

Manfred Toferer

Determining Flexible Maintenance Opportunities with an Algorithm for Flow Shop Problems

Master's Thesis

Graz University of Technology

Institute of Engineering and Business Informatics
Head: Univ.-Prof. Dipl.-Ing. Dr.techn.Siegfried Vössner

Supervisor: Dipl.-Ing. Dietmar Neubacher

Graz, November 2016

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.

Graz, _____

Date

Signature

Eidesstattliche Erklärung¹

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am _____

Datum

Unterschrift

¹Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008; Genehmigung des Senates am 1.12.2008

Acknowledgement

First of all I want to thank Univ.-Prof. Dipl.-Ing. Dr.techn. Siegfried Vössner, who gave me the possibility to write this thesis at the Institute of Engineering and Business Informatics. I want to express my biggest gratitude to my two supervisors Dipl.-Ing. Dietmar Neubacher and Dipl.-Ing. Dr.techn. Nikolaus Furian for their support, input and remarks. The purpose of the thesis was to develop an algorithm for the determination of flexible maintenance opportunities for flow job problems. The master's thesis gave me the possibility to strengthen my knowledge in the field of production planning. I am grateful to have the opportunity to give a contribution to such an exciting project.

A special thank goes to my friend Dominik, with whom I shared an office for the time of the thesis. His input and the interesting discussions helped me during my whole studies.

Furthermore, I want to thank Elisabeth for the thorough proofreading.

Another thank goes to my sister Christiane and my brothers Ernst, Reinhard and Martin who encouraged me already at a young age to study. Especially, I want to thank my twin brother Wolfgang, with whom I grew up and lived together during my studies.

Finally, I would like to thank my girlfriend Martina and my parents Anneliese and Ernst, for providing me unfailing support and help during my whole studies.

Abstract

In the last years the development of automation and digitalization of manufacturing technologies caused the fourth industrial revolution. These developments have had a major impact on the evolution of maintenance. Production facilities are getting more complex and new technologies arise in the area of maintenance. A constantly increasing number of researchers already observe the potential of these technologies in order to make maintenance more efficient. A high potential is already spotted in the area of machine maintenance, as this influences the production. In order to achieve a high availability and secure reliability, the optimized scheduling of tasks becomes critical. Usually the equipment has to be stopped to perform maintenance tasks and therefore a decrease of the throughput can be registered. One planning method which deals with this problem is the use of opportunity time windows. These windows occur due to line dynamics and are defined by opportunities of maintenance which have no influence on the throughput of the production line. The objective of this thesis is to develop an algorithm that makes use of such opportunity windows. Thereby, the algorithm is split into two major parts. Firstly, the opportunity windows are determined for every machine over a defined period. Secondly, the algorithm schedules predefined tasks within the calculated window. While the first part uses a simulation based approach, the second is defined as a linear integer problem. In order to validate the performance of the opportunity window calculator various experiments are conducted on simulated production lines. In the course of these experiments the machines are shut down according to the calculated windows of the algorithm. The algorithm creates the opportunity time windows almost without influence on the throughput, since the highest throughput decrease is 0.85%. Finally, a simplified example is used to highlight the potential of utilizing such flexible maintenance opportunities. Thereby fixed maintenance shifts could be significantly reduced, as several required tasks will be performed in these time windows. Common maintenance strategies already support autonomous and flexible task completion. Therefore, determining and utilizing these opportunities significantly increases machine availability and productivity, as unnecessary losses, like machine blocking and starving, are reduced.

Kurzfassung

Der Bereich der Instandhaltung hat in den letzten Jahren immer mehr an Bedeutung und Aufmerksamkeit gewonnen. Durch neue Entwicklungen im Bereich von Industrie 4.0 wurde ein größeres Potenzial für Optimierungen in diesem Sektor generiert. Ein spezielles Thema in diesem Bereich ist die optimale Planung von Instandhaltungsaufgaben. Der Prozess der Aufgabenverteilung stellt aufgrund der notwendigen Einflussnahme auf aktive Produktionszeiten und dem damit verbundenen Durchsatz ein kritisches Verfahren dar. In den vergangenen Jahren wurde in der Fachliteratur immer wieder aufgezeigt, dass es Möglichkeiten gibt, die laufende Produktion mit den notwendigen Instandhaltungstätigkeiten zu kombinieren. Pufferstände und Ausfälle von Maschinen können demnach genutzt werden, um präventive Instandhaltungstätigkeiten ohne Einfluss auf den Durchsatz durchzuführen. Die vorliegende Arbeit befasst sich mit der Entwicklung eines Algorithmus für die flexible Instandhaltungsplanung auf Basis der genannten Möglichkeiten. Im ersten Schritt erstellt der Algorithmus für jede einzelne Maschine Zeitfenster über einen definierten Zeitraum. In weiterer Folge werden vordefinierte, präventive Instandhaltungstätigkeiten auf die zuvor erstellten Fenster geplant. Der erste Teil des Algorithmus verwendet einen simulationsbasierten Ansatz, der zweite Teil wird über die Optimierung eines linearen Integer Models erreicht. Um die erstellten Zeitfenster validieren zu können werden mithilfe des Algorithmus alle möglichen Zeitfenster für drei unterschiedliche Produktionslinien berechnet. Im Anschluss wird in einer Simulation jede Maschine in den berechneten Zeitfenstern aus der Produktion genommen. Die Resultate der Validierung ergeben für den Zeitfensterplaner ein nahezu perfektes Ergebnis. Im schlechtesten Fall wird der Durchsatz um lediglich 0,85% verringert. Am Ende wird über ein vereinfachtes Beispiel das Potential des Algorithmus dargestellt. Fixe Instandhaltungsschichten können durch die Durchführung von Aufgaben in flexiblen Zeitfenstern stark verringert werden. Durch eine Nutzung dieser flexiblen Fenster, können zudem Durchsatz und Maschinenverfügbarkeit erhöht, sowie unnötige Wartezeiten verringert werden.

Contents

1	Introduction	1
1.1	Background and Motivation	1
1.2	Problem definition	3
1.3	Goals	3
1.4	Structure	4
2	Theoretical Framework	5
2.1	Maintenance	5
2.1.1	Introduction	6
2.1.2	Terms of maintenance	8
2.1.3	Maintenance strategies	11
2.1.4	TPM	13
2.2	Optimization Problems and Algorithms	17
2.2.1	Introduction to scheduling	17
2.2.2	Linear programming	19
2.2.3	Integer programming	21
2.3	Optimization in production and maintenance planning	25
2.4	Simulation	27
2.4.1	Introduction to Simulation	27
2.4.2	Simulation Approaches	32
2.5	Simulation in maintenance and manufacturing	37
2.6	Maintenance opportunity window calculation and simulation	38
3	Model Development	39
3.1	Problem Description	39
3.2	Concept	42
3.3	Assumptions and Simplifications	44
3.4	Mathematical model	45
3.4.1	Time window planner	45

Contents

3.4.2	Task scheduler	59
4	Implementation	67
4.1	Algorithm	67
4.1.1	Time Window Planner	67
4.1.2	Task scheduler	75
4.2	Validation	77
4.2.1	Validation design	77
4.2.2	Results	80
5	Conclusion	89
5.1	Exemplification Case	90
5.2	Future Work	91
	Appendix	92
	Bibliography	102

List of Figures

2.1	Tasks of maintenance adapted from DIN31051:2012-09 (2012)	6
2.2	Maintenance objectives adapted from Strunz (2012, pp.3)	7
2.3	Classical Maintenance strategies adapted from Eichler (1990, p.149)	11
2.4	Maintenance costs modified from Strunz (2012, p.18)	12
2.5	Eight pillars for TPM implementation adapted from Ahuja and Khamba (2008)	14
2.6	Overall Equipment Effectiveness and Goals adapted from Nakajima (1988, p.25)	17
2.7	Cutting Plane and Branch and Bound Method	22
2.8	Maintenance optimization classification framework (Van Horenbeek et al., 2010)	25
2.9	Artefacts of Conceptual Modelling (Robinson, 2013, p.381)	28
2.10	Steps in a simulation study adapted from Banks et al. (2004, p.13)	31
2.11	Approaches in Simulation (Borshchev and Filippov, 2004, p.3)	32
2.12	Classification according to the representation of time bases/state variables (Wainer, 2009, p.16)	33
2.13	Discrete event simulation vs. time driven simulation adapted from Hedtstück (2013, p.22)	34
2.14	Activity scanning adapted from Balci (1988)	36
2.15	Optimal problem formulation for different types of maintenance optimization problems adapted from Alrabghi and Tiwari (2015)	37
3.1	Problem definition	40
3.2	Project	41
3.3	Concept explanation	42
3.4	Employee assumptions	44
3.5	Single machine-buffer-single machine system	46
3.6	Redundant-machine-system	47

List of Figures

3.7	Comparison full to empty buffer	48
3.8	Cases of machine-buffer-machine structure	49
3.9	State changes of a draining window depending on the structure of the formation	50
3.10	State changes Case4 for a filling window	52
3.11	Redundant production line	52
3.12	Calculation of d_i in Case2 with filling buffer	55
3.13	Calculation of d_i in Case4 with filling buffer	57
3.14	Bottleneck detection	58
3.15	Bottleneck influence	58
3.16	Bottleneck influence of cycle time	59
3.17	Timeline of scheduled task j in time window k	61
3.18	Calculation availability	63
3.19	Employee matrices	64
3.20	Overlapping matrix O_{kl}	65
3.21	Penalty-Function	66
4.1	Gantt Chart of the time windows	74
4.2	Output Employee-Task List	76
4.3	Output Task-Window List	77
4.4	Validation process and abstraction levels	78
4.5	Single machine production line in Plant Simulation [®]	80
4.6	Waiting and blocking times of bottleneck for two days	81
4.7	Comparison of average times of the production lines	82
4.8	Comparison accumulated and average window durations	83
4.9	Utilization statistic single machine production line case2	84
4.10	Comparison of production lines and cases regarding the production statistic	85
4.11	Buffer comparison	86
4.12	Buffer comparison program- Plant Simulation [®]	87
A1	Redundant production line Case1	94
A2	Redundant production line Case2	95
A3	Production rate diagram for redundant production line cases	96
B1	ERM model	97
C1	Main window program	98
D1	Single machine production line in Plant Simulation	99
D2	Parallel production line	100
D3	Mixed production line in Plant Simulation	101

List of Tables

4.1	Input Machine	69
4.2	Input Buffer	69
4.3	Activity: Produce with individual rate	71
4.4	Activity: Produce with standard rate	71
4.5	Activity: Flexible time window	72
4.6	Output Time Window	74
4.7	Input Employee	75
4.8	Input Task	76
4.9	Cycle times and buffer capacity previous to bottleneck	79
4.10	Cycle times and buffer capacity after bottleneck	79
4.11	Utilization statistic of bottleneck	80
4.12	Buffer level mean deviation over two days	88
5.1	Throughput calculation	91

List of Algorithms

1	Discrete event simulation	34
2	Time driven simulation	35
3	Time window planner	68
4	Program sequence	70
5	Set new production rate	71
6	Reset production rate to standard	72
7	Set flexible time window	72
8	Create time window	73
9	Task scheduler	75

Chapter 1

Introduction

1.1 Background and Motivation

During the last decades maintenance changed in different ways. One reason for this change is an increase of the degree of automation and complexity at the shop floor. Another one is that maintenance strategies changed from a corrective to a predictive character. (Nakajima, 1995, pp.18-19) The history of maintenance covers five generations. The first generation (until the 1950s) focused on repair actions of broken machines. Preventive tasks were not carried out at all. In the 1960s the second generation was ushered in. In this period of time maintenance tasks were planned in advance. Furthermore, the first systems for maintenance were developed and preventive maintenance was introduced. In the 1980s, due to new production systems and a more efficient value chain, breakdowns of machines had higher influence on the output of the process. To avoid unnecessary downtimes, condition based maintenance was introduced for critical parts. With sensors, the actual state of a unit is measured and if necessary, maintained. The current generation of maintenance goes beyond the preventive and condition based maintenance planning. With the integration of the production, on the basis of the reliability studies, data and modern information, a holistic maintenance becomes possible. (Weißenbach, 2012)

These changes in the environment caused the necessity for new ways to organize maintenance. One of these new approaches is the concept of Total Productive Maintenance (TPM), developed in Japan in the 1970s. The main objective of TPM is to increase the OEE by including workers of the production line into maintenance tasks. Failure and unplanned downtimes should be totally eliminated. (Nakajima, 1995, p.19) In the last

years TPM was established in different sectors and became an industrial standard for high efficient maintenance.

Furthermore, the upcoming topics of Industry 4.0¹ and Big Data lead to a change in the future of maintenance. The combination of advanced automated technologies and the Internet of Things (IoT) create the possibility of new production systems. Due to higher digitalization of production units, new ways of monitoring and the evaluation of machines will be possible. (Güntner et al., 2015, p.7) This increased degree of digitalization involves a higher amount of collected data. On the basis of big data analysis these data can be dealt with. Big data analysis also helps to predict machine downtimes and to create a maintenance plan to prevent machine failures.

One major challenge for maintenance departments will be the integration of these new technologies into the maintenance process. According to the survey of Güntner et al. (2015, pp.7-8), the opportunities for companies of this evolution are on the one hand a higher integration of the production process into maintenance and therefore a higher availability of the machines. On the other hand the image of maintenance is changing from a cost generating view to a value adding view. The main challenge is that this integration increases the complexity of the system and therefore the requirements on maintenance rise.

These requirements and challenges lead to the necessity of research in the field of maintenance. Van Horenbeek et al. (2010) established a framework on how optimization in maintenance can be divided. The objectives of current research are:

- Evaluation/comparison concepts, policies and actions
- Maintenance/replacement timing
- Inspection timing/frequency
- Maintenance scheduling/planning
- Capital equipment replacement
- Resource requirements

Maintenance scheduling can be performed on the machine or on the line level. The latter also includes using line dynamics for optimal scheduling which is also often called opportunity scheduling. Chang et al. (2007) defined the maintenance opportunity as *"a time window for a specific machine being purposely shut down to do PM without substantially impacting the flow of the line for the smooth operation of a manufacturing or*

¹Industry 4.0 integrates new information technologies into the production. It changes the way how to work in a production facility. The basis are intelligent, connected systems which enable a self organizing production. (Plattform Industrie 4.0, 2016)

assembly plant". In other words, line dynamics, like buffer levels or unplanned breakdowns, should be used to maintain a machine during the production process without impacting the line throughput.

Nowadays many companies classify maintenance tasks as fixed weekly or monthly shifts. In these shifts the production line is completely shut down and therefore the throughput during this time is zero. The previously explained maintenance opportunity windows can create a maintenance plan, which is adjusted to the company's available resources. Thereby the fixed shifts are changed to flexible maintenance times without influencing the throughput. In course of this thesis a algorithm is developed which allows to create such a maintenance plan that enables the change of the structure from fixed to a flexible maintenance on production lines. The study is done aligned with a TPM4.0 project from the Institute of Engineering and Business Informatics at the Technical University of Graz in cooperation with a German car manufacturer. The subject of the project is the maintenance of the future with a strong focus on data analysis to determine optimal maintenance strategies, evaluate organizational structures and efficiently schedule maintenance tasks.

1.2 Problem definition

As already mentioned in the previous section some companies as well as the project partner currently work with fixed shifts for preventive maintenance on their machines. From an economic point of view these fixed shifts could lead to a decrease in the throughput and could therefore result in inefficiency. The key problem is, if machines do not work or other types of unplanned errors in the standard production line occur, the machine in the following or previous step will have not planned, and more importantly, an unnecessary waiting or blocking time. Especially at longer production lines this leads to a high amount of unused machines. Moreover, standard maintenance tasks could lead to a decrease in throughput if they are carried out at the wrong point in time. These two major problems define the initial situation. A detailed problem definition can be found in chapter 3.

1.3 Goals

As part of the project TPM4.0 the objective of this thesis is to create a tool to enable flexible maintenance planning. Therefore, an algorithm has to be developed which uses

the dynamics of the production line through flexible scheduling of TPM activities on the machines. Finally, the findings should be used to improve the throughput by using available resources in an optimized way.

This main objective leads to two main tasks:

- The first task is to develop an algorithm which schedules tasks of the TPM shift on a flexible base. There can be two different initiators for the algorithm. On the one hand an initiator can be a unplanned breakdown of a machine in a production line. The algorithm should suggest TPM tasks for the upstream and downstream machines. On the other hand the algorithm can be triggered by the regular weekly or monthly maintenance scheduling process.
- The second task is to validate the developed algorithm in a virtual environment (done by a discrete event simulation model implemented in Plan Simulation^{®2}) and to adapt this algorithm in iterative steps.

1.4 Structure

This thesis consists of a theoretical and a practical part, so the remainder is structured as follows: A theoretical overview is given which provides the necessary background for the applied methods in the practical part. The theoretical part starts with an introduction of the field of maintenance and an explanation of common strategies like TPM. Afterwards, algorithms and simulation techniques for maintenance and production planning will be discussed. The practical part will start with the basic concept of the algorithm and the model of the problem. Furthermore, the developed algorithm is presented. Finally the validation of the developed algorithm is done.

²Plant Simulation is a software, developed by Siemens PLM Software, for simulation, optimization, and modeling of production systems.

Chapter 2

Theoretical Framework

The theoretical framework provides the necessary background information for the practical part. It is divided into the three main chapters:

- maintenance
- optimization problems and algorithms
- simulation

Furthermore recent developments and researches in these fields are shown.

