, this system should deliver optimized sequences for short planning horizons. The runtime of the optimization should be below 5 minutes and the system should be integrated into the existing IT architecture. Maintenance cycles, setup procedures as well as staff planning should be covered in the optimization.

Objectives

- Detailed sequence for all lots as well as start times for each machines
- Maximize the throughput on bottleneck machines
- Minimize the average lead-time of all lots

Approach

The simulation system simcron MODELLER is used in combination with an online optimization using heuristic algorithms. The duration of the project was several years. Within the project the objectives could be fulfilled and the production scheduling system is now also implemented in other production locations.

7.2.2 Production control using simulation based heuristic optimization

This case from Gruber [2011] gives an insight into a production control system of voestalpine Schienen GmbH.

Initial situation

The rail production of voestalpine is heavily depending on the order situation. Unexpected disturbances and large deviations from the production plan lead to insufficient delivery reliability and therefore to additional costs. The optimization tool should be used to increase the delivery reliability and also leads to a more even utilization of the roller mill.

Objectives

The main targets are: increase the delivery reliability, decrease inventory costs and increase the utilization of pivotal machines and buffer.

The delivery reliability is covered with a penalty cost function (see Figure 38). On the left side of this function additional costs arise, because of the inventory holding. The right side of the cost function covers the direct and indirect costs of a delivery delay.

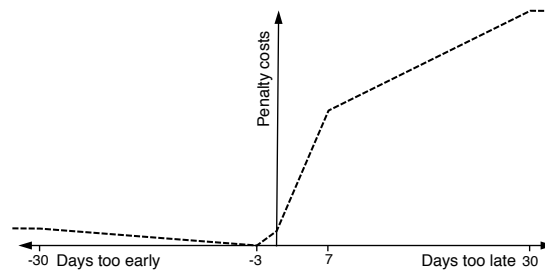


Figure 38: Penalty costs function

Approach

The simulation system SIRO is used in combination with the heuristic SIRO optimizer from the company PROFACTOR.

7.2.3 Simulation based scheduling optimization

This case from Krug & Schwöpe [2011] gives an insight into a production scheduling system, which is interconnected with SAP, of a car axles manufacturer.

Initial situation

The axle manufacturer is scheduling the production orders with SAP on a long-term horizon. As the PP-module of SAP is not offering operative planning, short-term changes cannot be taken into account. With a scheduling system this planning gap should get filled.

Objectives

The main targets are: minimize costs, reduce inventory, high utilization, reduce the lead-time and increase the delivery reliability.

These targets lead to the objective of an optimum sequence of the production orders and an optimum shift schedule.

Approach

The simulation system SPEEDSIM is used in combination with the intelligent ISSOP optimizer from the company DUALIS. In the graphical user interface the software tool QuickGANTT visualizes the order sequence. Furthermore, the whole production scheduling system is integrated well into SAP.

8 Summary

In this diploma thesis a simulation based production scheduling tool, which uses a meta heuristic improvement method, was developed. Typically, the production planning is in the field of tension between maximizing the utilization and the delivery reliability, but minimizing the lead-time and the inventory. The two main control parameters for the production planning are on the one hand the lot size and on the other hand the sequence.

Through the use of the developed software tool the in process inventory of the multilayer varistor production of TDK in Deutschlandsberg could be reduced by around 10% on planning level, only through optimizing the sequence of the production jobs. An inventory reduction will lead to a higher ROI because of the higher profit and also the lower capital employed. Furthermore, inventory is covering problems within the production and therefore an inventory reduction leads to more stable processes and thereby to a lead-time reduction.

The production scheduling tool standardizes the process of production scheduling and it shows the great possibilities of an online production planning and control system. Such an online production planning and control system is a closed loop control system of the production, which uses data from the Production Data Acquisition (PDA) system as feedback. The control variables can be on the one hand the staffing and on the other hand the production scheduling or even both. Thereby, the software system can control the production progress and in case of deviations to the simulated optimized plan, an online production planning and control system can adjust the production plan through a new optimization.

Bibliography

- Akers, S.B., 1956. Graphical Approach to Production Scheduling Problems. *Operations Research*, pp.244-45.
- AMPL, 2012. *AMPL Modeling Language for Mathematical Programming*. [Online] Available at: <http://www.ampl.com/> [Accessed 20 February 2012].
- Arnold, D. et al., 2008. *Handbuch Logistik*. Berlin: Springer.
- Bauer, J., 2011. *Handbuch Maschinenbau: Grundlagen und Anwendungen der Maschinenbau-Technik*. Wiesbaden: Springer.
- Brucker, P., 1995. *Scheduling Algorithms*. Heidelberg: Springer.
- Černý, V., 1985. Thermodynamical Approach to the Travelling Salesman Problem: An Efficient Simulation Algorithm. *Journal of Optimization Theory and Applications*, pp.41-51.
- Dangelmaier, W. & Aldinger, L., 1986. Kurzfristige Fertigungssteuerung mit Leitständen. *Werkstattstechnik*, pp.101-04.
- Deloitte & Touche, 1998. *Vision in Manufacturing*. Düsseldorf.
- Domschke, W., Scholl, A. & Voß, S., 1993. *Produktionsplanung: Ablauforganisatorische Aspekte*. Heidelberg: Springer.
- Dueck, G. & Scheuer, T., 1990. Threshold accepting: a general purpose optimization algorithm. *Journal of Computational Physics*, pp.161-75.
- EPCOS, 2010. *CTVS - Ceramic Transient Voltage Suppressors*. EPCOS.
- Erlach, K., 2010. *Wertstromdesign*. Stuttgart: Springer.
- Fourer, F., Gay, D.M. & Kernighan, B.W., 1990. A Modeling Language for Mathematical Programming. *Management Science*, pp.519-54.
- Gruber, M. et al., 2011. Vorausschauende Produktionsregelung durch simulationsbasierte heuristische Optimierung. In März, L. *Simulation und Optimierung in Produktion und Logistik*. Berlin: Springer. pp.65-77.
- Gutenberg, E., 1951. *Grundlagen der Betriebswirtschaftslehre. Band 1: Die Produktion*. Berlin: Springer.
- Heib, C. & Nickel, S., 2011. Performancevergleich zwischen simulationsbasierter Online- und Offline Optimierung anhand von Scheduling-Problemen. In März, L. *Simulation und Optimierung in Produktion und Logistik*. Berlin: Springer. pp.205-14.