2.1 Maintenance

When looking at the etymology of the word maintenance one can find the origin in different languages. While the meaning of the word in Middle English used to describe bearing or deportment, the meaning of the actual word was mainly influenced by the Old French word "maintenance" which basically stood for upkeep, shelter or protection. (Harper, 2016)

This subsection starts with a basic introduction to maintenance. Afterwards the main targets and common strategies will be discussed. At the end a detailed explanation of Total Productive Maintenance will be given.

2.1.1 Introduction

According to DIN31051:2012-09 (2012) maintenance can be defined as follows: Maintenance is the combination of all technical and administrative measures, as well as activities from the management to preserve the functional status of a unit or to return to this during the life cycle. Thereby a life cycle includes all periods, from the concept to the disposal phase. The function is defined by the specifications (attributes) a unit has after the manufacturing process has been taken place. The word unit in the definition is defined by any component, device, subsystem, functional unit, resource or system which can be taken by itself.

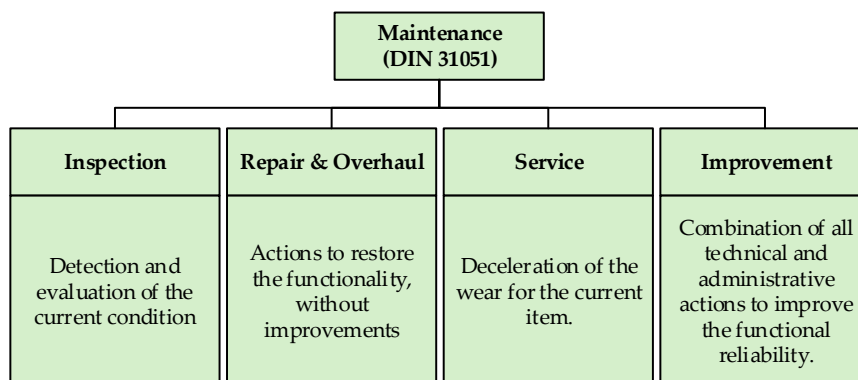


Figure 2.1: Tasks of maintenance adapted from DIN31051:2012-09 (2012)

Figure 2.1 illustrates the four basic actions of maintenance. All executed tasks in the field maintenance can be assigned to one of them.

- **Inspection:** All activities are taken to identify and assess the actual state of an item including the determination of reasons for wearing and ensuing initiative consequences for the future.
- **Repair and Overhaul** is a physical measure which has to be performed to restore the function of a defect item.
- The **Service** is a measure to delay the degradation of existing wear reserves.
- The **Improvement** is the combination of all technical and administrative measures, as well as all activities from the management which increase the reliability, safety or maintainability of an item without changing its original function.

As shown in figure 2.2 the basis to define maintenance objectives are the given company's goals. This corporate goals are mainly influenced by the factors leadership, environmental

protection and safety. Prevention of technical problems, consequential damages, damage to health and the increase of productivity are the main management benefits of maintenance. (Strunz, 2012; Renkes, 1993)

From this higher level, objectives for area and system maintenance are defined. The obvious target of maintenance is to reduce technical breakdowns and thus increase the productivity. To achieve this objective, different maintenance strategies can be used, which will be discussed in chapter 2.1.3. After the selection of the right maintenance strategy has taken place, the optimal maintenance cycles, which have to be applied to the individual units, are chosen. These cycles are often defined by experience. Innovative systems use Big Data analysis to identify strategies and cycles. The next step is to find out a cost-optimal coordination strategy on the system level. This can be achieved through either centralization or decentralization. When the strategy is defined, an efficient maintenance plan on the area level has to be selected. Finally, an optimal maintenance plan on the system level has to be developed. (Strunz, 2012, pp.2-4)

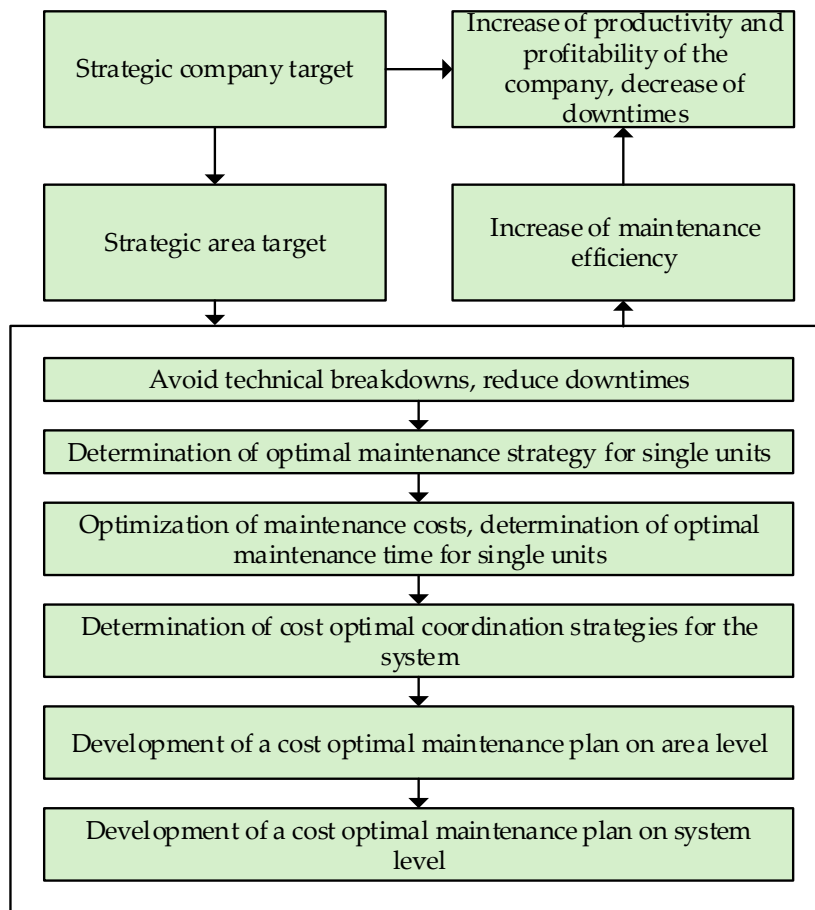


Figure 2.2: Maintenance objectives adapted from Strunz (2012, pp.3)

2.1.2 Terms of maintenance

In this section important terms of maintenance will be defined.

- **Availability**

”is defined as the probability that a component or system is performing its required function at a given point in time when used under stated operating conditions.” (Ebeling, 1997, p. 6)

The availability can be calculated in different ways. It always depends on what is regarded as planned production time. For the calculation of the Overall Equipment Effectiveness (OEE) the planned production time must be reduced by the planned downtime. Planned downtimes are maintenance tasks which are scheduled previously. (Nakajima, 1988, p. 22)

$$Availability = \frac{Run\ Time}{Planned\ Production\ Time}$$

The availability can also be calculated with mean times. According to Ebeling (1997, pp. 254-256), there are four different types of availability.

Inherent availability can be seen as an equipment design parameter. It is equivalent to the availability of the OEE calculation.

$$A_{inh} = \frac{MTBF}{MTBF + MTTR}$$

Achieved availability is a key figure for the overall effectiveness of an equipment and the maintenance strategy. Compared to the inherent availability it also includes the preventive maintenance influence of planned downtimes onto the availability. \bar{M} stands for the mean system downtime.

$$A_a = \frac{MTBM}{MTBM + \bar{M}}$$

Operational availability also includes the supply and maintenance delays as part of unplanned downtimes. \bar{M} is calculated by replacing MTTR with MTR= MTTR+SDT+MDT.

$$A_o = \frac{MTBM}{MTBM + \bar{M}'}$$

Generalized operational availability is used if the system is not operating continuously. The interval times of time to preventive maintenance and the time to failure are measured in the operating time.

$$A_g = \frac{MTBM + \text{ready time}}{MTBM + \text{ready time} + \bar{M}'}$$

- The **Performance** is an indicator for the output of the system. It calculates the speed which a machine actually runs compared to its defined speed. Losses at the performance are often speed losses at the machine. (Stamatis, 2010, p.26) The performance is a product of the operating speed rate and the net operating rate. (Nakajima, 1988, p. 24)

$$\begin{aligned} \text{Operating speed rate} &= \frac{\text{Ideal Cycle Time}}{\text{Actual cycle time}} \\ \text{Net operating rate} &= \frac{\text{Total Count} \cdot \text{Actual cycle time}}{\text{run time}} \\ \text{Performance} &= \frac{\text{Ideal Cycle Time} \cdot \text{Total Count}}{\text{Run Time}} \end{aligned}$$

- **Quality rate:** This ratio determines the losses related to quality fails. It is calculated by the number of good parts in relation to the number of total parts. It is also often defined as the process yield of a machine. (Stamatis, 2010, p.26)

$$\text{Quality rate} = \frac{\text{Good Count}}{\text{Total Count}}$$

- **Reliability**

”is defined to be the probability that a component or system will perform a required function for a given period of time when used under stated operating conditions.” (Ebeling, 1997, pp. 5-6)

Reliability is a measure for the probability of a non-failure time. The difference between reliability and availability is that the latter one is a measure for the probability that the equipment is in a non-failure state, even though it has failed previously and has been repaired.

- **Maintainability**

”is defined to be the probability that a failed component or system will be restored or repaired to a specified condition within a period of time when maintenance is performed in accordance with prescribed procedures.”
(Ebeling, 1997, p. 6)

- **Mean Time Between Failures (MTBF)** is the average time lasting from the repair to the next failure of a maintainable unit.

$$MTBF = \frac{\sum \text{start of unplanned downtime} - \text{start of uptime}}{\text{number of breakdown}}$$

- **Mean Time Between Maintenance (MTBM)** Mean Time Between Maintenance: In addition to MTBF it includes preventive maintenance tasks.

$$MTBM = \frac{\sum \text{start of downtime} - \text{start of uptime}}{\text{number of maintenance activities}}$$

- **Mean Time To Repair (MTTR)**: Is the average time to repair the failed technical unit and set it back to the usual operating conditions.

$$MTTR = \frac{\text{Total maintenance time}}{\text{number of repairs}}$$

- **O.E.E (Overall Equipment Effectiveness)**: The OEE is calculated by the multiplication of the three key figures Performance, Quality and Availability. Each part is one aspect of the process. Therefore it is a measure for the overall value of a process. The OEE could be calculated for single machines but it could also be rolled up to the department or plant level. A common value for the OEE is around 85%, depending on the industry. (Stamatis, 2010, p.26) A detailed description is given in section 2.1.4 on page 13.

$$OEE = \textit{Availability} \cdot \textit{Performance} \cdot \textit{Quality}$$

2.1.3 Maintenance strategies

According to Strunz (2012, p.294), there are two basic definitions for the overall term strategy and the specific term maintenance strategy. A **strategy** is a general concept to reach a specific goal. **Maintenance strategy** aims to achieve a general concept for the preservation of a specified availability of units from a technical feasible and economical reasonable viewpoint.

Figure 2.3 illustrates the three basic maintenance strategies.

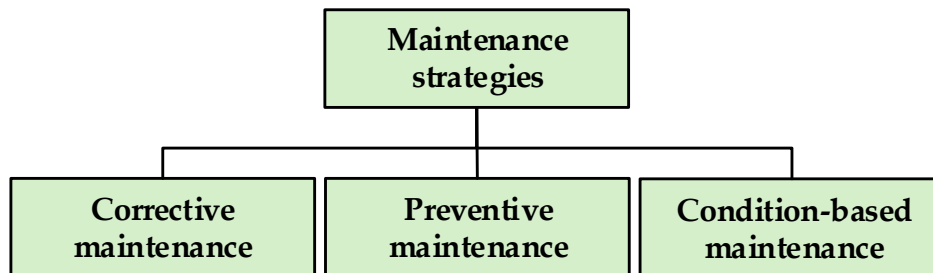


Figure 2.3: Classical Maintenance strategies adapted from Eichler (1990, p.149)

- **Corrective maintenance:** After the detection of a failure and the related downtime, maintenance is carried out. The main objective of maintenance is to restore the usual operating conditions. The downtime can diversify depending on the unpredictability of the failure time.(Eichler, 1990, p.151) The benefit of this approach lies in the optimal utilisation of the wear reserve. This strategy is useful if the sustained costs for downtime and repair are lower than the investment costs. It is usually used for small and uncritical items. (Strunz, 2012, p.295)

The disadvantages are:

- Unpredictable downtimes and repair schedules
- Long downtimes
- Losses in throughput and other process steps
- Possibilities of consequential damages

- **Preventive maintenance:** Maintenance tasks are performed with a predetermined schedule regardless of the degree of the wear reserve. This strategy aims to reduce operating failures with additional maintenance activities.(Eichler, 1990, p.154)

The advantages compared to corrective maintenance are:

- Avoidance of unpredictable downtimes
- Possibility to plan maintenance tasks to a large extent
- Reduction of downtime losses

The disadvantages compared to corrective maintenance are:

- No complete utilization of wear reserves
- Higher space effort

- **Condition-based maintenance:** A condition-based maintenance action is carried out if the item exceeds a specified degree of wear. The actual item condition is continuously assessed by sensors or human inspections. The inspections take place at a periodic or non-periodic frequency. (Mobley, 2004, p.10) Eichler (1990, p.160) determined following advantages and disadvantages compared to preventive maintenance:

Advantages:

- Better utilization of wear reserves

Disadvantages:

- Accurate information about damage behaviour, wear process and damage limits are necessary
- Sufficient procedures of technical diagnostics and right predictions of wear process are necessary

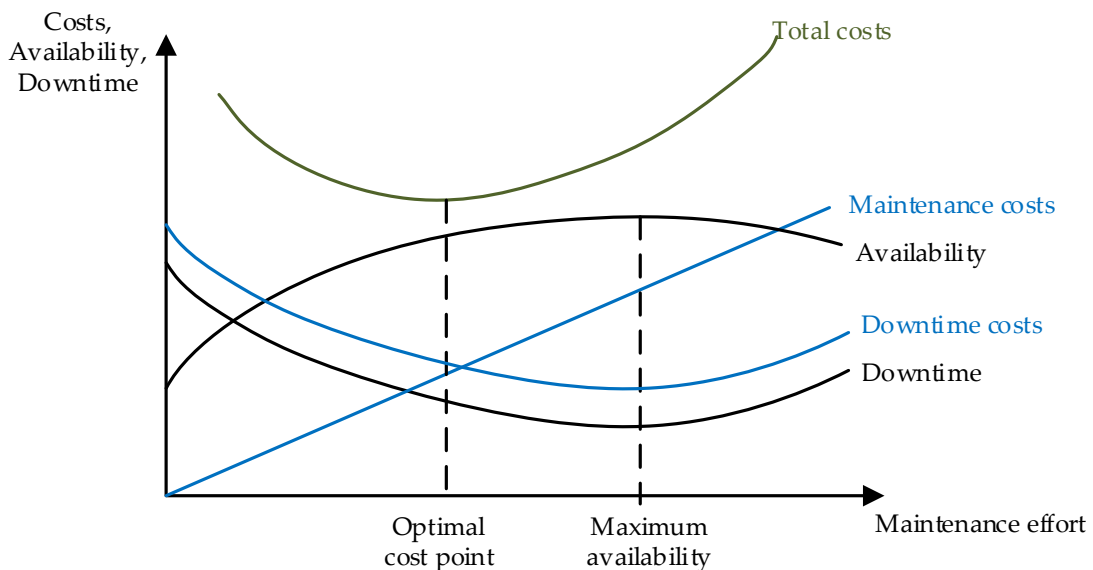


Figure 2.4: Maintenance costs modified from Strunz (2012, p.18)

Figure 2.4 shows the influence of maintenance on the availability and the total cost of the equipment. The availability starts to increase the higher the maintenance effort is. After a specific point the downtime through preventive maintenance gets so intense that the availability decreases again. The total cost is a function of maintenance cost and

downtime costs. An optimal cost strategy does not necessarily mean that the availability is at a maximal level. This has to be considered at the selection of an optimal maintenance strategy.

In the last decades special types of these strategies have been developed:

The **predictive maintenance** means maintenance supported by monitoring devices and different analysing techniques to predict optimal maintenance times. (Nakajima, 1988, p.10)

Risk based maintenance: The objective of risk-based maintenance is to reduce the maintenance effort with consideration of fixed safety levels. Through the determination of economic losses and risks for humans and the environment, the most effective maintenance strategy has to be chosen. The financial risk is calculated through the extent of damage and the probability of occurrence. The basis is a detailed risk analysis of every equipment. (Bandow and Schaefer, 2009, p.741)

Total productive maintenance is a concept which was developed in the 1960s in Japan. The main objective is to decrease the unplanned downtime to zero. As the project partner is currently enhancing this strategy it will be discussed in detail in the subsequent chapter.

2.1.4 TPM

According to Nakajima (1988, p.1), the inventor of TPM, the definition of total productive maintenance is,

”productive maintenance carried out by all employees through small group activities.”

The two main goals of TPM are zero breakdowns and zero defects.

The aim of TPM is to maximize the equipment effectiveness. It integrates a system of preventive maintenance which covers the total life-cycle of the equipment and all departments. TPM affects all employees from the top management to the workers on shop floor level. It supports planned maintenance through motivation management. (Bikash, 1998)

With the implementation of TPM, worker productivity increased by 60 percent and unplanned breakdowns were reduced from 1/50 to 1/500. The equipment operation

rates increased by 17-26 percent and process defects were reduced by up to 90 percent. (Nakajima, 1988, p.xviii)

According to Nakajima (1988, p.xix), the three most important features of TPM are:

- activities to maximize equipment effectiveness
- autonomous maintenance by operators
- company-led small group activities

Furthermore, TPM includes: (Nakajima, 1988, p.10)

- maximization of overall equipment effectiveness (OEE)
- the development of a system of productive maintenance for the life of the equipment
- the involvement of all departments and areas of a company in the implementation of TPM
- the involvement of all employees company-wide
- the promotion of TPM through motivation management

A major difference between TPM and the common preventive maintenance is the involvement of the operators in maintenance tasks.

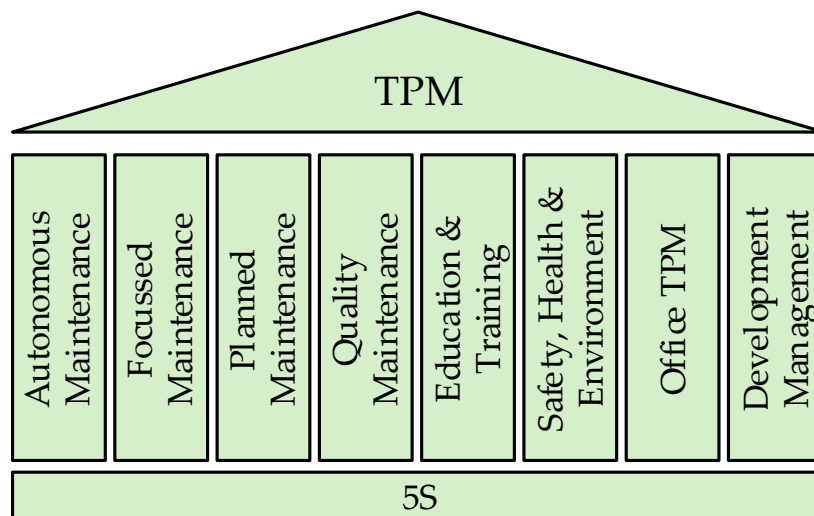


Figure 2.5: Eight pillars for TPM implementation adapted from Ahuja and Khamba (2008)

As shown in figure 2.5, the implementation of TPM can be divided into eight pillars. The basis for a successful implementation are the 5S, upon which the eight pillars, suggested by Japan Institute of Plant Maintenance (JIPM), are realized:

- **Autonomous Maintenance** means that the employee is responsible to carry out some maintenance tasks on a daily, weekly or monthly basis. The advantage is that the employees on the shop floor feel responsible for the machine. It improves the deterioration of the item. (Nakajima, 1988, p.72)
Furthermore, Nakajima (1988, p.76) suggest seven steps to implement autonomous maintenance:
 - Cleaning
 - Determination and implementation of countermeasures at the root problems
 - Definition of standards for cleaning and lubrication
 - Inspection routines
 - Self inspections
 - Organization and tidiness
 - Full autonomous maintenance
- **Focussed Maintenance:** One part of TPM is to identify the seven types of waste and eliminate them subsequently. (Stamatis, 2010, p.3) The seven types are:
 - Unnecessary transportation
 - Inventories beyond the absolute minimum
 - Unnecessary motions of employees
 - Waiting for the next process steps
 - Overproduction ahead of demand
 - Over-processing
 - Production of defective parts
- **Planned Maintenance:** Planning maintenance task ahead is to achieve zero breakdowns and to coordinate autonomous maintenance with tasks from the maintenance department. (Nakajima, 1988, p.86)
- **Quality Maintenance** means the improvement of the quality of maintenance with tools like Poka Yoke. Losses through humans should be prevented. (Willmott, 2001, p.15)
- **Education and Training** should be introduced to increase the maintenance skills of employees. This leads to a better understanding of the equipment and to more efficient maintenance. (Nakajima, 1988, p.90)
- **Safety, Health and Environment:** This pillar aims to achieve zero work-related accidents. (Ahuja and Khamba, 2008)

- **Office TPM:** The introduction of TPM, also in the office area, is to achieve a similar improvement as on the shop floor. Furthermore, the concept of TPM in the whole company is to be established. (Ahuja and Khamba, 2008)
- The **Development Management** helps to reduce losses occurring throughout the implementation phase of a machine. (Nakajima, 1988, p.96)
- **5S:** The basis for TPM are the 5S, which were also invented in Japan by Toyota. The 5S describe a workplace organization method. (Stamatis, 2010, p.15)
 - Seiri: Sort - All items which are not necessary for the workplace have to be sorted out.
 - Seiton: Set In Order - All remaining items are put into a specific place, taking ergonomic and process aspects into account.
 - Seiso: Shine - All employees are responsible for cleaning their own workplace and for keeping the workplace save. In addition, the working should be easy.
 - Seiketsu: Standardize - Set up standards for orderliness, processes and the whole workplace.
 - Shitsuke: Sustain - Self-discipline to preserve the standards and continuous improvement of the existing ones. Performing audits frequently.

To achieve a high overall equipment effectiveness, TPM minimizes the six big losses. (Nakajima, 1988, p.14)

- Equipment failure - breakdown of machines
- Setup and adjustments - losses through exchange of tools
- Idling and minor stoppages - the result of small problems during the operation
- Reduced speed - due to differences of the actual speed to the designed speed
- Process defects - because of scrap and rework of parts
- Reduced yield - a process which takes place/is carried out in the time of setup in order to stable the production.

Figure 2.6 shows how the overall equipment effectiveness and the six big losses are related. The losses of failing equipment and setup/ adjustment tasks decrease the availability of the equipment. Minor stoppages and reduced speed at the equipment have an influence on the performance of a system. The last two losses through defective parts and a reduced yield influence the third point of the OEE quality rate.

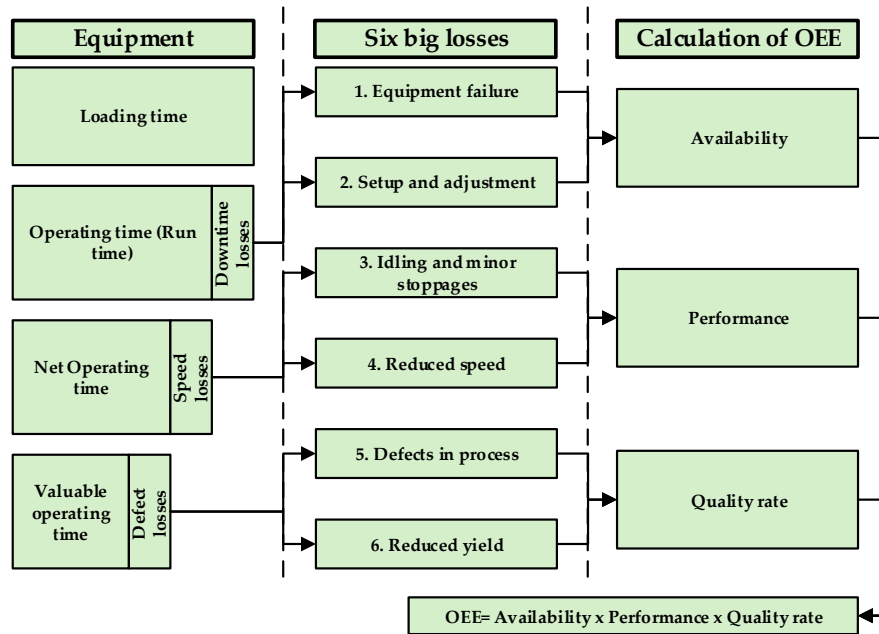


Figure 2.6: Overall Equipment Effectiveness and Goals adapted from Nakajima (1988, p.25)

2.2 Optimization Problems and Algorithms

As stated in the previous section TPM, the pillar "Planned maintenance", has the purpose to schedule tasks ahead and to align production with maintenance. Such a maintenance plan can be achieved by optimization algorithms, which will be introduced in this chapter. First, the problem of scheduling will be outlined. It gives a basic explanation of the problem. Second, linear programming and the special case of integer programming will be discussed. This two optimization programs are often used to solve scheduling problems. Finally, actual optimization algorithms for maintenance scheduling will be reviewed.

2.2.1 Introduction to scheduling

According to the Oxford Dictionary (2016b) a schedule is "a plan for carrying out a process or procedure, giving lists of intended events and times."

Baker and Trietsch (2009, p.2) define a schedule as a tangible plan which tells when things should happen. It shows exactly when events should occur. To create such a plan, the sequence of the events is important. Which event should occur first and which second? Scheduling is defined by the process of generating a schedule.

Conway et al. (1967, p.1) give a basic explanation of the sequencing problem: There are two jobs, A and B, which have to be scheduled, whereby α is the aggregate consequence of the sequence A first and B second. β is the aggregate consequence of the sequence B first and A second. If α is more preferable than β , the sequence will be A first and B second. No matter in which field the problem of scheduling occurs, a fundamental similarity to the given explanation of sequencing exists.

In the field of optimisation, Pinedo (2002, p.1) defines scheduling as the allocation of resources over time with specific constraints. Given the fact that in production tasks are jobs, they have to be processed by resources like machines. The objective of scheduling is to optimize the sequence. The problem could have one or more objectives, which have to be optimized. In the course of the production scheduling, e.g., the minimization of production costs can turn out as the objective.

According to Conway et al. (1967), a scheduling problem contains input and output variables with specific attributes. In the production planning optimization the fundamental problem starts with a set of jobs $1, 2, \dots, n$, which have to be scheduled on a set of machines $1, 2, \dots, m$.

A job i has the following attributes:

$r_i \dots$ ready time or arrival time, indicating the earliest start of a job

$d_i \dots$ the due date defines the latest point in time a job could be finished

$a_i = d_i - r_i \dots$ is the possible duration in which a job can be done

$g_i \dots$ is the number of operations a job i has

$m_{ij} \dots$ identifies which job has to be done on which machine

$p_{ij} \dots$ is the time a job i needs on the machine j

$p_i = \sum_{j=1}^{g_i} p_{ij} \dots$ total processing time of a job i

The output of a scheduler is the start time of a job i . Conway et al. (1967), define this time as waiting time because the job has to wait until the processing starts. The following variables define the output:

$W_{ij} \dots$ waiting time of a job i on the machine j

$W_i = \sum_{j=1}^{g_i} W_{ij} \dots$ total waiting time of a job i

The final result of a scheduling problem is specified by a set of W_{ij} .

This explanation defines the basic model for the given problem. It is used as the central framework for the definition of the maintenance task scheduler in chapter 1.2. The next sections will show different ways to set up a mathematical model for this explanation.

2.2.2 Linear programming

One method to solve the problem of sequencing is linear programming. This sub section contains the fundamentals of this method and the three steps to create a linear model. Furthermore, linear programming serves as a basis for integer programming, which will be discussed in chapter 2.2.3. To conclude this chapter, an overview of bounds and axioms of linear models is given.

Before the linear model can be defined, the terms *mathematical model* and *linear programming* have to be explained.

Dantzig (1966, p.8) defines mathematical model as a formulation of a system with mathematical relationships which characterize the set of feasible solutions. He also states "the linear programming problem is to determine the values of the variables of the system that (a) are nonnegative or satisfy certain bounds, (b) satisfy a system of linear constraints, and (c) minimize or maximize a linear form in the variables called an objective."

This formulation of a linear programming problem leads to the following mathematical model. The linear model has an objective z , which has to be minimized or maximized with decision variables x and a set of constrains b .

In standard form:

$$\begin{aligned}
 \text{Objective function: } \min/\max z &= c_1 \cdot x_1 + c_2 \cdot x_2 + \dots + c_n \cdot x_n \\
 \text{Constrains: } b_1 &\geq a_{11} \cdot x_1 + a_{12} \cdot x_2 + \dots + a_{1n} \cdot x_n \\
 b_2 &\geq a_{21} \cdot x_1 + a_{22} \cdot x_2 + \dots + a_{2n} \cdot x_n \\
 &\vdots \quad \quad \quad \vdots \quad \quad \quad \vdots \quad \quad \quad \vdots \quad \quad \quad \vdots \\
 b_m &\geq a_{m1} \cdot x_1 + a_{m2} \cdot x_2 + \dots + a_{mn} \cdot x_n \\
 x_1 &\geq 0 \quad x_2 \geq 0 \quad \dots \quad x_n \geq 0
 \end{aligned} \tag{2.1}$$

In canonical form:

$$\begin{aligned}
 \max/\min \quad & c^T x \\
 \text{subject to} \quad & Ax \leq b \\
 & x \geq 0
 \end{aligned}$$

To formulate such a model, Dantzig and Thapa (1997, p.11) suggest the following three steps:

- Step one: *Define the decision variables*. The first step is to decompose the entire system in its basic functions, activities or processes. For each of them a unit to measure has to be found. Finally, the decision variables of the system have to be defined. Often they are quantities like parts to produce, buy, etc.. These variables are stated as x_1, x_2, x_n .
- Step two: *Define the Item Set*. Step two is to define classes of objects which are required as inputs or outputs. They are also needed to choose a unit to measure. The important objects to consider are potential bottlenecks of the system. The next and final step is to select one item that the quantity produced as a whole measures the cost. The type of item i is usually labeled by i .
- Step three: *Set Up Constraints and the Objective Function*. At the beginning of this step, all constraints, which influence the bottleneck, have to be written down. A relation of how the decision variables are influenced by the constraint have to be set up. In the end the objective function by summing up the multiplications of the decision variables with their costs is written down.

After the model of the problem is formulated, it has to be solved with an algorithm. Common algorithms are the simplex method, the interior point method or the column generation algorithm. The problems that appear at linear programming vary in size, from small to large problems. This size is defined by the number of constraints. Special software has been developed to solve large systems, like linear programming models can be in practice. There are also other special tools that help to organize and direct the generation of coefficients from essential data. These tools are called matrix generators and have been developed due to the process of how models are being managed as well as the increase of the extent of solvable models. Main features of these programs are the formulation and update of the model on the one hand (input), the display of the result (output) like in graphs on the other hand. This makes the outcome easier to understand and use in practice. (Dantzig and Thapa, 1997, p.2)

Specific **bounds** for the decision variables may occur. Usually the decision variables in linear models are **non-negative**. For example it is not possible to produce less than zero products. This characteristic is known as the non-negativity assumption and is stated by the constraints $x_n \geq 0$. In the standard form of equation 2.1, the decision variables are defined as non-negative. In real world problems this is not always the case, therefore

upper and lower bounds can be set. (Dantzig and Thapa, 1997, pp.21-22)

With **upper and lower levels** decision variables could get negative. This could be important for financial applications. The quantity of a variable may also be limited by an upper bound. This constrain is be stated by $l_j \geq x_n \geq u_j$.

The **axioms** or assumptions for a linear program are proportionality, additivity and continuity. Other types of mathematical models do not satisfy these axioms, e.g. an integer program model does not does not satisfy the assumption of continuity.

Linear models always assume that the inflow of an system influences the outflow of a system in a **proportional** manner. If one buys two products of a variable, the cost per product will be the same as if two-hundred products were bought. Mathematically formulated it looks like this: If a_{ij} units of the item i are necessary for one unit level of activity j , x_j units of the activity j require $a_{ij} \cdot x_j$ units of the item i .

The **additivity** assumption implies that the objective function of a linear model is additively separable in its variables. No mixed variables occur. This can be described in a mathematical way for a production item as follows: If a_{ij} tasks of the job i are supplied with one element of the j^{th} part, and a_{ik} tasks of the job i are supplied with one element of the k^{th} part, $a_{ij}x_j + a_{ik}x_k$ tasks of the job i are provided by x_j elements of the j^{th} part and x_k by elements of the k^{th} part.

Continuity defines that activities and variables may be any real numbers in their defined limits. If some activities or variables have to be of a finite set of variables, like integer values, the problem is not a linear programming model any-more. These problems could be reformulated to integer programs, which are commonly much harder to solve.

2.2.3 Integer programming

As stated in the previous section, the assumption of continuity is not valid for integer programming. It is an optimization problem with which some or all off the variables are restricted by integers. Integer programming is used in different areas like production planning or machine scheduling. In the special case of a linear integer problem(ILP) the objective function and the constrains of the optimization problem are linear. (Nemhauser and Wolsey, 1999, p.3)

Compared to the linear model in the previous section, the constrain that the decision variable x only can have the values of integers ($x \in \mathbb{Z}^n$) has to be added.

The definition of the special case of integer and real number variables is called the linear mixed integer problem (MIP). The mathematical formulation is follows:

$$\begin{aligned}
 & \text{maximize} && c^T x + hy \\
 & \text{subject to} && Ax + Gy \leq b \\
 & && x \geq 0 \quad y \geq 0 \\
 & \text{and} && x \in \mathbb{Z}_+^n \quad y \in \mathbb{R}_+^p
 \end{aligned}$$

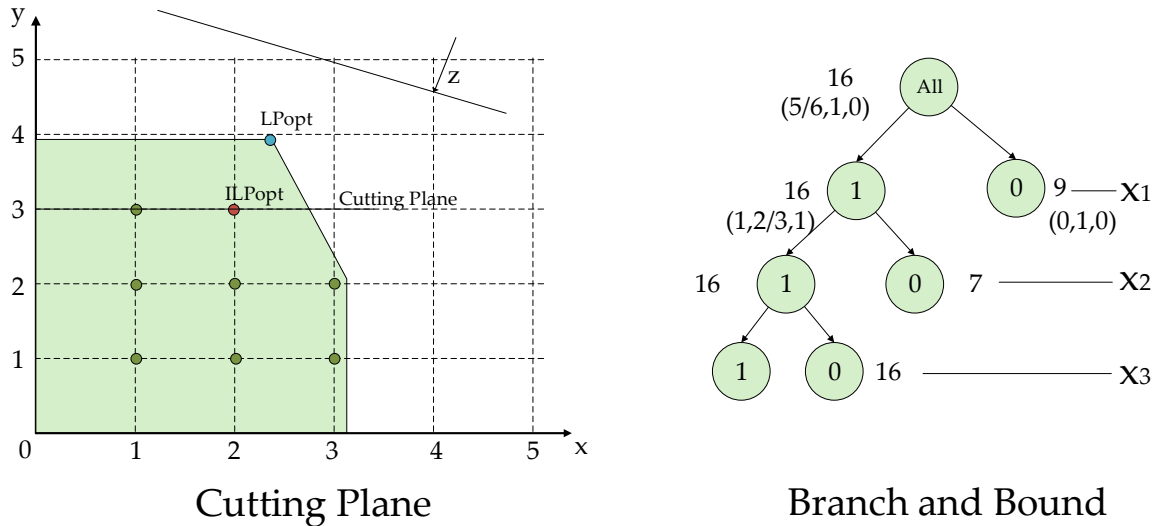


Figure 2.7: Cutting Plane and Branch and Bound Method

Figure 2.7 shows the linear programming solution (left) compared to the integer solution. This optimization problem is an example with the two decision variables x and y . The grey area symbolizes the feasible solution set for a linear problem. The blue point illustrates the optimal linear solution for the objective function z . As this is not an integer value, this is not a feasible solution for an ILP. The feasible solution is indicated by the green points and according to the objective function z , the red point illustrates the optimal solution. To obtain this integer solution, different algorithms exist. Throughout the next paragraphs, the methods **branch and bound** as well as **cutting planes** planes, which are needed to solve a problem, will be explained. In advance a brief explanation of the **simplex algorithm** for linear problems by Dantzig (1966) is given.