- Heipcke, S., 2002. *Applications of optimization with Xpress-MP*. Paris: Editions Eyrolles und Dash Optimization Ltd.
- Hopp, W. & Spearman, M., 2001. *Factory Physics: Foundations of Manufacturing Management*. New York: McGraw-Hill.
- Johnson, S.M., 1954. Optimal Two- and Three-Stage Production Schedules with Setup Times Included. *Naval Research Logistics Quarterly*, pp.61-68.
- Kätschel, J., Teich, T., Köbernik, G. & Meier, B., 1999. Algorithms for the Job Shop Scheduling Problem - a comparison of different methods. In *European Symposium on Intelligent Techniques*. Creta, 1999. European Symposium on Intelligent Techniques.
- Kirkpatrick, S., Gelatt, C.D. & Vecchi, M.P., 1982. *Optimization by Simulated Annealing*. IBM Research Report RC 9355.
- Klemmt, A., Horn, S. & Weigert, G., 2011. Simulationsgestützte Optimierung von Fertigungsprozessen in der Halbleiterindustrie. In März, L. *Simulation und Optimierung in Produktion und Logistik*. Berlin: Springer. pp.49-63.
- Kondili, E., Pantelides, C.C. & Sargent, R.W., 1993. A General Algorithm for Short-Term Scheduling of Batch Operations - 1. MILP Formulation. *Computers & Chemical Engineering*, pp.211-27.
- Krug, W. & Schwöpe, M., 2011. Simulationsbasierte Reihenfolgeoptimierung in der Produktionsplanung und -steuerung. In März, L. *Simulation und Optimierung in Produktion und Logistik*. Berlin: Springer. pp.105-16.
- Lödding, H., 2005. *Verfahren der Fertigungssteuerung*. Berlin: Springer.
- Little, J.D.C., 1961. A Proof for the Queuing Formula: $L = \lambda W$. *Operations Research*, 9(3), pp.383-87.
- Lothar, M., Krug, W., Rose, O. & Weigert, G., 2011. *Simulation und Optimierung in Produktion und Logistik*. Heidelberg: Springer.
- Metropolis, N. et al., 1953. Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, pp.1087-159.
- März, L., Krug, W., Rose, O. & Weigert, G., 2011. *Simulation und Optimierung in Produktion und Logistik*. Berlin: Springer.
- Monden, Y., 1981. Adaptable Kanban System Helps Toyota Maintain Just-In-Time Production. *Industrial Engineering*, pp.29-46.

- Nance, R.E., 1993. *A History of Discrete Event-Simulation Programming Languages*. Blacksburg: Virginia Polytechnic Institute and State University.
- Nyhuis, P., 1999. *Logistische Kennlinien. Grundlagen, Werkzeuge und Anwendungen*. Berlin: Springer.
- Pinson, E., 1995. The Job Shop Scheduling Problem: A Concise Survey and Recent Developments. In Chretienne, P., Coffmann, E.G., Lenstra, J.K. & Liu, Z. *Scheduling Theory and its Applications*. Chichester: Jon Wiley & Sons. pp.277-93.
- Robinson, S., 2004. *Simulation: The Practice of Model Development and Use*. Chinchester: John Wiley & Sons Ltd.
- Rossi, H., 2003. *Ein heuristisches Dekompositionsverfahren für mehrstufige Losgrößenprobleme*. Berlin: FU Berlin.
- Schoner, P., 2007. *Operative Produktionsplanung in der Verfahrenstechnischen Industrie*. Kassel: kassel university press.
- Siemens Plant Simulation, 2012. *Plant Simulation: Siemens PLM Software*. [Online] Available at <http://www.siemens.com/plm/PlantSimulation> [Accessed 30 January 2012].
- Stadtler, H., 1996. Mixed integer programming model formulations for dynamic multi-item multi-level capacitated lotsizing. *European Journal of Operational Research* , 94, pp.561-81.
- Stalk, G. & Hount, T.M., 1990. *Zeitwettbewerb. Schnelligkeit entscheidet auf den Märkten der Zukunft*. Frankfurt: Campus-Verlag.
- Völker, S. & Schmidt, P.M., 2010. *Simulationsbasierte Optimierung von Produktions- und Logistiksystemen mit Tecnomatix Plant Simulation*. Karlsruhe: KIT Scientific Publishing.
- Vaessens, R.J., Aarts, E.H. & Lenstra, J.K., 1996. Job shop scheduling by local search. *INFORMS Journal on Computing*, pp.302-17.
- Vahrenkamp, R., 2003. *Quantitative Logistik für das Supply Chain Management*. München: Oldenbourg Wissenschaftsverlag.
- Wagner, H.M. & Whitin, T., 1958. Dynamic version of the economic lot size model. *Management Science*, pp.89-96.
- Wiendahl, H.P., 1997. *Fertigungsregelung. Logistische Beherrschung von Fertigungsabläufen auf Basis des Trichtermodells*. München: Carl Hanser Verlag.

List of Figures

Figure 1: Production logistic within the supply chain [Arnold et al. 2008]	3
Figure 2: Different views on inventory [modified from Wiendahl 1997]	8
Figure 3: Operation planning dilemma [Erlach 2010]	9
Figure 4: Target dilemma of the production [Erlach 2010]	10
Figure 5: Production planning with seasonal demand [Lödding 2005]	12
Figure 6: Solution steps of a meta heuristic method	23
Figure 7: Plant Simulation with an optimization problem	28
Figure 8: Protection level of the CTVS	29
Figure 9: Internal construction of a multilayer chip component	29
Figure 10: Product improvement during the production	31
Figure 11: Assembler (L90)	32
Figure 12: Histogram of the amount of foils per stack of all different MLV products	33
Figure 13: Press	33
Figure 14: Cutting (L90)	34
Figure 15: Side printing	35
Figure 16: Setter with chips for the thermal processes	35
Figure 17: Binder burn out	36
Figure 18: Sintering (Rotary hearth furnace)	36
Figure 19: Overview of the production process	38
Figure 20: Overview of the simplified production process	39
Figure 21: Architecture of the production scheduling tool	42
Figure 22: Overview of the simulation module	43
Figure 23: Example of the core simulation modules	44
Figure 24: <i>ClsQueue</i> functions: <i>EnQueue</i> and <i>DeQueue</i>	46
Figure 25: Example for the discrete-event simulation	47
Figure 26: Simulation model of the simplified production process (see Figure 20)	51
Figure 27: Overview of the optimization module	53

Figure 28: Example of the inner-, inter- and outer permutation	56
Figure 29: Production process	57
Figure 30: Gantt chart of order ABC	57
Figure 31: All six possible sequences.....	58
Figure 32: Cutout of the Gantt chart	62
Figure 33: Visualization of the side printing process.....	62
Figure 34: Visualization of the pusher type furnace.....	63
Figure 35: Visualization of the output of the production.....	64
Figure 36: Possibility of simulating the side print process and the production output...	66
Figure 37: Architecture of an online production scheduling system.....	67
Figure 38: Penalty costs function.....	70

List of Tables

Table 1: Aims of production logistic [Bauer 2011].....	3
Table 2: Target values of the logistic output performance and costs [Wiendahl 1997]...	4
Table 3: Production program planning [Lödding 2005]	12
Table 4: Main production processes of the MLV production.....	30
Table 5: Overview of case sizes of the different MLV products	34
Table 6: Furnace temperature assignment.....	37
Table 7: Process details of main frontend processes	40
Table 8: Description of all simulation modules.....	52
Table 9: Influence of the variable <i>SimulationStep</i> on the computing time and the objective values	52
Table 10: Production plan	57
Table 11: Comparison of average lead-times	60
Table 12: Inputs into the production scheduling tool.....	61
Table 13: Chronological starting sequence.....	64
Table 14: Examples of the testing results of the production scheduling tool	65

List of Equations

Equation 1: Delivery date deviation	5
Equation 2: Delivery reliability	5
Equation 3: Order lead-time	5
Equation 4: Date deviation	6
Equation 5: Date reliability	6
Equation 6: Little's law	6
Equation 7: Economic order quantity	14
Equation 8: Linear programming formulation	18
Equation 9: Disjunctive formulation	19
Equation 10: Boltzmann probability distribution	25
Equation 11: Starting temperature Simulated Annealing	25
Equation 12: Decline factor Simulated Annealing	25
Equation 13: Total work in process	58
Equation 14: General objective value	59
Equation 15: Example objective value	59
Equation 16: Implemented objective value	59

List of Algorithms

Algorithm 1: Discrete-event simulation	21
Algorithm 2: Simulated Annealing	24
Algorithm 3: Threshold Accepting	26
Algorithm 4: VBA-Class: ClsQueue	45
Algorithm 5: VBA-Class: ClsMachine	46
Algorithm 6: VBA-Function: Simulation	48
Algorithm 7: VBA-Function: RunMachine	49
Algorithm 8: VBA-Sub: SimulatedAnnealing	53

List of Abbreviations

B&B	<i>Branch and Bound</i>
BOM	<i>Bill of material</i>
CIP	<i>Continuous improvement process</i>
CPU	<i>Central processing unit</i>
CTVS	<i>Ceramic transient voltage suppressors</i>
DES	<i>Discrete-event simulation</i>
EOQ	<i>Economic order quantity</i>
ERP	<i>Enterprise resource planning</i>
ES	<i>Evolution Strategy</i>
FIFO	<i>First in first out</i>
GA	<i>Generic Algorithm</i>
GUI	<i>Graphical user interface</i>
MILP	<i>Mixed integer linear programming</i>
MLCLSP	<i>Multi-level capacitated lot sizing problem</i>
MLV	<i>Multilayer varistor</i>
PDA	<i>Production data acquisition</i>
ROI	<i>Return of investment</i>
SA	<i>Simulated Annealing</i>
SLULSP	<i>Single-level uncapacitated lot sizing problem</i>
STN	<i>State-Task-Network</i>
TA	<i>Threshold Accepting</i>
TS	<i>Tabu Search</i>
VBA	<i>Visual Basic for Application</i>
WIP	<i>Work in process</i>
WIQ	<i>Work in queue</i>

Attachment

Attachment 1: MLCLSP	82
Attachment 2: Flow-Shop problem as an assignment problem	83
Attachment 3: Production scheduling tool – Input.....	84
Attachment 4: Gantt chart of the actual sequence CW07 2012.....	85
Attachment 5: Gantt chart of the optimized sequence CW07 2012	86
Attachment 6: Side printing process of CW07 2012	87
Attachment 7: D7 furnace of CW07 2012	88
Attachment 8: Production output of CW07 2012.....	89