The **Simplex Algorithm** solves linear problems in standard form. The solution to the problem is found by the algorithm after a finite number of steps have been taken or after the insolubility or limitlessness of the problem has been noted. On the left-hand side of figure 2.7 the feasible solution for the linear problem is marked green. An optimal

solution would either be one on a corner point or infinite optimal solutions would be on the edge of a line. The simplex algorithm has two phases. Phase one is to find a feasible solution for the canonical form of the linear problem, whereas phase two is to optimize the objective function by changing the decision variable within the feasible solution. From a geometric point of view the algorithm would start at one of the corner-point solutions (CPF). The next step is to test the optimality of the point by calculation of the objective values of the adjacent CPFs. If none of these values is the better, an optimal solution is found, otherwise the objective values of the next CPFs, taken from the better corner, have to be calculated. (Hiller and Liebermann, 2001, pp.110-112)

The **Branch and Bound algorithm** for solving integer problems is based on the fact that for every bounded pure integer problem a finite number of solutions exist. It solves the problem by using a kind of enumeration procedure to find the optimal solution. The basic concept is to divide and conquer. The original large problem is divided into smaller sub problems. The first step of the branch and bound method is to remove all the integrality restrictions. The result is the so-called linear-programming relaxation of the original mixed integer programming problem. Afterwards this problem has to be solved with linear programming. If this already results in an integer solution, it will turn out to be the optimal one. If not, the next step is to branch a variable to obtain two sub problems. Afterwards the bounds of the problem have to be obtained by a LP relaxation, rounding up and off to the next integer solutions. For each of the sub problems the three fathoming tests have to be applied in order to figure out whether a variable should be further branched or not. The first test is shown in figure 2.7 on the right-hand side on level x_1 . For node $x_1 = 0$ the solution is already an integer solution with $(x_1, x_2, x_3) = (0, 1, 0)$. Therefore this is an optimal solution for the first node and further branching is not necessary. The second option to dismiss a node is, if the solution of the subproblem would be smaller than the current optimal solution. The example illustrates this case at the second level with $x_2 = 0$. The solution is 7 and therefore smaller than the current best solution of 9. The third way to dismiss a node is as follows: If the simplex method finds that the LP relaxation has no feasible solution. The algorithm can be ended if there are no remaining sub problems left and the current one is optimal. (Hiller and Liebermann, 2001, pp.605-608)

The **Cutting Plane Method** reduces the feasible region by introducing a new constrain without eliminating any possible feasible solution. Figure 2.7 shows a cutting plane with the mathematical constrain $y \leq 3$. One simple example is the following constrain $6x_1 + 3x_2 + 4x_3 + 5x_4 + 5x_5 \leq 13$. The decision variables are binary. Furthermore, a LP-relaxion has been executed and the decision variables have the following values $x_1 = 1, x_2 = 0, x_3 = x_4 = x_5 = \frac{3}{4}$. Hence it is not possible that $x_3 = x_4 = x_5 = 1$, since

$4 + 5 + 5 > 13$. Therefore the constrain $x_3 + x_4 + x_5 \leq 2$ is introduced. Consequently, the procedure of cutting planes is to consider any given constrain with the form \leq and non negative coefficients first. The second step is to find a group of variables which violate this constrain. The last step is to formulate the new constrain that the sum of the variables \leq sum of the variables - 1. (Hiller and Liebermann, 2001, pp.628-629)

Modeling with binary variables

In some integer problems it is necessary to model the decision variables as 0-1 variables. This represents a binary choice between the event occurring at $x = 1$ or the event not occurring at $x = 0$. In literature different problems depending on the situation are discussed. To give an insight into the modeling with binary variables, the common **Assignment Problem** will be discussed in detail.

The **assignment problem** deals with the problem of the allocation of n items (e.g. people) to m other items (e.g. jobs). This problem could be put into a linear form with the objective to minimize or maximize a function. Each item j can only be assigned to one item i . The cost to assign the item j to the item i is given. The objective is to minimize the total cost. (Nemhauser and Wolsey, 1999, p.5)

The assignment variable is defined by the representation of a binary choice:

$$x_{ij} = \begin{cases} 1 & \text{if the event occurs} \\ 0 & \text{the event does not occur} \end{cases}$$

The constrain that the job has to be done by one person only is $\sum_{j=1}^n x_{ij} = 1$ for $i = 1, \dots, m$. Since each person cannot do more than one job, the second constrain is $\sum_{i=1}^m x_{ij} \leq 1$ for $j = 1, \dots, n$. The decision variable is 1 if it is allocated at item j , otherwise it is zero. The objective function is defined by

$$\min/\max \sum_{i=1}^n \sum_{j=1}^m c_{ij}x_{ij}$$

c_{ij} defines the cost for example the cost of the assignment i to j for example.

2.3 Optimization in production and maintenance planning

This section reviews different approaches of maintenance scheduling. Van Horenbeek et al. (2010) introduced a maintenance optimization framework in their study. They state that many articles have been published and most of them focus on only one objective. They have discovered that there is a big gap between academic models and the models used in practice. Therefore maintenance optimization should not start with an model trying to apply it to find a corresponding application. Maintenance optimization should rather start with the application to which a possible optimization method is found. The framework in figure 2.8 outlines the used techniques and parameters in maintenance optimization.

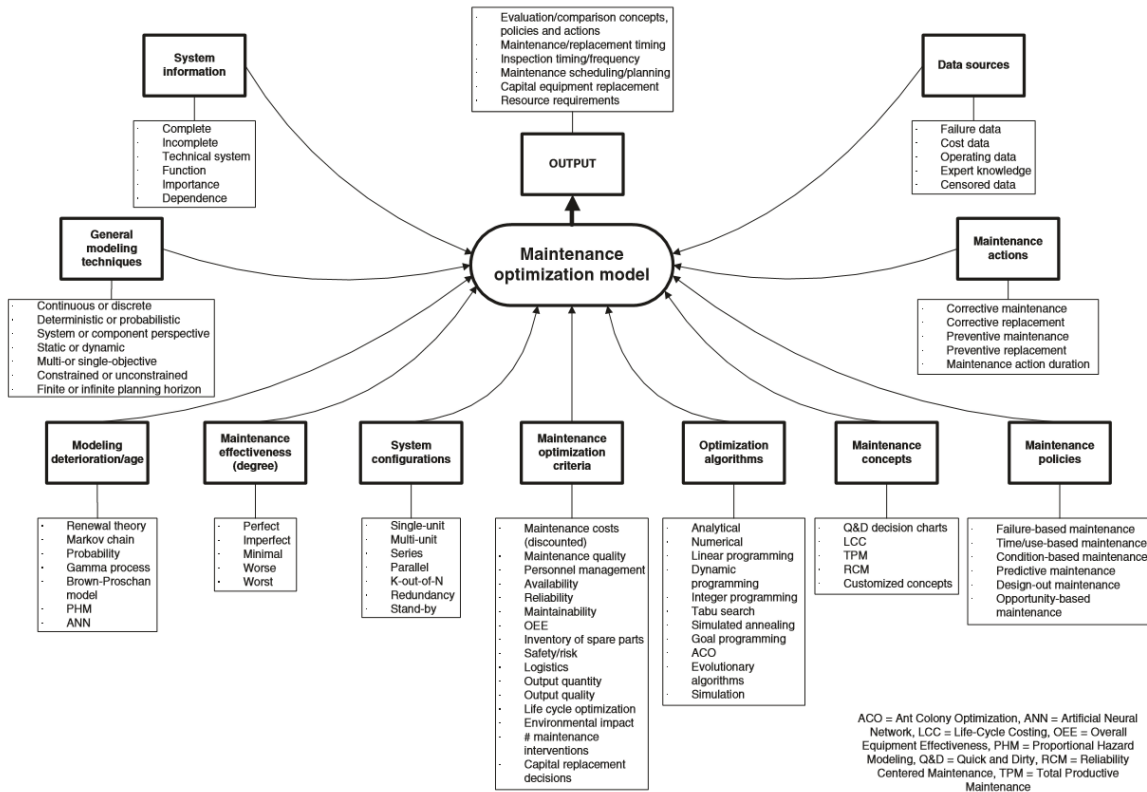


Figure 2.8: Maintenance optimization classification framework (Van Horenbeek et al., 2010)

Manzini et al. (2015) show the possibility of a resource-constraints mixed integer linear programming model for maintenance scheduling. The objective of their study was to

minimize the cost for preventive maintenance by optimizing the assignment of pre-defined tasks to pre-defined time buckets. They assumed that the duration of a task is deterministic and costs for spare parts, personnel and failure costs are known. They also take unplanned costs for tasks with a failure probability function into account. A case study was done with the objective to minimize the global cost for a maintenance service provider. The model was solved by Gurobi solver with a maximal task and a bucket size of 40.

Ebrahimipour et al. (2013) investigated the preventive maintenance scheduling in a multi production line. The multi objective function takes the reliability of the production lines, the maintenance costs and the downtime of the system into account. The availability of employees, the spare parts and the periods for maintenance are considered with a threshold. The production lines in the paper consist of serial and parallel machines. The failure rate of the components was assumed with a Weibull distribution ($\lambda(t) = \beta\lambda(\lambda t)^{\beta-1}$). The tasks were planned in a time horizon of $[0, T]$, and J periods were introduced with a length of J/T to get discrete intervals. After the replacement of a component had taken place, the failure rate was set back to no failure state. The proposed algorithm was tested on three different production lines with a number of time periods of 30, 90 and 300.

Moghaddam and Usher (2011) present a model for the optimal preventive maintenance scheduling with the methodology of dynamic programming combined with branch and bound algorithm. The objective was to minimize the total cost with subject to maximize the system reliability and budgetary constrains. They also split also the time horizon in a discrete number of intervals.

Wildeman et al. (1997) suggest a rolling horizon approach for grouping maintenance tasks on a short term basis. Maintenance activities often involve set up costs which are the same for all maintenance tasks. A grouping of tasks could decrease these set up costs. For the deviation from the optimal point in time for maintenance, they suggest to introduce a penalty cost function.

Canto (2011) discusses the problem of the power plant maintenance scheduling with a 0-1 mixed integer linear programming model. The objective was to maximize the reliability of the power plant. The model was also applied to a practical example. The time horizon was also split into intervals, which is explained in greater detail in the second and third paper. The constrains covered mainly time depended restrictions.

Cassady et al. (2005) deal with the integration of preventive maintenance activities in the production process of a single machine. Based on the fact that maintenance activities and production planning highly depend on one another, the chosen objective was to

minimize the total weighted completion time. Total enumeration was used to solve the problem. For larger problems they suggest a heuristic approach.

This section shows some of the actual researches in the field of maintenance optimization. The most problems are special cases and therefore the used algorithms and solutions are specified for them. However, it can be seen that integer programming and the objective to minimize the cost are a valid technique for maintenance scheduling problems. The upcoming chapter will give an introduction to simulation and the special case of discrete event simulation which is a common technique to simulate production systems.

2.4 Simulation

This sub-chapter starts with a brief introduction to simulation and modeling. Afterwards common simulation approaches are shown and the approach of discrete simulations is discussed in detail. Discrete simulations are used to model and optimize production systems. This technique is applied in the practical part for the opportunity window calculation and the according validation.

2.4.1 Introduction to Simulation

The Oxford English Dictionary Oxford Dictionary (2016a) defines the word simulation as:

The technique of imitating the behaviour of some situation or process (whether economic, military, mechanical, etc.) by means of a suitably analogous situation or apparatus, esp. for the purpose of study or personnel training.

Banks et al. define the same term as follows:

"A *simulation* is the imitation of the operation of a real-world process or system over time. Whether done by hand or on a computer, simulation involves the generation of an artificial history of a system and the observation of that artificial history to draw inferences concerning the operating characteristics of the real system." (Banks et al., 2004, p.3)

According to these two definitions, the objective of a simulation is to imitate the behaviour of a system from the real world. To create a simulation, the first step is to abstract the system variables with a conceptual model. It is important to get the level of abstraction

and the simplification of the real world right. Figure 2.9 shows the artefacts of conceptual modeling. The problem domain includes the real world and the system description. This description is generated by knowledge acquisition of the real world problem. Assumptions have to be made for uncertainties or beliefs of the system. The model has to be abstracted through simplifications to the conceptual model according to the system description. This model describes the objective, outputs, inputs, simplifications and assumptions of the system. The model design transfers the conceptual model into the computer model. At this stage constructs for the computer model as data or components are defined. Finally, the computer model is a simplified representation of the real world problem. (Robinson, 2013)

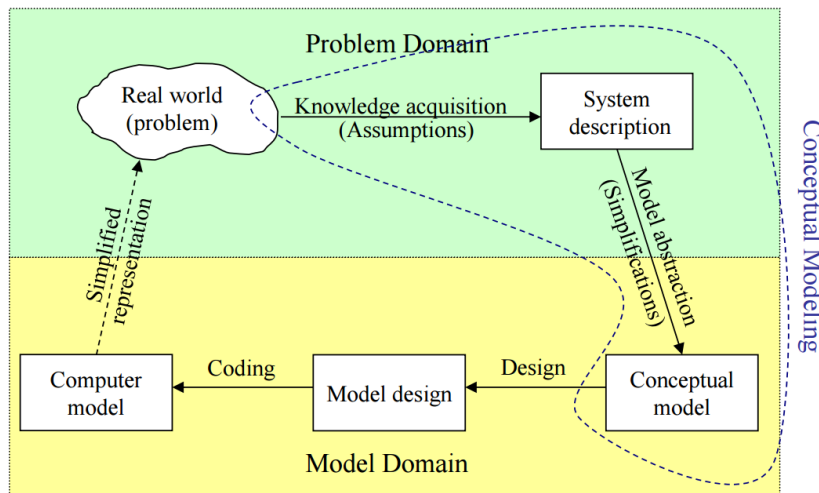


Figure 2.9: Artefacts of Conceptual Modelling (Robinson, 2013, p.381)

The process of a simulation study (Figure 2.10) can be divided into twelve basic steps: (Banks et al., 2004, pp.13-16)

Step1: Problem Definition

At the beginning of every study a clear problem definition should be given. The policy maker and the analyst should have the same picture of the problem.

Step2: Setting of objectives and overall project plan

The question, which should be answered in the simulation, leads to its objective. At this point in time a discussion if simulation is the appropriate technique to solve the problem should be held. Also the objectives should be discussed. If this is fact the project plan should include a statement of the alternative systems and also facts to the number of people involved, costs and expected milestones and the end of the study.

According to Birta and Arbez (2013, p.4) there are several main objectives for simulations:

- Comparison of control policy options
- Education and training
- Engineering design
- Evaluation of decision or action alternatives
- Evaluation of strategies for transformation or change
- Forecasting
- Performance evaluation
- Prototyping and concept evaluation
- Risk/safety assessment
- Sensitivity analysis
- Support for acquisition/procurement decisions
- Uncertainty reduction in decision making

Step3: *Model conceptualization*

This step can be compared with the conceptual model (figure 2.9). The key element of modelling is to abstract the essential features of the problem. It is not necessary to copy the whole system in its details, it is necessary to understand the key principles. An appropriate way is to start with a simple model of the problem and only then to increase the complexity with time.

Step4: *Data Collection*

Data collection is one of the most important steps. Data about processes, input and activities are necessary to understand and abstract the problem in the right way. This is the reason why one should start collecting data in the early phase of the simulation, parallel to the modelling step. The quality of the output highly depends on appropriate data.

Step5: *Model translation*

At this stage the conceptual model has to be transferred into a computer understandable language, into a so-called "program". The problem of the model translation often is the complexity of the model and/or the high amount of data. This means that the problem can only be solved with computational assistance. This step is comparable to the computer model (figure 2.9).

Step6: *Verification*

The next step is a decision level. After the model has been translated, it has to be verified if the program is operating properly. In a highly complex system this often leads to a

high amount of debugging time. If the logic of the program and its input parameters represent the model correctly, this stage is completed. Otherwise the model translation has to be adapted.

Step7: *Validation*

The validation of the program and the model is carried out by adjustments of the model. It is an iterative step by comparing the output of the program to the real world. If the validation is incorrect, there are two possibilities: One is that the conceptual model has to be changed to a specific degree, the other one is that data, for example input parameters, are wrong.

Step8: *Experimental design*

The alternatives, which could be simulated, have to be determined. It is important to observe different scenarios. Decisions of the number and length of runs have to be made for the next step.