Attachment 1: MLCLSP

$$\text{Min.} \quad \sum_{j=1}^J \sum_{t=1}^T h_j \cdot I_{jt} + \sum_{j=1}^J \sum_{t=1}^T sc_j \cdot Y_{jt} + \sum_{m=1}^M \sum_{t=1}^T oc_{mt} \cdot O_{mt} \quad (1.1)$$

subject to

$$I_{j,t-1} + X_{jt} = P_{jt} + \sum_{k \in S_j} r_{jk}^d \cdot X_{kt} + I_{jt} \quad \forall j=1, \dots, J, t=1, \dots, T \quad (1.2)$$

$$\sum_{j=1}^J a_{mj} \cdot X_{jt} + \sum_{j=1}^J st_{jm} \cdot Y_{jt} \leq C_{mt} + O_{mt} \quad \forall m=1, \dots, M, t=1, \dots, T \quad (1.3)$$

$$X_{jt} \leq B_{jt} \cdot Y_{jt} \quad \forall j=1, \dots, J, t=1, \dots, T \quad (1.4)$$

$$\left. \begin{array}{l} I_{jt} \geq 0 \\ O_{mt} \geq 0 \\ X_{jt} \geq 0 \\ Y_{jt} \in \{0,1\} \end{array} \right\} \quad \begin{array}{l} \forall j=1, \dots, J, t=1, \dots, T \\ \forall m=1, \dots, M, t=1, \dots, T \\ \forall j=1, \dots, J, t=1, \dots, T \\ \forall j=1, \dots, J, t=1, \dots, T \end{array} \quad (1.5)$$

Indices and index sets:

j Items or operations (e.g. end products, intermediate products, raw materials), $j=1, \dots, J$

m Resources (e.g. personnel, machines, production lines), $m=1, \dots, M$

t Periods, $t=1, \dots, T$

S_j Set of immediate successors of item j in the bill of material

Data:

a_{mj} Capacity needed on a resource m for one unit of item j

B_{jt} Large number, not limiting feasible lot-sizes of item j in period t

C_{mt} Available capacity of resource m in period t

h_j Holding cost for one unit of item j in a period

oc_{mt} Overtime cost for one unit of resource m in period t

P_{jt} Primary, gross demand for item j in period t

r_{jk}^d Number of units of item j required to produce one unit of the immediate successor item k

sc_j Setup cost for a lot of item j

st_{jm} Setup time for item j on resource m

Variables:

I_{jt} Inventory of item j at the end of period t

O_{mt} Amount of overtime on resource m used in period t

X_{jt} Production quantity of item j in period t (lot-size)

Y_{jt} Binary setup variable (=1, if item j is produced in period t , 0 otherwise)

The objective function (1.1) aims at minimizing the sum of inventory holding, setup and overtime costs. All other production costs are assumed to be fixed and independent of time, consequently no direct production costs are attributed to a lot-size X_{jt} .

Multi-level inventory balance constraints (1.2) make sure that no backlogging will occur. For multi-level production a lot-size of item k will result in a dependent demand for its immediate predecessor items j . Required capacities for lot-size production must not exceed available normal capacities (possibly extended by overtime; 1.3). Capacity requirements result from both production time per item times the amounts produced as well as setup times incurred with each lot. Setup constraints (1.4) enforce binary variables Y_{jt} to unity, in case a lot of item j is produced in a period t . All variables are restricted to non-negative or binary values, respectively (1.5).

Attachment 2: Flow-Shop problem as an assignment problem

Für das Flow Shop-Problem mit den Auftragsmenge \mathcal{J} ($|\mathcal{J}| = n_{\mathcal{J}}$), der Menge der Maschinen \mathcal{M} ($|\mathcal{M}| = n_{\mathcal{M}}$), und der Rangliste \mathcal{R} mit $|\mathcal{J}| = |\mathcal{R}|$, definiere die Entscheidungsvariable $x_{j,r}$ mit

$$\begin{aligned} \forall j \in \mathcal{J}, r \in \mathcal{R} & : x_{j,r} \in \{0, 1\} \\ \forall r \in \mathcal{R} & : \sum_{j \in \mathcal{J}} x_{j,r} = 1 \\ \forall j \in \mathcal{J} & : \sum_{r \in \mathcal{R}} x_{j,r} = 1 \end{aligned}$$

Hierbei bestimmt $x_{j,r}$, welcher Rang einem Auftrag j zugeordnet ist.

Zur Berechnung der Gesamtdauer definiere die reellwertige Variable $e_{m,r}$, die die ungenutzte (*empty*) Zeit der Maschine m zwischen den zwei Aufträgen vom Rang r und $r+1$ angibt, und die reellwertige Variable $w_{m,r}$, die die Wartezeit des Auftrags r zwischen den Operationen auf den Maschinen m und $m+1$ angibt.

$$\begin{aligned} \forall m \in \mathcal{M}, r = 1, \dots, (n_{\mathcal{J}} - 1) & : e_{m,r} \geq 0 \\ \forall r = 1, \dots, (n_{\mathcal{J}} - 1) & : e_{1,r} = 0 \\ \forall m = 1, \dots, (n_{\mathcal{M}} - 1), r \in \mathcal{R} & : w_{m,r} \geq 0 \\ \forall m = 1, \dots, (n_{\mathcal{M}} - 1) & : w_{m,1} = 0 \end{aligned}$$

Bei der Produktionsdauer $t_{m,j}$ eines Auftrags j auf einer Maschine m werden jetzt die Warte-, Leer- und Produktionszeiten miteinander verknüpft durch

$$\forall m = 1, \dots, (n_{\mathcal{M}} - 1), r = 1, \dots, (n_{\mathcal{J}} - 1) : \\ e_{m,r} + \sum_{j \in \mathcal{J}} t_{m,j} \cdot x_{j,r+1} + w_{m,r+1} = w_{m,r} + \sum_{j \in \mathcal{J}} t_{m+1,j} \cdot x_{j,r} + e_{m+1,r}$$

Für eine Minimierung des Makespan lautet das Optimierungsproblem dann

$$\min \sum_{m=1}^{n_{\mathcal{M}}-1} \sum_{j \in \mathcal{J}} t_{m,j} \cdot x_{j,1} + \sum_{r=1}^{n_{\mathcal{J}}-1} e_{n_{\mathcal{M}},r}$$

Formulation from [Heipcke 2002]

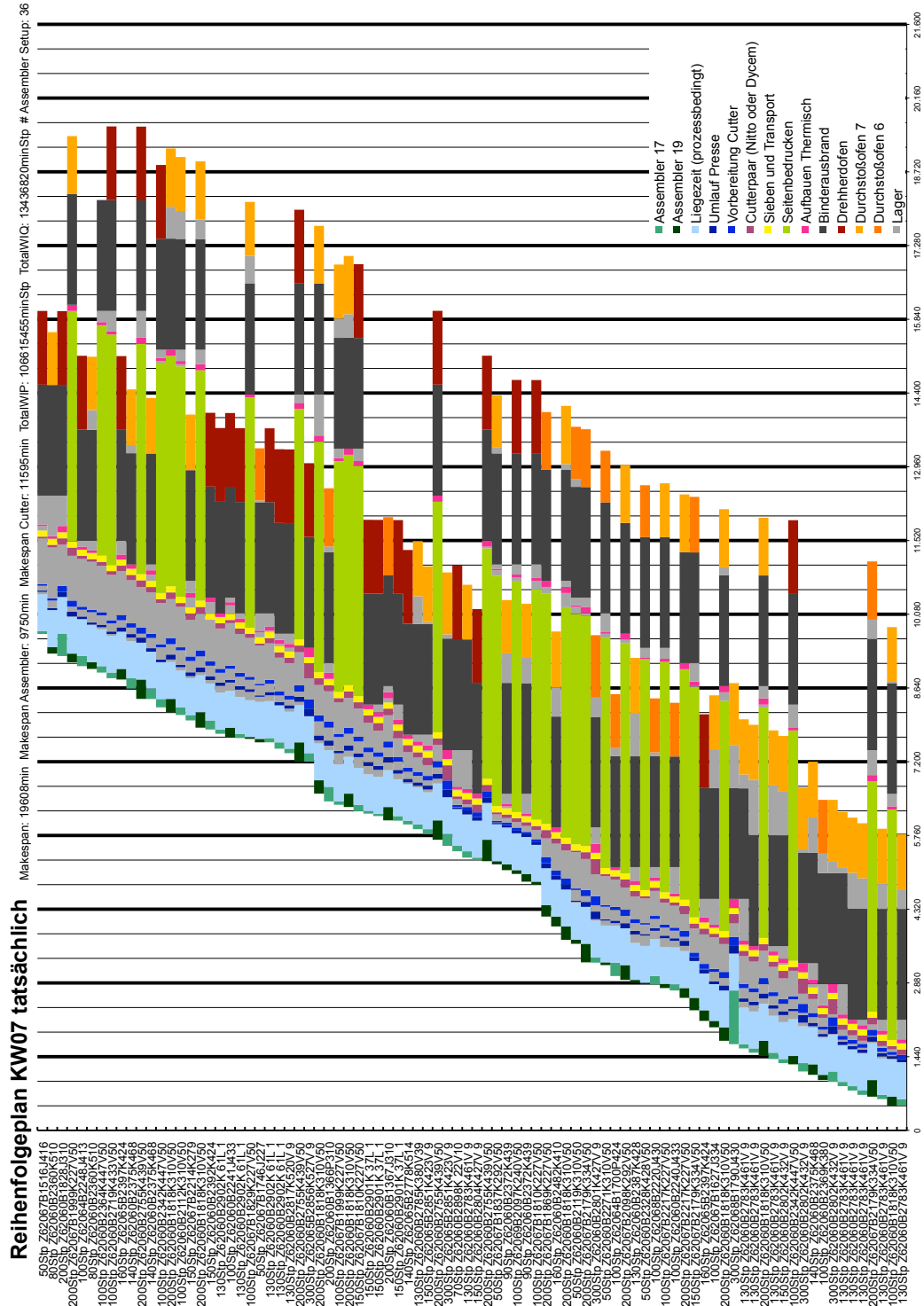
Attachment 3: Production scheduling tool – Input