Step9: *Production runs and analysis*

Production runs have to be made to estimate the performance of the simulation and associated measures to improve the program.

Step10: *More runs?*

The result of the analysis of the production runs should determine if more runs are needed. Also, the specific design of these experiments have to be determined.

Step11: *Documentation*

This is a very important step. On the one hand documentation has to be done for the program, on the other hand for the progress. The program documentation deals with the content of the program. The progress documentation includes the information of the simulation steps and the history of the simulation.

Step12: *Implementation*

The last step is implementing the results obtained from simulation. For the successful implementation it will be necessary that the model user has already been involved in the previous steps.

The most important step of this twelve is the validation phase. Because an invalid model will always lead to an unsatisfactory result which could be at the end dangerous and costly.

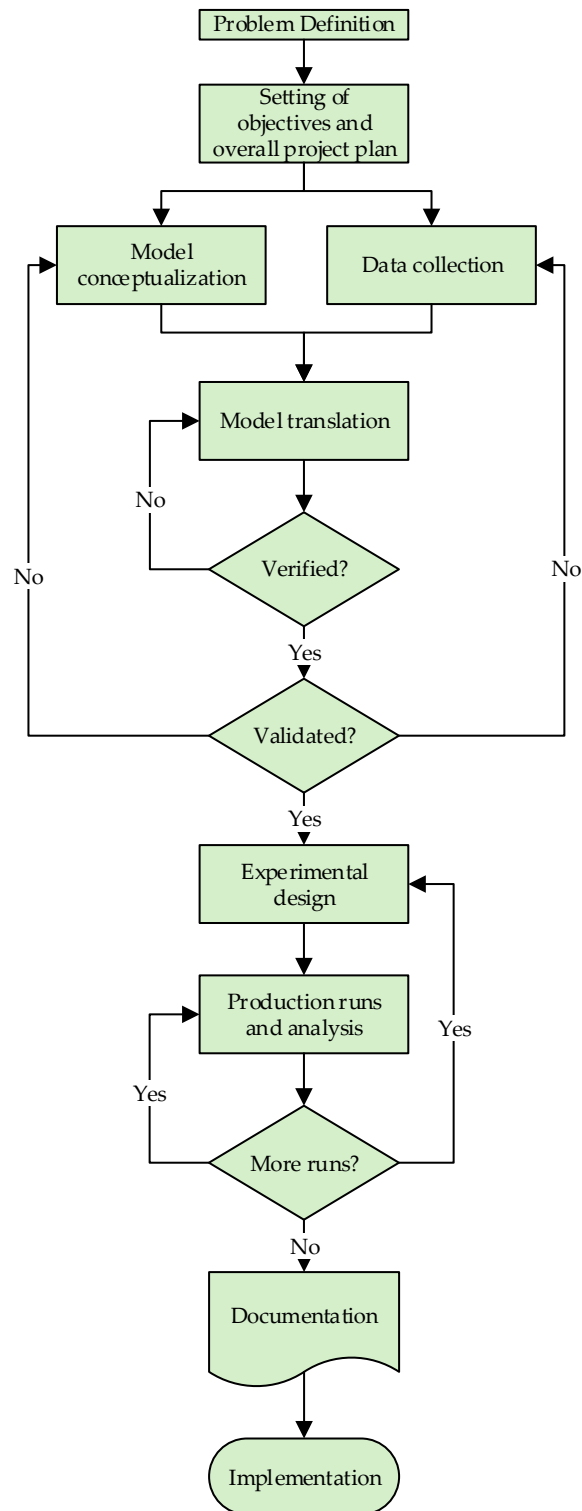


Figure 2.10: Steps in a simulation study adapted from Banks et al. (2004, p.13)

2.4.2 Simulation Approaches

Figure 2.11 illustrates the different simulation modeling approaches according to Borshchev and Filippov (2004, p.3). The approaches can be distinguished in their abstraction level. System Dynamics focuses on a strategic, macroscopic level. Discrete event simulations acts mainly lie their focuses on middle to low abstraction levels. Agent based simulation could be used on all kinds of levels. For the thesis the discrete event approach is used and therefore only this simulation technique will further be discussed in detail. DES is chosen on the one side because production lines work mainly discrete on a low abstraction level in contrast to System Dynamics. On the other side it is not necessary to model the production line with agents, because the complexity of the behaviour from the system on entity level is low.

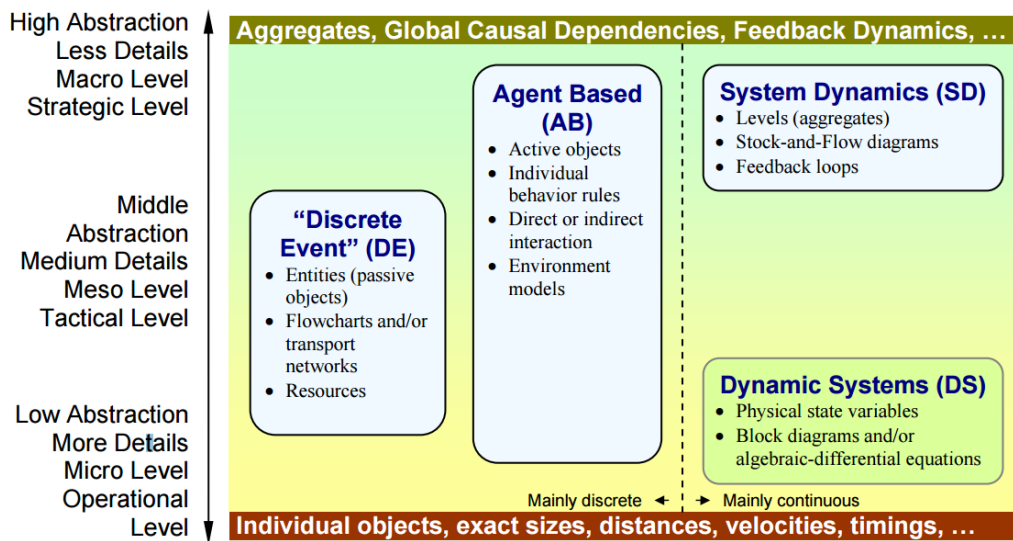


Figure 2.11: Approaches in Simulation (Borshchev and Filippov, 2004, p.3)

In figure 2.11 the approaches are divided at the bottom into discrete and continuous simulations. Figure 2.12 illustrates the difference between these two. Furthermore, a classification for the field of simulation is shown. The four main types of simulation are Continuous Variable Dynamic Systems, Discrete-Time Dynamic Systems, Discrete Event Dynamic Systems and Discrete Dynamic Systems. On the one hand the figure shows the time base for the simulation. On the other hand it presents the values of the state variables. As illustrated the time could evolve continuously or discretely in time steps. Also the variables could change continuously over time or discrete. (Wainer, 2009, pp.15-16) System Dynamics and Dynamic Systems mainly use continuous variables. Discrete Event simulation and Agent Based simulation are mainly discrete.

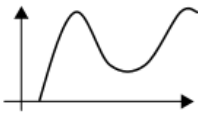
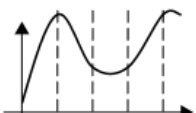

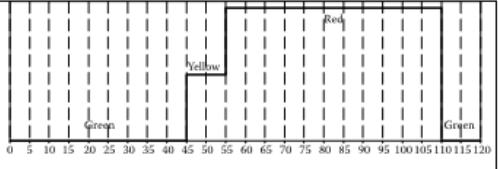
Vars./Time	Continuous	Discrete
Continuous	<p>{1} Continuous Variable Dynamic Systems</p> 	<p>{2} Discrete-Time Dynamic Systems</p> 
Discrete	<p>{3} Discrete-Event Dynamic Systems</p> 	<p>{4} Discrete Dynamic Systems</p> 

Figure 2.12: Classification according to the representation of time bases/state variables (Wainer, 2009, p.16)

According to Banks et al. (2004, p.61) parameters of discrete processes are:

- The **system** is a combined structure of entities which interact with each other within defined boundaries and pursue a specific goal.
- The **model** is an abstraction of the system describing the essential relationships, entities, states, attributes, processes, activities, events and delays in a structural, logical or mathematical way.
- The **system state** can be defined as a collection of information describing the state of the system at a specific time.
- The **entity** is an object or component of a system (e.g machine, person).
- The **attribute** defines the characteristic of an entity (e.g. priority of an entity in a waiting queue, routing of a job in a job shop).
- A **list** means a collection of connected entities permanently or temporary (e.g. customers ordered in a waiting list through a priority).
- The **event** means a change of the system state (e.g customer leaves the waiting queue).
- The **event notice** is a record which includes the event type, event time and data which is necessary to execute an event.
- A **event list** is a list of all upcoming event notices, ordered by time.
- An **activity** is a state of an entity for a duration of time with a specified length.
- A **delay** is a duration of time with an unspecified length. The duration is unknown until it ends.
- The **clock** is the variable which defines the simulated time.

Hedtstück (2013, p.22) defined Systems (3) and (4), as in figure 2.12, as the discrete event simulation and the time driven simulation.

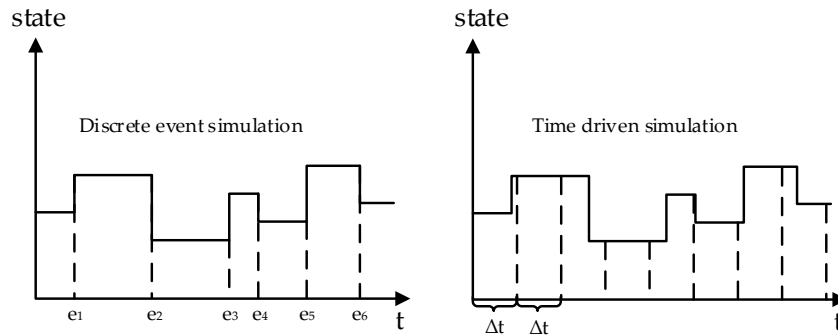


Figure 2.13: Discrete event simulation vs. time driven simulation adapted from Hedtstück (2013, p.22)

Discrete event simulation DES

Discrete event simulation is defined by a list of events. This list contains a data pair of the event name and time. The contents are sorted in chronologically ascending order. This list is updated dynamically as the simulation proceeds. (Zeigler, 1976, p.153) As shown in figure 2.13, compared to time driven simulation, the time steps are flexible according to the event occurrence. Algorithm 1 shows the basic steps of a simulation in pseudo code. At the beginning all initial values as well as the simulation time is set. Afterwards a loop is started with the condition that the actual time has to be less or equal to the end of the simulation. The next event on the event list is picked and the actual one is deleted. The time is updated to the start time of the next event. The actual event type is set to the event type of the next event and the event notice of the actual type is performed.

Algorithm 1: Discrete event simulation

Input: simEnd, simTime=0, initial variables

while $simTime \leq simEnd$ **do**

 Get next event **nextEvent** from event list and delete this event;

 simTime = „Start of” **nextEvent**;

 typeActual = Event type of **nextEvent**;

 Perform event notice to typeActual

end

Time driven simulation

If the state of the system is not relevant after a single event but after a specific period, the time driven simulation will be chosen. By updating the system state after this fixed incremental change of time the dynamic of the system can be illustrated. (Hedtstück, 2013, p. 30) The algorithm is shown in Algorithm 2.

Algorithm 2: Time driven simulation

```

Input: simEnd, simTime=0,  $\Delta t$  , initial variables
while simTime  $\leq$  simEnd do
    |   simTime = simTime+ $\Delta t$ ;
    |   Calculate system variables and system state;
end

```

For the present thesis activity scanning is used. It can be classified as a specific approach of the time driven simulation. The following paragraph will give an overview over the process.

Activity scanning

Activity scanning is a state based approach for simulation modelling. Figure 2.14 shows the process. At the beginning all variables are initialized and the simulation clock is set to the start time. Afterwards the time flow mechanism (TFM) starts. This is a strategy in the course of which the simulation time is incremented by a fixed Δt . This process can be compared to the time driven simulation (figure 2.13) by Hedtstück. An appropriate size of Δt is essential for the output of the simulation. On the one hand a too small Δt may lead to an inefficient computing time, on the other hand a too high Δt could lead to an invalid output. After the increase of the simulation time an activity scan is performed. This scan consists of two major parts. The condition is a logical requirement which has to be tested if it is true or false. If the condition is fulfilled, the action will be performed. An activity is defined as an action which changes the state of the system. It might happen that one action would cause the satisfaction of the condition of earlier scanned activity. This leads to another scan of the system. Scanning has to be carried out until none of the conditions is satisfied at the current time. The next step is to ask if the clock has reached the end of the simulation or if the scanning should continue. It is important to prioritize the activities to decrease the computation time. (Balci, 1988)

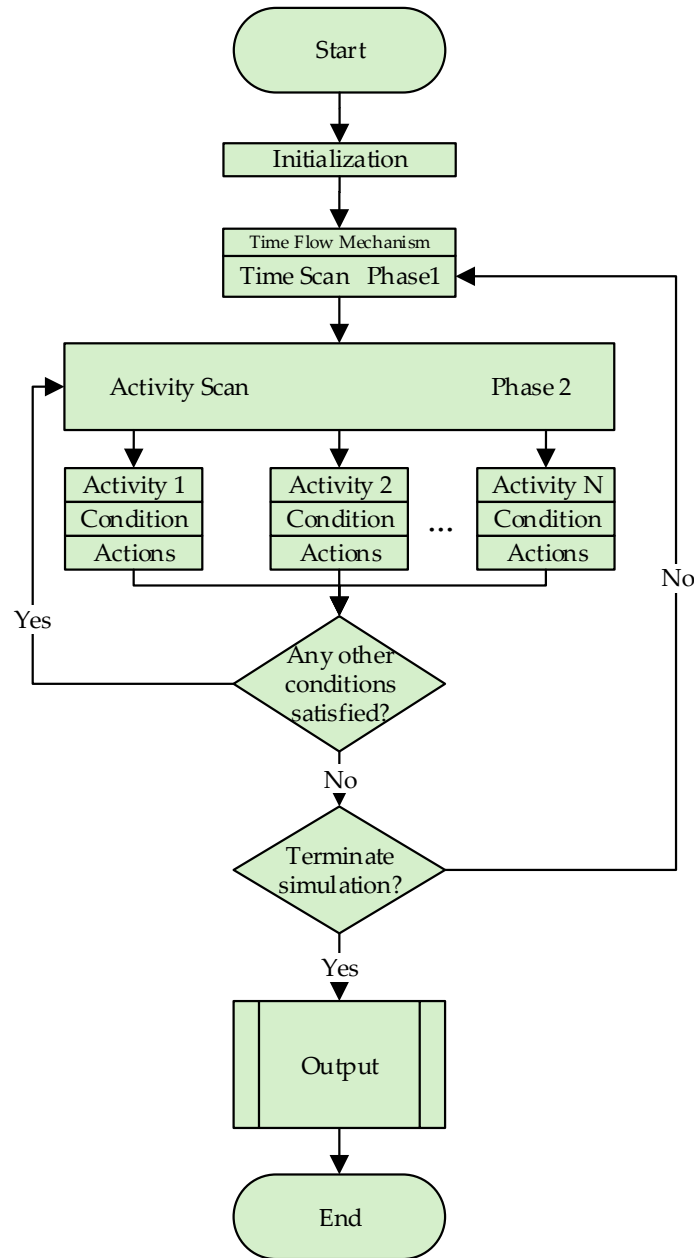


Figure 2.14: Activity scanning adapted from Balci (1988)

The disadvantages evoked through the fixed incremental time steps are the higher computation time and recurring testing of conditions. According to Balci (1988) advantages are the modularity of the program, the maintainability, easiness of modifying, easiness of implementation and easiness of understanding.

2.5 Simulation in maintenance and manufacturing

According to Alrabghi and Tiwari (2015), the field of optimization maintenance in manufacturing through simulation has been receiving more attention. The problems solved in the researches can be divided into several sections 2.15. In their research Alrabghi and Tiwari (2015) found out, that primarily discrete event simulation techniques are used. The articles focusses on single machine optimization compared to merely less investigations in multi component systems. The main maintenance strategy, which was conducted in the last studies, was the strategy of preventive maintenance (70%). The reasons for the simulations also vary widely. As a result, the degree of abstraction and consequently the level of detail also differs a lot along the investigated simulation studies. The topics have been chosen from the areas of perfect strategy determination, spare part management, maintenance cycle detection and maintenance scheduling on machine and line level. The main objective of the most articles is to minimize the costs, whereas they do not only comprise the maintenance costs but also the costs for downtimes, for defective products as well as spare part management costs. Another objective was to increase the availability instead of costs. A higher availability does not necessarily lead to higher throughput, therefore Alrabghi and Tiwari suggest considering the whole manufacturing system. As decision variables most papers selected PM frequency, but also maintenance thresholds, spare parts and buffer sizes are used. Constraints applied in the papers are the maximum stock level, the maximum budget or specified preventive windows where PM actions have to be taken for each machine.

	Maintenance strategy				General maintenance			Joint optimization		
	PM		CM					Spare parts		Production
Decision variables	PM frequency	Maintenance schedule	Inspection frequency	Maintenance threshold	Technicians	Equipment	Maintenance priorities	Recorder level	Recorder level	Buffer size
								Maximum stock level	Order quantity	
Objectives	Min cost, max availability, max throughput									

Figure 2.15: Optimal problem formulation for different types of maintenance optimization problems adapted from Alrabghi and Tiwari (2015)

2.6 Maintenance opportunity window calculation and simulation

The papers discussed in the following paragraphs deal with the problem of the creation of maintenance time windows in a production line. Commonly, simulation based solutions are used to solve this problem. These time windows are often defined as opportunity windows in literature.

Chang et al. discovered possibilities to perform maintenance tasks on a flow shop during production times by strategically shutting down machines for short periods. The goal of the research was to find out these possibilities using real-time information. A simulation based algorithm was developed for optimization. At the end a throughput impact calculation of different maintenance opportunity windows was done. The difference of these windows was the safety margin of the buffer level (complete empty, 75% empty, 50% empty). The model is based on a machine-buffer-machine system with a specific material flow. This abstraction will be applied and discussed in further detail in chapter 3. (Chang et al., 2007)

Gu et al. used the maintenance opportunity windows model from Chang et al. and implemented it on various line configurations. The simulations were further used to deal with uncertainties in the production line such as blockages, starvation and machine failures. The authors created a solution for serial and parallel production lines. Finally, the algorithm was tested on an automotive assembly plant. (Gu et al., 2013)

Lee et al. have established an analytical approach to compute stochastic maintenance opportunity windows for an unreliable two-machine-one-buffer-system. They used discrete and continuous time Markov models to solve the problem. They did not set buffer limits to zero in order to get buffer reserves for unexpected failures in the production line. Chang et al. and Gu et al. suggested to use simulation models to handle uncertainties in the calculation of time windows. In the end the obtained stochastic windows were validated with a simulation experiment. (Lee et al., 2013)

Neubacher et al. (2016) use the hierarchical conceptual control model to generate a generic model for a flexible production system. The main objective of this paper is to illustrate the impact of downtimes on the throughput. The advantage of the hierarchical control structure is the flexibility of modeling, though the computational effort is still acceptable. The result are estimated productivity losses due to downtimes to enhance preventive maintenance scheduling.

Chapter 3

Model Development

After the previous theoretical chapters the next chapter discusses the model development of the problem. This chapter is introduced by a detailed problem description. Subsequently the basic concept of the solution strategy and the correlating theoretical classification are presented. The third section deals with the assumptions on the algorithm and the model which are primarily company driven. The chapter closes with a mathematical description of the problem.

3.1 Problem Description

As stated in the introduction, nowadays a lot of companies in the field of production have implemented TPM. As already explained in chapter 2.1.4, TPM involves employees on the production level in the execution of maintenance tasks. Therefore, daily, weekly or monthly tasks have to be carried out. The current maintenance practice is to perform these tasks during a non-production shift. If this is not possible, usually one shift per week is used for executing these tasks. Therefore, the production time shortens and the weekly throughput decreases. At the OEE (2.1.2, page 10) calculation this is often hidden by not considering planned downtimes. This throughput reduction can be prevented by an optimized scheduling of maintenance tasks. In almost every production line the operation steps comply with different cycle times. Inbetween these procedures different buffer exists, in most cases this are simple conveyors or even fixed storage systems such as paternoster warehouses. Due to the time differences these buffer will be drained or filled. Over time the buffer will become empty or full and at this moment the machines next to it will starve or block. These buffers are hidden maintenance opportunities for

short term tasks, without influencing the throughput on the bottleneck machine. The second possibility is that a machine breaks down. After some time the upstream machines start to block and the downstream machines start to starve. This could also be labelled as a maintenance possibility. Both occurrences are triggers for flexible maintenance windows and illustrated in figure 3.1. Thereby the top illustration shows the first case of a different cycle time with a full buffer B_1 inbetween. The first machine could be turned off without influencing machine2, for a time of the cycle time of machine2 multiplied with the buffer level. The bottom image shows machine one breaking down. Machine2 could be turned off after the buffer one is empty. Otherwise the machine would starve from this point in time.

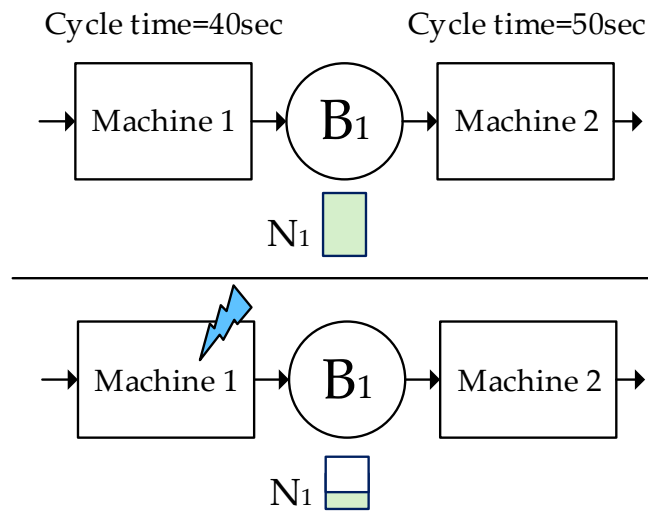


Figure 3.1: Problem definition

The next question that arises is how to schedule these daily, weekly or monthly tasks on the flexible window, since scheduling has not been necessary before, due to fixed maintenance shifts. Thus every employee has known exactly when to carry out which maintenance task. The aforementioned problem defines the second task of the present thesis, i.e. optimizing the scheduling of the maintenance tasks in order to utilize the occurring windows. To summarize, the two main tasks are:

- Determination of opportunity windows for flexible maintenance in the production line with the associated duration and starting time
- Optimized scheduling of the maintenance tasks in order to utilize the occurring windows

The following paragraph establishes a link to chapter 2.1, in which literature about maintenance is discussed. Of all the defined objectives of maintenance (page 7) this thesis can be assigned to the goal of "Development of a cost optimal maintenance plan on area level". The algorithm serves as the basis for the next objective, a system optimized maintenance plan. The optimized determination of maintenance times and an optimal coordination strategy for the system are part of another project (figure 3.2) and are assumed to be handed in with the task and the employee list. From a maintenance point of view the algorithm works with all tasks which have to be scheduled prior. This means the algorithm supports the strategies of preventive and predictive maintenance. This thesis mainly contributes to optimize TPM and thus focuses on the first three pillars in figure 2.5. The "Autonomous Maintenance" is still preserved because all employees on the machine level are still involved in carrying out the task. The second pillar, "Focused Maintenance", maintains the function of eliminating the seven types of waste. The algorithm should eliminate unnecessary downtimes for preventive maintenance and thus cut waste. The third pillar, "Planned Maintenance", should be put into practice by the second part, the task scheduler.

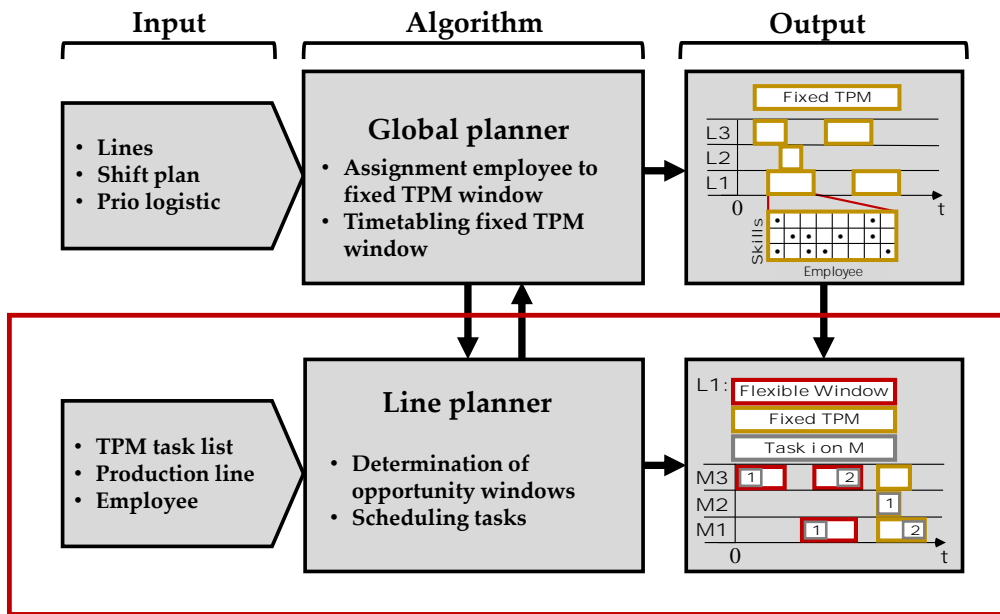


Figure 3.2: Project

As stated in the introduction, this thesis has been developed in the course of a project with an industrial partner. Therefore, the requirements on the algorithm as well as the problem definition were tailor-made for the company's needs. Two master theses dealing with the planning of maintenance tasks and employees emerge from this project. Figure 3.2 shows the connection to the second planning tool. The maintenance department of

the company is structured into a de-central part on the production line and a central part on company level. The de-central organization involves maintenance engineers and some workers on the machine. They mainly carry out TPM-tasks. The central department has been established for difficult tasks and critical breakdowns of machines. The major tasks of both thesis are shown in figure 3.2 (this thesis framed in red). Furthermore, the structure, input data, interfaces and the output of the planning tool are illustrated. Interfaces to the global planner are the fixed TPM windows of the production line as shown on the output side.

3.2 Concept

The basic concept of the algorithm is illustrated in figure 3.3. The two tasks described in the previous section are also split at the algorithm into separate program parts. As illustrated, the first step is to determine the opportunities for flexible maintenance windows. The three input variables are the cycle times, the buffers and the structure of the line. The maximum capacity and the start level from the buffer are needed. First of all, the structure of the line provides information whether machines run parallel or not. Secondly, information about the sequence of working operations is given. The output of this part of the algorithm can be seen on the right-hand-side of figure 3.3, illustrated as a Gantt chart.

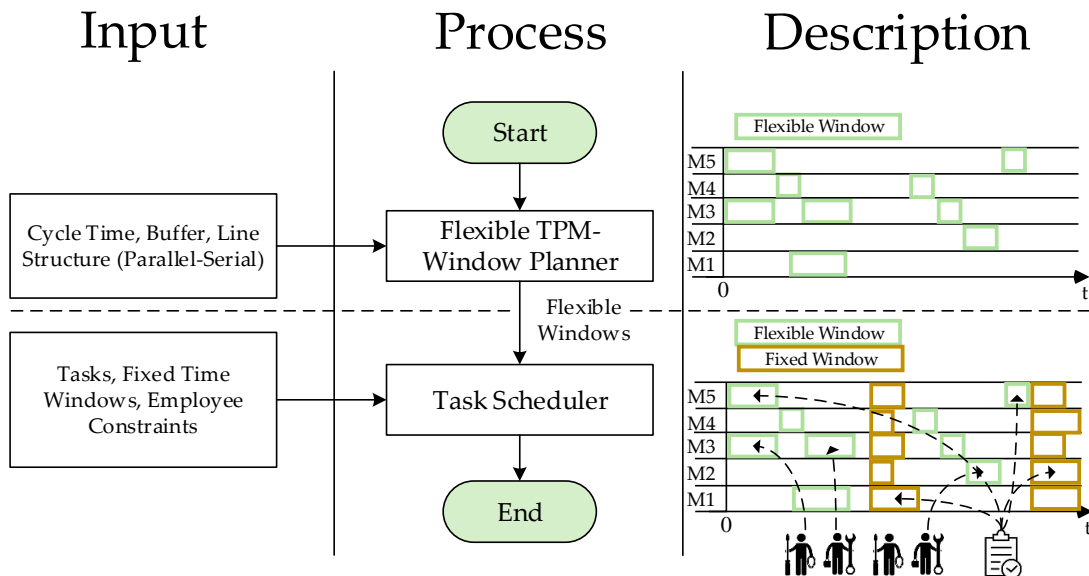


Figure 3.3: Concept explanation

At the end, every machine has a list of calculated opportunity windows with corresponding starting and end times. The next step is that the information is forwarded to the task scheduler. Other input sources for this scheduler are a task list for every machine, the fixed TPM windows from the global planner and the resource constraints like employees. As illustrated in figure 3.3, at the bottom right, a scheduler should plan the tasks and schedule the employees within the given time windows. The overall output is a list of tasks with assigned employees on specific time windows.

The methods used to solve the problem are also divided into the two program parts. For the opportunity time window planner, a simulation based algorithm using the method of activity scanning (Chapter 2.4.2) is developed. The simulation process used for this thesis is based on Banks et al. (2004) and on Robinson (2013), as shown in figure 2.10 (page 31) and 2.9 (page 28). The first step of the process, defining the problem and describing the system, was done in the previous section. The following part of this chapter explains the steps of the model conceptualization. Chapter 4.1 includes the steps of the model translation according to Banks et al. (2004). Chapter 4 also focuses on the model design of Robinson (2013). In order to validate the performance of the algorithm, a simulation model was implemented in *Plant Simulation*®¹. The developed algorithm was subsequently tested in this virtual environment. Thereby, this validation procedure concludes the practical part of this thesis. In order to create an optimized kind of task scheduling, the first step is to set up a linear integer model according to chapter 2.2.3. Afterwards the model is solved with the optimization software *Gurobi*².

The concept and algorithm can be classified according to the following areas (Van Horenbeek et al., 2010):

- **Output** - Maintenance scheduling/planning
- **Data sources** - Operating data, cost data
- **Maintenance actions** - Preventive maintenance
- **Maintenance policies** - Opportunity-based maintenance
- **Maintenance concepts** - TPM
- **Optimization algorithm** - Integer programming
- **Maintenance optimization criteria** - Maintenance cost, output quantity
- **System configuration** - Multi-unit, parallel/series, redundancy
- **General modeling techniques** - Discrete, deterministic, system perspective (Line), single objective, constrained, finite planning horizon

¹https://www.plm.automation.siemens.com/en_gb/

²Optimization software to solve all major problem types <http://www.gurobi.com/>

The following section discusses the assumption which have been made for the modeling step.

3.3 Assumptions and Simplifications

According to Robinson (2013), the abstraction of a real world problem to a conceptual model requires different assumptions and simplifications (see figure 2.9). This section shows the assumptions which have been made for the employees, the production line structure and the task assignments.

The **employees** are considered to have a fixed presence threshold for every time window. This threshold is calculated with the shift plan and availability of the employee for the machine. Due to the fact that TPM maintenance tasks are also carried out from the worker on the machine, the production employee-machine assignment has to be considered. Another assumption is that not every employee can carry out every task. Special skills are required. Figure 3.4 illustrates these assumptions. The two employees on the left-hand side are maintenance workers on line level. They are available for every machine on the production line, but worker one is not available for time window 2, e.g. Therefore the threshold is zero. The two employees on the right-hand side, however, are workers at the production. If employee 3 worked on machine 1, he would not be available for window 1 and 2.

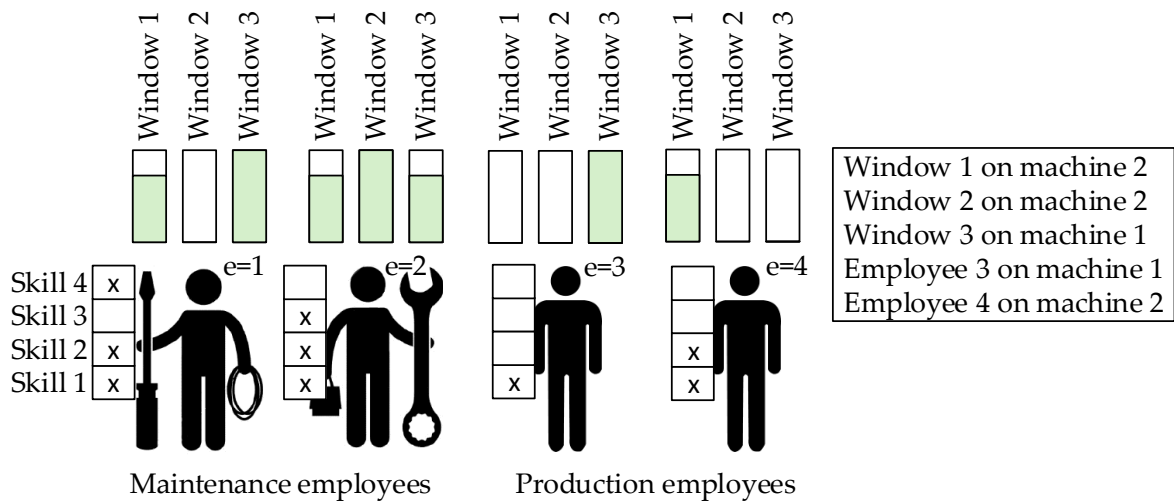


Figure 3.4: Employee assumptions

The **production line** is assumed to be structured as illustrated in figure 3.5. There is always a buffer between two machines. If no buffer exists, a buffer with a capacity set to zero is modeled. Therefore, every production line consists of m machines and m buffer as it also starts with a buffer. The reliability of the machines is assumed to be 100%. Quality and performance rates are not considered for the production output in the modeling process and therefore are set to 100%. For the bottleneck calculation the cycle times are used. A production line can also have redundant machines, illustrated in figure 3.6. These redundant machines carry out the same working steps. Split production line with parallel line sections are not considered.

Since the tasks are mainly planned TPM tasks, which have a daily, weekly or monthly cycle, the first **task** assumption is that spare parts are not considered for planning and therefore the spare part availability is set to 100%. The second assumption is that the tasks can also be classified according to skills required by the employees. Durations and start times of tasks are assumed to be deterministic, and deviations of the optimal maintenance time are modelled with a penalty cost function.

3.4 Mathematical model

This sub-chapter presents the mathematical model of the opportunity time window planner as well as the linear integer model for the task scheduler. Both formulations are based on the previous sections of the problem definition, concept and assumptions.

3.4.1 Time window planner

Figure 3.5 shows a way to abstract a production process. The investigated system contains a buffer B_2 with a buffer-level of N_2 and a maximum buffer capacity of C_2 . Prior to the buffer is the machine M_1 with the specific production rate of m_1 . The machine M_2 with the production rate of m_2 follows. All the following abstractions and modelling steps are based on this generic formulation. This sub-chapter will start with the mathematical formulation for a single machine-buffer-single machine system. Afterwards a redundant system follows. Through these two basic formulations the whole system of a production line with its different line formations is modelled.