Daten - Feinplanung

Job Nummer	Material Nummer	Loggröße	Bauform	Dickeklasse	Typ	Stk pro Stapel	Schablone	Paste	Folien Assembler	Klebefolie	Temperatur	Assembler	Press	Vorbereitung	Cutter	Vorbereitung	Cutter	Thermal	Rüstcluster	
1																				
2																				
3																				
4	109903282	Z62065B2856K527V	9	300	402	0.5	V9	15.510	z509-h041	M3031	9	Nitto	1020	121	148	177	295	177	1	
5	109903276	Z62065B2778K514	140	402	0.5	HDMI	14877	z509-g020	M3031	5	Nitto	100	32	69	83	138	83	2	2	
6	109903267	Z62060B2817K520V	9	130	603	0.8	HDMI	5.724	z508-g020	M3031	8	Nitto	100	47	64	77	105	77	2	
7	109903247	Z62060B2783K461V	9	130	603	0.8	V9	5.616	b608-n173	M3031	14	Nitto	1030	82	64	77	105	77	3	
8	109903248	Z62060B2783K461V	9	130	603	0.8	V9	5.616	b608-n173	M3031	14	Nitto	1030	82	64	77	105	77	3	
9	109903249	Z62060B2783K461V	9	130	603	0.8	V9	5.616	b608-n173	M3031	14	Nitto	1030	82	64	77	105	77	3	
10	109903250	Z62060B2783K461V	9	130	603	0.8	V9	5.616	b608-n173	M3031	14	Nitto	1030	82	64	77	105	77	3	
11	109903251	Z62060B2783K461V	9	130	603	0.8	V9	5.616	b608-n173	M3031	14	Nitto	1030	82	64	77	105	77	4	
12	109903253	Z62060B2783K461V	9	130	603	0.8	V9	5.616	b608-n173	M3031	14	Nitto	1030	82	64	77	105	77	4	
13	109903254	Z62060B2783K461V	9	130	603	0.8	V9	5.616	b608-n173	M3031	14	Nitto	1030	82	64	77	105	77	4	
14	109903255	Z62060B2783K461V	9	130	603	0.8	V9	5.616	b608-n173	M3031	14	Nitto	1030	82	64	77	105	77	4	
15	109903256	Z62060B2783K461V	9	130	603	0.8	V9	5.616	b608-n173	M3031	14	Nitto	1030	82	64	77	105	77	4	
16	109903262	Z62060B2802K432V	9	300	402	0.5	V9	15.561	b509-z146	M3031	12	Nitto	1030	162	148	177	295	177	5	
17	109903263	Z62060B2802K432V	9	300	402	0.5	V9	15.561	b509-z146	M3031	12	Nitto	1030	162	148	177	295	177	5	
18	109903264	Z62060B2802K432V	9	150	402	0.5	V9	15.561	b509-z146	M3031	12	Nitto	1030	81	74	89	148	89	6	
19	109903278	Z62065B2851K423V	9	300	402	0.5	V9	15.561	b509-z146	M3031	14	Nitto	1030	189	148	177	295	177	6	
20	109903280	Z62065B2851K423V	9	150	402	0.5	V9	15.561	b509-z146	M3031	14	Nitto	1030	95	74	89	148	89	6	
21	109903617	Z62060B2901K 37L	1	150	402	0.5	V9	15.015	b509-t198	M3031	7	Nitto	100	48	74	89	148	89	7	
22	109903618	Z62060B2901K 37L	1	150	402	0.5	V9	15.015	b509-t198	M3031	7	Nitto	100	48	74	89	148	89	7	
23	109903619	Z62060B2901K 37L	1	150	402	0.5	V9	15.015	b509-t198	M3031	7	Nitto	100	48	74	89	148	89	7	
24	109903261	Z62060B2801K427V	9	300	402	0.5	V9	15.015	b509-t197	M3031	4	Nitto	1060	54	148	177	295	177	8	
25	109903266	Z62060B2804K427V	9	150	402	0.5	V9	15.015	b509-t197	M3031	7	Nitto	100	48	74	89	148	89	8	
26	109920553	Z62060B2902K 61L	1	130	603	0.8	V9	5.616	b508-n174	M3031	14	Nitto	100	82	64	77	105	77	9	
27	109920554	Z62060B2902K 61L	1	130	603	0.8	V9	5.616	b508-n174	M3031	14	Nitto	100	82	64	77	105	77	9	
28	109920555	Z62060B2902K 61L	1	130	603	0.8	V9	5.616	b508-n174	M3031	14	Nitto	100	82	64	77	105	77	9	
29	109920556	Z62060B2902K 61L	1	130	603	0.8	V9	5.616	b508-n174	M3031	14	Nitto	100	82	64	77	105	77	9	
30	109903260	Z62060B2785K380V	39	130	603	0.8	CC	5.616	b508-n150	M3031	9	Nitto	1040	63	64	77	105	77	10	
31	109903610	Z62060B2898K 22V	10	70	1206	1.1	CC	1.350	b112-n150	M3031	20	Nitto	100	53	35	42	53	42	11	
32	109903198	Z62060B2271K310V	50	603	603	0.8	CT	5.724	z508-g020	P237	5	Nitto	1050	14	25	30	41	30	12	
33	109903195	Z62060B2112K310V	50	100	1206	1.1	CT	1.350	b612-n150	P237	20	Nitto	1040	108	50	59	75	59	13	
34	109903201	Z62060B2342K447V	50	100	1206	1.3	CT	1.350	b612-n150	P237	30	Nitto	100	162	50	59	75	59	13	
35	109903214	Z62060B2375K468	140	1206	1.5	CC	1.350	b612-n150	P237	27	Nitto	1040	204	69	83	105	83	13	13	
36	109903213	Z62060B2375K468	140	1206	1.5	CC	1.350	b612-n150	P237	27	Nitto	1040	204	69	83	105	83	13	13	
37	109903220	Z62060B2719K433V	50	100	1206	1.1	CT	1.350	b612-n150	P237	20	Nitto	100	108	50	59	75	59	14	