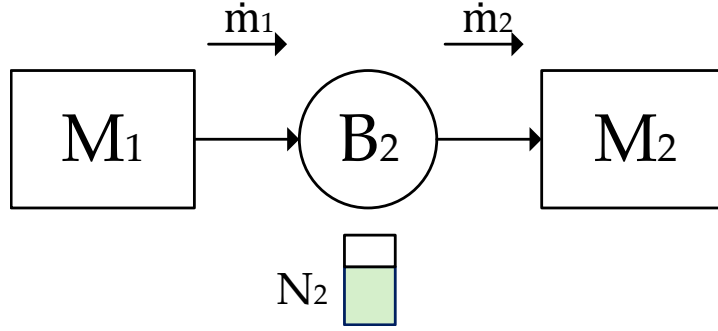


Figure 3.5: Single machine-buffer-single machine system

Notations

Machine variables

$i = 1, \dots, n$ Machine

$CT_i(t)$ Cycle time of machine/machine combination i at time t

CT_{start_i} Cycle time of machine/machine combination i if all machines of i are producing

CT_{high_i} Cycle time of machine combination i if one machine of i is turned off

$\dot{m}_i(t)$ Production rate of machine i $\dot{m}_i(t) = 1/CT_i(t)$

Buffer variables

N_i Level of Buffer i

C_i Capacity of buffer i

Time window variables

$k = 1, \dots, K$ Time windows

Δt_k Duration of time window

tws_k Start time of time window k

twe_k End time of time window k

Single machine

The following equation represents the buffer change \dot{N}_i of buffer i based on the model of figure 3.5. This abstraction is inspired by the research of Chang et al., 2007.

$$\dot{N}_i = \dot{m}_{i-1} - \dot{m}_i \quad (3.1)$$

$$\int_0^T \dot{N} dx = \int_0^T (\dot{m}_{i-1} - \dot{m}_i) dx \quad (3.2)$$

$$N_i(T) - N_i(0) = T(\dot{m}_{i-1} - \dot{m}_i) \quad (3.3)$$

$$T = \frac{N_i(T) - N_i(0)}{(\dot{m}_{i-1} - \dot{m}_i)} \quad (3.4)$$

This equation is the basic formula for the window planner. T defines the time a buffer level $N_i(T)$ has been reached. $N_i(T)$ is known for the calculation. Either the buffer runs at full capacity or it falls down to zero capacity depending on which machine is switched off.

Redundant machine combination

Another possible scenario is that a machine is redundant with other ones. In this case the material flow of the redundant machine combination is not zero if one machine does not work.

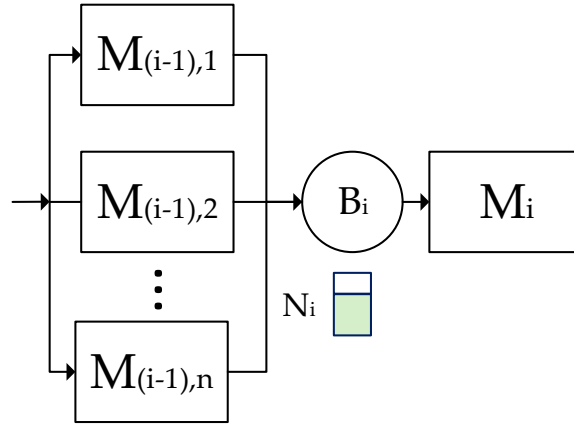


Figure 3.6: Redundant-machine-system

$$\dot{m}_{i-1} = \dot{m}_{(i-1),1} + \dot{m}_{(i-1),2} + \dots + \dot{m}_{(i-1),n} \quad (3.5)$$

$$\dot{N} = \dot{m}_{(i-1),1} + \dot{m}_{(i-1),2} + \dots + \dot{m}_{(i-1),n} - \dot{m}_i \quad (3.6)$$

$$\int_0^T \dot{N} dx = \int_0^T (\dot{m}_{11} + \dot{m}_{12} + \dot{m}_{13} - \dot{m}_2) dx \quad (3.7)$$

$$T = \frac{N_i(T) - N_i(0)}{(\dot{m}_{(i-1),1} + \dot{m}_{(i-1),2} + \dots + \dot{m}_{(i-1),n} - \dot{m}_i)} \quad (3.8)$$

Type of time window

Figure 3.7 shows the two basic cases how a time window can be created without having to deal with a breakdown. The image on the left-hand side shows the possibility of a time-window on machine M_{i-1} because the buffer next to it is full. If no time window would be created, this machine would switch to the cycle time of M_i , if the following cycle time is higher than the previous one. $\rightarrow \dot{m}_{i-1} = \dot{m}_i$, Unnecessary waiting times would be created and the buffer would block. On the right-hand side the exact opposite

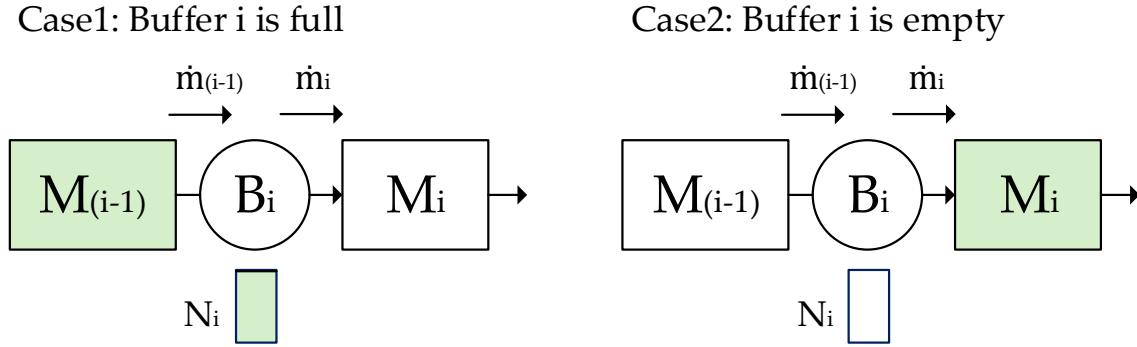


Figure 3.7: Comparison full to empty buffer

is shown: The buffer B_i is not set to the maximum level but it is empty. This means machine M_i produces with a higher rate than machine M_{i-1} . If no window was created the machine M_i , would encounter unnecessary waiting times and the buffer would starve. In the following sections these two cases will be called the type of time window. Case one is a the type of a "Draining window", case two is a "Filling window".

To calculate the possible window-duration, the equation from chapter 3.4.1 is used.

$$T = \frac{N_i(T) - N_i(0)}{\dot{m}_{i-1} - \dot{m}_i}$$

Through dynamic line changes and the time driven simulation approach a maximal utilization of these time windows could lead to a throughput minimization. To counteract this scenario, a utilization degree μ of the buffer is introduced. This percentage represents how full or empty a buffer after a time window can get. For the of a draining window, this leads to a time window of:

$$\dot{m}_{i-1} = 0; N_i(T) = C_i * (1 - \mu); N_i(0) = C_i$$

$$T = \frac{C_i(1 - \mu) - C_i}{-\dot{m}_i} \rightarrow T = \frac{C_i\mu}{\dot{m}_i}$$

For the case of a filling window:

$$\dot{m}_i = 0; N_i(T) = C_i\mu; N_i(0) = 0$$

$$T = \frac{C_i\mu}{\dot{m}_{i-1}}$$

Possible structure of machine-buffer-machine system

The last section points out the differences of the time windows regarding the buffer. The following paragraphs will illustrate the different line formations and their possible starting cycle times and their changes over time. Figure 3.8 shows the four possible cases.

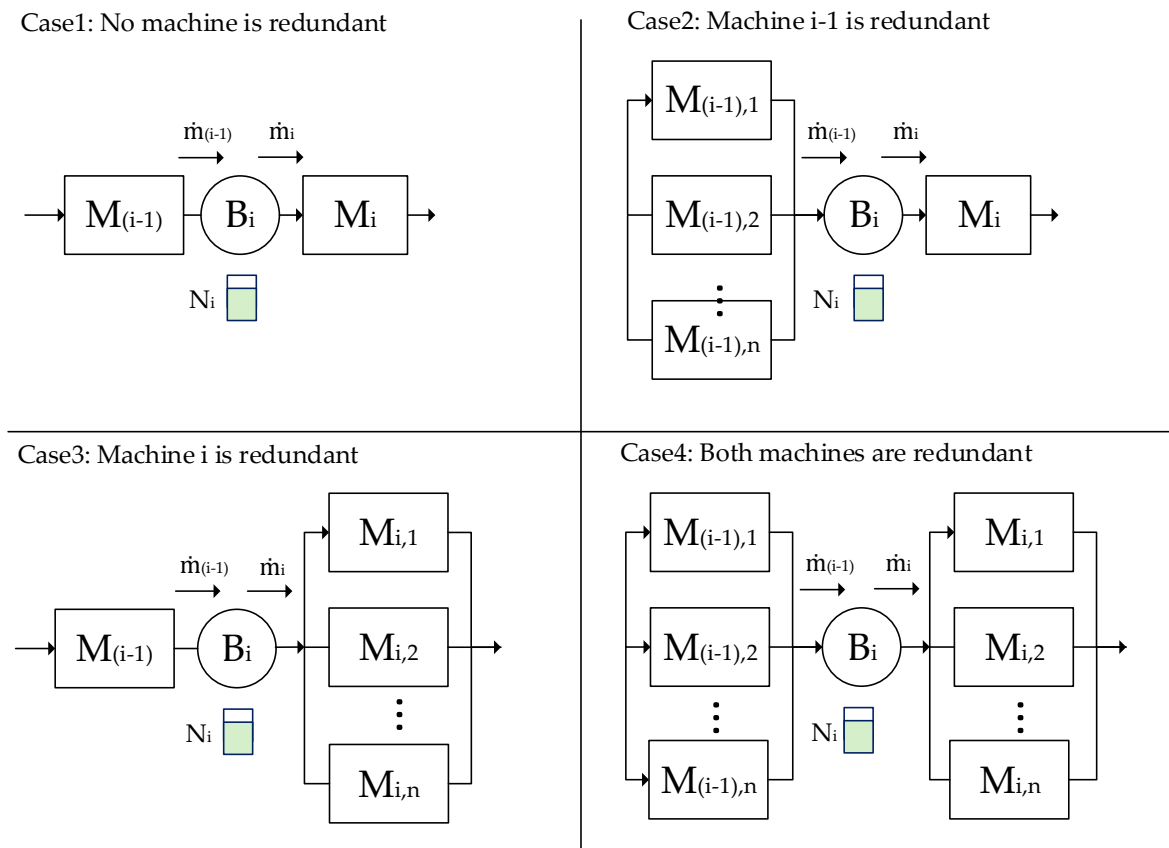


Figure 3.8: Cases of machine-buffer-machine structure

With the equation 3.4 the duration of the time a buffer will starve or block can also be calculated.

$$N_i(T) = \begin{cases} C_i\mu & \text{if } \dot{m}_{i-1} > \dot{m}_i \\ C_i(1 - \mu) & \text{if } \dot{m}_{i-1} < \dot{m}_i \end{cases}$$

The different starting cycle times and the cycle time changes in the flexible windows depend on the structure of the line.

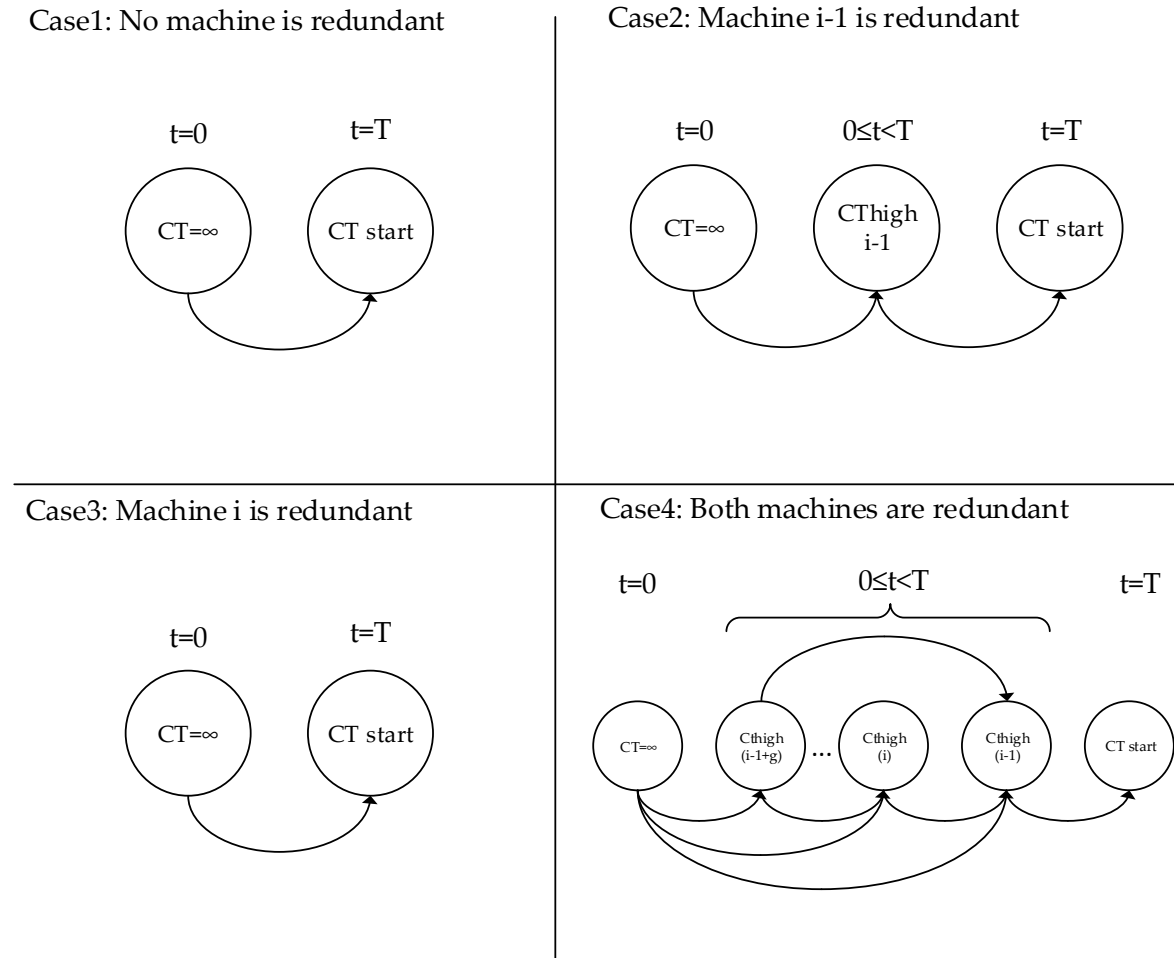


Figure 3.9: State changes of a draining window depending on the structure of the formation

Figure 3.9 illustrates the possible cycle time changes of the machine depending on the structure of the machine-buffer-machine system of figure 3.8. The following cases and definitions have been set up for a draining window. Given the fact that there was a filling window, case two and three of figure 3.9 would have to be exchanged and $i - 1$ would have to be presumed for every case.

Case1 no machine is redundant: At this case two cycle time changes occur. At the beginning of the time window the cycle time of i or $i - 1$, depending on case one or two of figure 3.7, is set to infinity. There is only one cycle time change. At the end of the time window, $CT = \infty$ is set back to the start cycle time of the machine.

$$CT_{i-1}(0) = \infty$$

Case2 machine $i - 1$ is redundant: In case two the time window of the redundant machines $i - 1$ can start with a cycle time of infinity for all redundant machines if $CT_i(0) = \infty$ or with $CT_{i-1}(0) = CThigh_{i-1}$ in the case of $CT_i(0) = CTstart_i$. If the time window starts with $CT = \infty$, the cycle time has to change to $CThigh_{i-1}$ before it is set back to $CTstart_{i-1}$ at the end of the time window.

$$CT_{i-1}(0) = \begin{cases} \infty & \text{if } CT_i(0) = \infty \\ CThigh_{i-1} & \text{if } CT_i(0) = CTstart_i \end{cases}$$

Case3 machine i is redundant: Regarding the cycle time changes, this case is similar to case one. Due to the fact that the machine which is turned off for maintenance is a single machine, it starts with $CT_{i-1}(0) = \infty$. The interesting aspect of this case is the calculation of the time window duration T which will be discussed on page 55.

$$CT_{i-1}(0) = \infty$$

Case4 both machines are redundant: This case is the most complex version for calculation of the time window duration and the cycle time changes. As shown in figure 3.9 the possible starting times of a time window depend on the number of redundant machine combinations subsequent to it and their actual cycle times. A detailed explanation of this case can be found in Appendix A. The case of a filling buffer window is shown in figure 3.10.

$$CT_{i-1}(0) = \begin{cases} \infty & \text{if } CT_i(0) = \infty \\ CThigh_{i-1} & \text{if } CThigh_{i-1} > CT_i(0) \\ CT_i(0) & \text{if } CThigh_{i-1} < CT_i(0) \end{cases}$$

Figure 3.11 shows the case in which machine $i - 1$ starts with the cycle time $CThigh_{i+2}$ of machine $i + 2$, because all machines in-between are in a maintenance window. This is indicated by the full buffer. All machines from $i \rightarrow i + 2$ produce with a cycle time of $CThigh_{i+2}$. The condition which has to be fulfilled for such constellations is

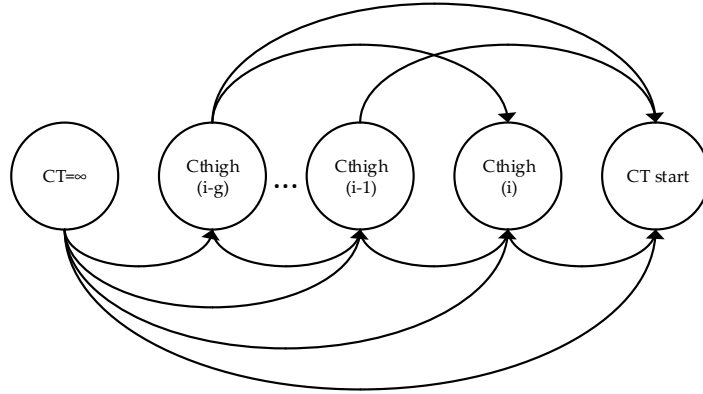


Figure 3.10: State changes Case4 for a filling window

$CThigh_{i+2} > CThigh_{i+1} > CThigh_i > CThigh_{i-1}$. In this case g is 3. Afterwards the cycle time of machine $i - 1$ will take on every $CThigh$ of the upcoming machines. Therefore CT changes five times in this cycle.