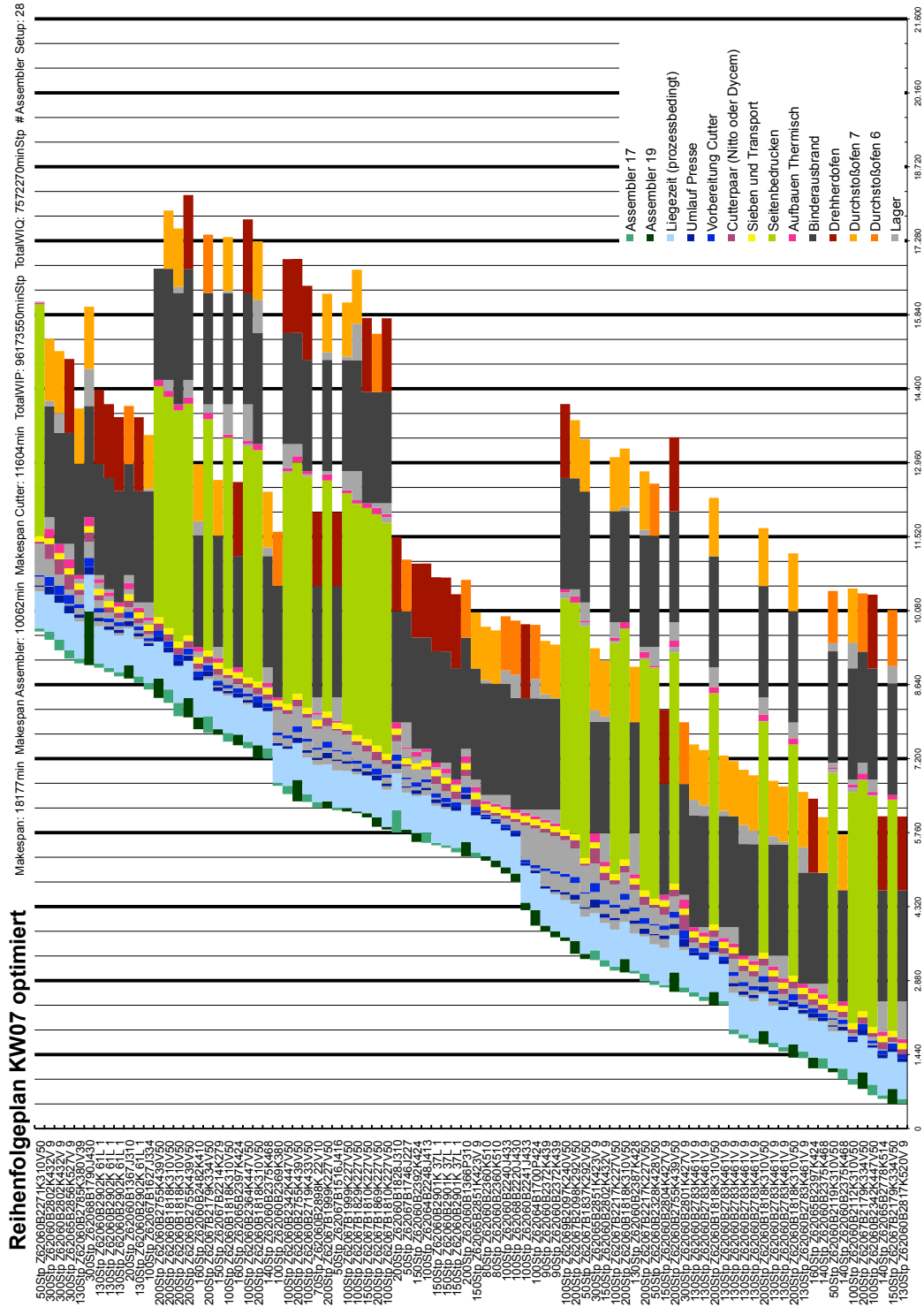
Screenshot of the main input of the production scheduling tool.

Attachment 4: Gantt chart of the actual sequence CW07 2012



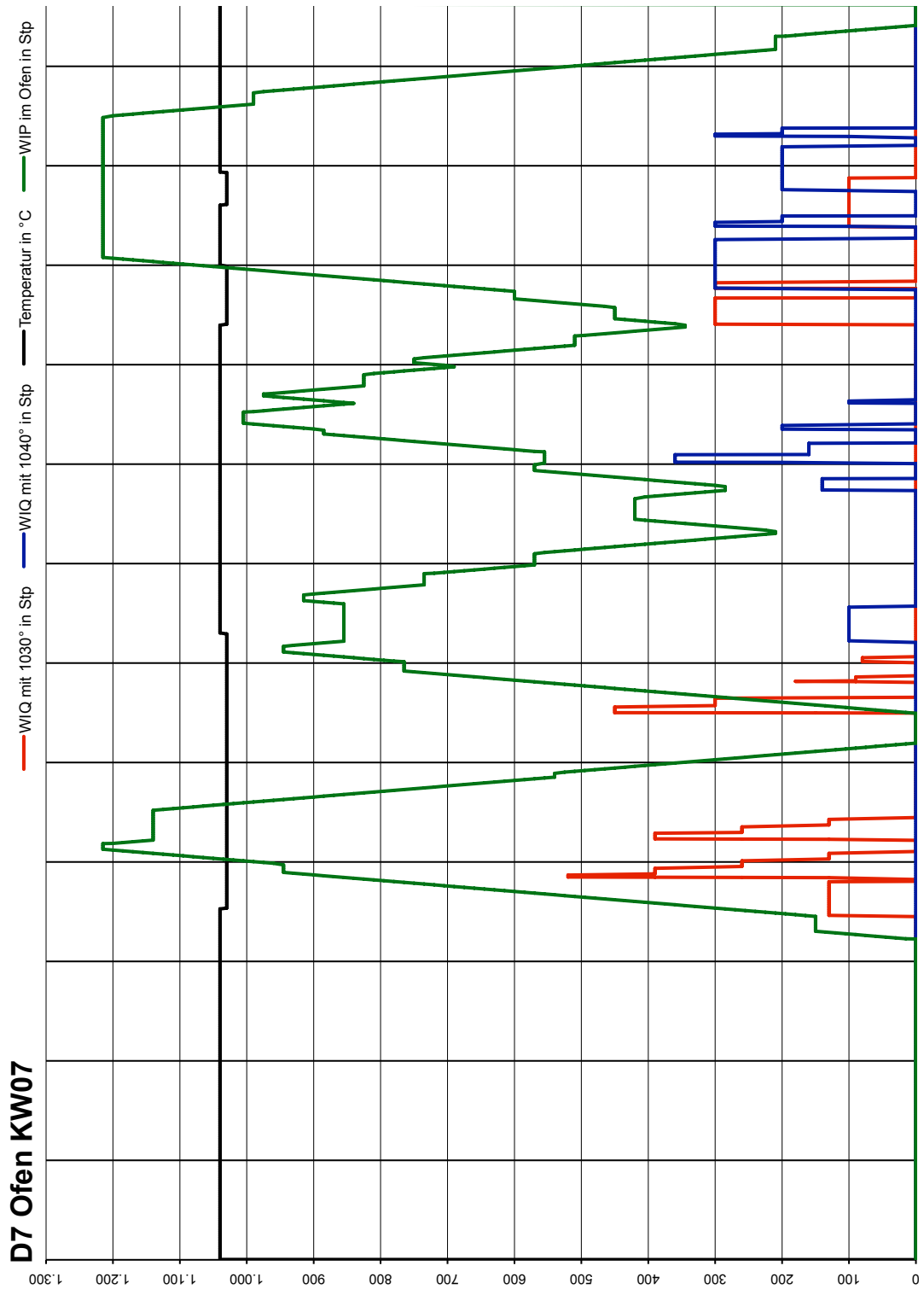
Screenshot of the Gantt chart output of the production scheduling tool of the actual sequence of the CW07 2012.

Attachment 5: Gantt chart of the optimized sequence CW07 2012



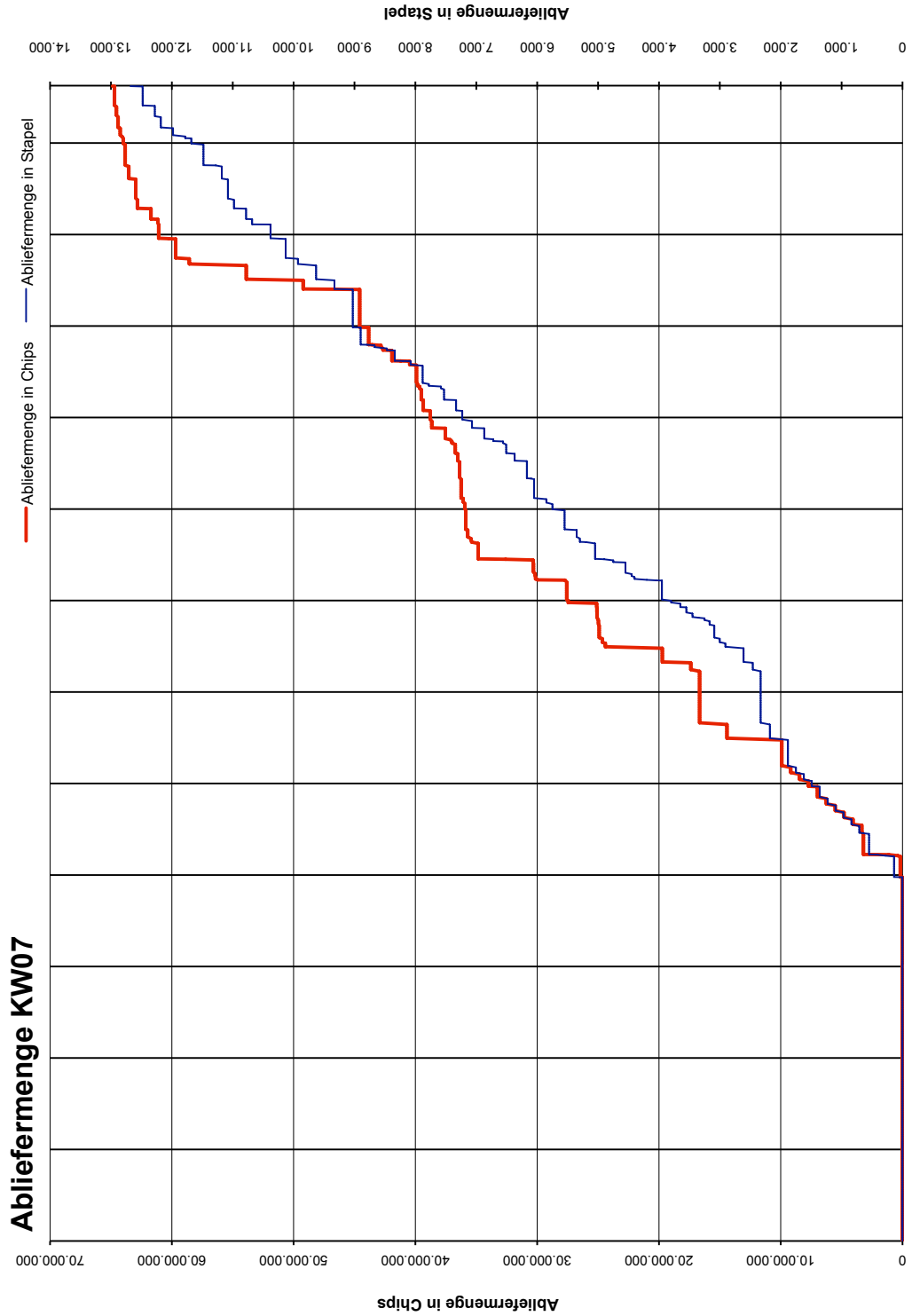
Screenshot of the Gantt chart output of the production scheduling tool of the optimized sequence of the CW07 2012.

Attachment 7: D7 furnace of CW07 2012



Screenshot of the visualization of the D7 pusher type furnace of the production scheduling tool of the optimized sequence of the CW07 2012.

Attachment 8: Production output of CW07 2012



Screenshot of the visualization of the production output of the production scheduling tool of the optimized sequence of the CW07 2012.