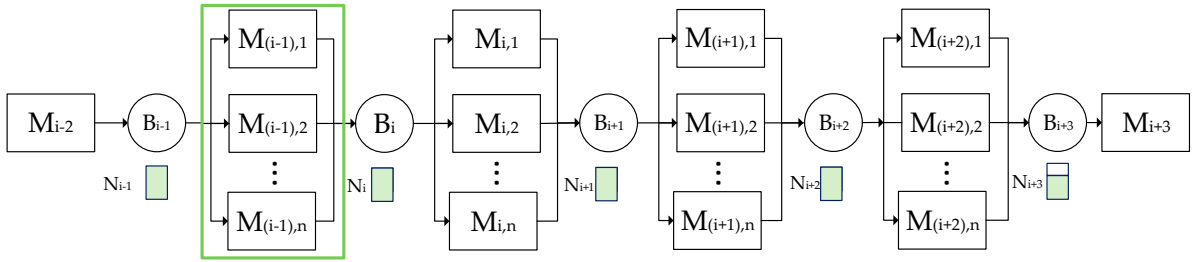


Figure 3.11: Redundant production line

Time window duration calculation

In this section the calculation for the time window duration of the stated cases will be discussed. These durations primarily depend on the discussed cycle time changes of the other machines. The calculation links the possible structure of the machine-buffer-machine system with the type of time window. In the subsequent parts, cycle time changes of the machines will be abbreviated with twc_n, c , whereby n indicates the machine and c the number of time changes with the type of a specific time in point. The end of a time window will be abbreviated by twe . Start times of the time window from the other machines are not needed because this point in time lies in the past. The final factor which

is needed is the actual time which is abbreviated with t_{act} . The calculated duration of the time window is indicated by Δt_i . The production rate \dot{m}_{low} of a redundant machine is the rate if one machine of this redundant system is shut down ($\dot{m}_{low} = \frac{1}{CT_{high}}$). The used abbreviations for the calculation are:

C_i	Capacity of buffer i
N_i	Buffer level of i
P_i	Parts available for time window calculation
μ	Utilization of buffer-capacity
δt	Time until products are produced
$\dot{m}_{i-1}(t)$	Time dependent function for the production rate from machine $i - 1$
$\dot{m}_i(t)$	Time dependent function for the production rate from machine i
c	Number of cycle time changes
v	Indices of interval $v = 1, \dots, c_{i-1}$ of machine $i - 1$
w	Indices of interval $w = 1, \dots, c_i$ of machine i
$\dot{m}_{w,i}$	Cycle time of interval w on machine i
$l(I_w)$	Length of interval $l(I_w) = t_{w-1} - t_w$
I_w	Interval $I_w = (t_{w-1}, t_w)$

The standard material flow equation 3.3 by Chang et al. (2007) considers a constant production rate in the time of an opportunity window. Due to the fact that the cycle times can change within the time windows at redundant machines, this equation has to be generalized. The change of the buffer level within the time window is abbreviated with P_i . An example for the change of the cycle time between a time window can be seen in figure 3.12.

$$P_i = C_i \mu - N_i$$

$$P_i = \int_0^T \dot{m}_{i-1}(t) - \dot{m}_i(t) dt = \sum_{v=1}^{c_{i-1}} \dot{m}_{v,(i-1)} l(I_v) - \sum_{w=1}^{c_i} \dot{m}_{w,i} l(I_w) \quad (3.9)$$

The equation 3.9 is the basic formula for all following calculations. The cases are, again, based on the system of machine-buffer-machine formations in figure 3.8. The equations are separately stated for filling and draining windows. Furthermore, the cycle time changes within the time window are indicated with $twc_{i,h}$. h stands for the number of time window changes.

Case1 The first case is a draining buffer with a single machine-buffer-single machine structure. The time window is calculated for the machine $i - 1$.

Draining buffer:

$$\Delta t_{i-1} = \begin{cases} \frac{P_i}{\dot{m}_i} & \text{if } \dot{m}_i > 0 \\ (twe_i - t_{act}) + \frac{P_i}{\dot{m}_i} & \text{if } \dot{m}_i = 0 \end{cases}$$

Filling buffer:

$$\Delta t_i = \begin{cases} \frac{P_i}{\dot{m}_{i-1}} & \text{if } \dot{m}_{i-1} > 0 \\ (twe_{i-1} - t_{act}) + \frac{P_i}{\dot{m}_{i-1}} & \text{if } \dot{m}_{i-1} = 0 \end{cases}$$

Case2 Machine $i - 1$ is redundant. One cycle time change occurs at the draining window.

Draining buffer:

$$\Delta t_{i-1} = \begin{cases} \frac{-P_i}{\dot{m}_{low(i-1)} - \dot{m}_i} & \text{if } \dot{m}_i > 0 \\ (twe_i - t_{act}) + \frac{-P_i}{\dot{m}_{low(i-1)} - \dot{m}_i} & \text{if } \dot{m}_i = 0 \end{cases}$$

$$twc_{(i-1),1} = twe_i$$

Filling buffer:

$$\Delta t_i = \begin{cases} \frac{P_i}{\dot{m}_{i-1}} & \text{if } i - 1 \text{ has } \dot{m}_{start(i-1)} \\ \text{equation 3.10} & \text{else} \end{cases}$$

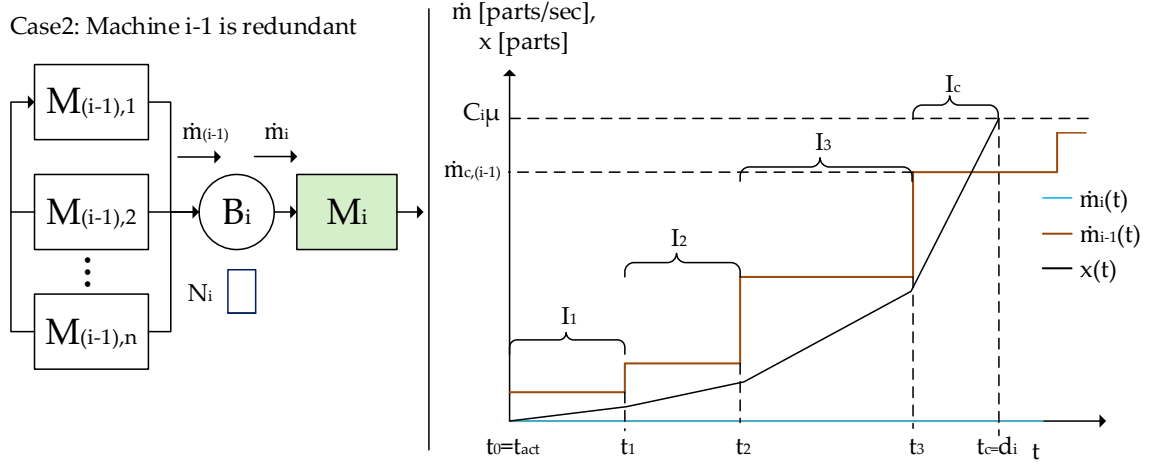
$$P_i = \sum_{v=1}^{c_{i-1}} \dot{m}_{v,(i-1)} l(I_v) = \sum_{v=1}^{c_{(i-1)}-1} \dot{m}_{v,(i-1)} l(I_v) + (T - t_3) \dot{m}_{3,(i-1)}$$

The general equation is:

$$T = \frac{P_i - \sum_{v=1}^{c_{(i-1)}-1} \dot{m}_{v,(i-1)} l(I_v)}{\dot{m}_{c,(i-1)}} - t_{c-1} \quad (3.10)$$

With $v = c_{i-1}$ at $N_i > C_i \mu$

Figure 3.12 shows Case2 with a filling buffer. As illustrated on the left hand side, the time window is calculated for machine M_i . At the point in time t_{act} both machines have a cycle time of ∞ and therefore the material flow is zero. Due to that machine i is a single machine the material flow is for the whole time window zero, illustrated with the


 Figure 3.12: Calculation of d_i in Case2 with filling buffer

blue line. The cycle time of the redundant machine combination prior to machine M_i changes over time. At the interval of I_c the maximal buffer capacity is reached.

Case3 Machine i is redundant. One cycle time change occurs at the filling window.
Draining buffer:

$$\Delta t_{i-1} = \begin{cases} \frac{P_i}{\dot{m}_i} & \text{if } i \text{ has } \dot{m}_{start(i)} \\ \text{equation 3.11} & \text{else} \end{cases}$$

$$-P_i = -\sum_{w=1}^{c_i} \dot{m}_{w,(i)} l(I_w) = -\left(\sum_{w=1}^{c_i-1} \dot{m}_{w,(i)} l(I_w) + (T - t_{c-1}) \dot{m}_{c,(i)}\right)$$

$$T = \frac{P_i - \sum_{w=1}^{c_i-1} \dot{m}_{w,i} l(I_w)}{\dot{m}_{c,i}} - t_{c-1} \quad (3.11)$$

With $w = c_i$ at $N_i > C_i(1 - \mu)$

Filling buffer:

$$\Delta t_i = \begin{cases} \frac{P_i}{\dot{m}_{i-1} - \dot{m}_{low(i)}} & \text{if } \dot{m}_i > 0 \\ (t_{we_{i-1}} - t_{act}) + \frac{C_i \mu}{\dot{m}_{i-1} - \dot{m}_{low(i)}} & \text{if } \dot{m}_i = 0 \end{cases}$$

$$t_{wc_{i,1}} = t_{we_{i-1}}$$

Case4

Both machines are redundant:

$$\Delta t_{i-1} = \begin{cases} \frac{-P_i}{\dot{m}_{low(i-1)} - \dot{m}_i} & \text{if } i \text{ has } \dot{m}_{start(i)} \\ \text{equation 3.12} & \text{else} \end{cases}$$

$$P_i = \int_0^T \dot{m}_{i-1}(t) - \dot{m}_i(t) dt = \sum_{v=1}^{c_{i-1}} \dot{m}_{v,(i-1)} l(I_v) - \sum_{w=1}^{c_i} \dot{m}_{w,i} l(I_w)$$

$$P_i = \sum_{v=1}^{c_{(i-1)}-1} \dot{m}_{v,(i-1)} l(I_v) + \dot{m}_{c,(i-1)} * (T - t_{c_{(i-1)}-1}) - \sum_{w=1}^{c_i-1} \dot{m}_{w,i} l(I_w) - \dot{m}_{c,i} * (T - t_{c_i-1})$$

$$T = \frac{P_i - \sum_{v=1}^{c_{(i-1)}-1} \dot{m}_{v,(i-1)} l(I_v) + \dot{m}_{c,(i-1)} t_{c_{(i-1)}-1} + \sum_{w=1}^{c_i-1} \dot{m}_{w,i} l(I_w) - \dot{m}_{c,i} t_{c_i-1}}{\dot{m}_{c,(i-1)} - \dot{m}_{c,i}} \quad (3.12)$$

The equation 3.12 is illustrated in the diagram in figure 3.13. In a production line where many redundant machines are linked in a row the possible cycle times within a time window depend on the number of redundant machines previous or subsequent to the machine. In figure 3.13 the time window is calculated for the machine i . As shown, the buffer level is about 45% of the capacity. The available parts in this time window are $P_i = C_i \mu - N_i(0)$. The blue line shows the production rate of the machine i over the time until the end of the time window t_c . As also illustrated the machine starts with the same production rate as machine $i - 1$ with $CThigh_{i-3}$. This means that also the redundant machine combinations $i - 3$ and $i - 2$, which are not shown in figure 3.13, are in a maintenance time window. At the point in time t_1 the machine $i - 3$ switches to $CT_{i-3}(t_1) = CT_{start_{i-3}}$ and the cycle times of the remaining machines switch to $CThigh_{i-2}$. Afterwards the buffer prior to machine $i - 2$ runs full and at the point in time t_2 all machines of $i - 2$ work again. Due to the higher cycle time of machine i compared to machine $i - 1$ the cycle time changes to $CThigh_{i-1}$ for machine $i - 1$ and to $CThigh_i$ for machine i . Now the buffer level between the two machines starts to increase due to the production rate differences. At the point in time t_3 the gap between the production rates get larger, because the time window of machine $i - 1$ ends and the cycle time changes to $CT_{i-1}(t_3) = CT_{start_{i-1}}$. After the time d_i the buffer of B_i reached the level $C_i \mu$ and the machine $M_{i,1}$ starts to work again and the time window ends.

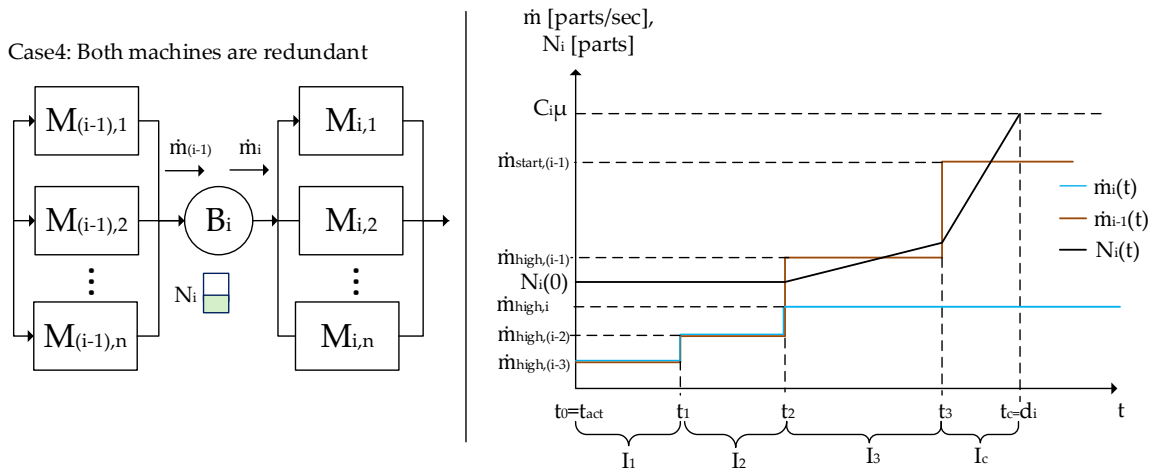


Figure 3.13: Calculation of d_i in Case4 with filling buffer

Influence of the bottleneck on the time window calculation

In the following paragraph the influence of the bottleneck and the line structure will be discussed. In figure 3.14 the detection of the bottleneck is shown. As stated at the beginning of this chapter the bottleneck is the machine with the highest cycle time without considering MTTR or MTBF. The production line illustrated is in a stable situation. The buffers previous to the bottleneck are full and the subsequent buffers are empty. Therefore, all machines except the bottleneck have waiting times. What can also be seen is that the cycle times increase to the bottleneck and decrease afterwards again. ($CT_1 < CT_2 < CT_3 < CT_4 > CT_5 > CT_6 > CT_7$) For the time window calculation this formation is beneficial because before the bottleneck only draining time windows are possible and after it only filling windows. In this case, for the time window calculation only the empty/full buffer next to the machine has to be considered. If buffer B_4 is full, e.g., the time window is calculated for machine M_3 . Buffer B_3 does not have to be considered for the flex window duration. In figure 3.15 a different case is shown. As illustrated the cycle times of the production line are given with $CT_1 < CT_2 < CT_3 < CT_4 > CT_5 > CT_6 < CT_7$. The bottleneck is similar to the first case machine four. Due to the fact that the cycle times are not monotonously decreasing (after the bottleneck), full buffers are possible, too. This situation is shown in figure 3.14. At the point in time t_0 the initial situation of the production line is illustrated. Since all buffers are full or empty, flexible time windows are created for every machine. At the point in time t_1 the time window of machine seven ends. All other machines are also producing. The problem which occurs now is that buffer N_7 is full and machine M_6 is producing faster than machine M_7 . That leads to a new

time window calculation for machine M_6 due to a full buffer. This is a draining window after the bottleneck and could have an influence on machine M_5 if it is too long. As a result also buffer B_6 has to be considered in the course of the time window calculation. To summarize if filling windows occur in front of the bottleneck the buffer previous to the machine has to be taken into account and if draining windows after the the bottleneck occur the buffer next to the machine has to be taken into account.

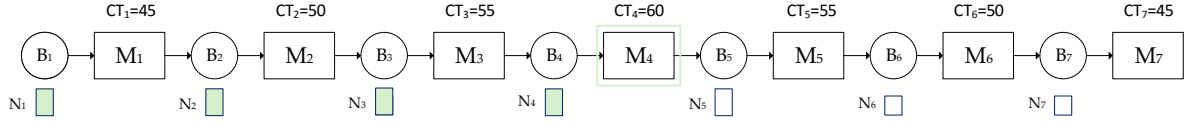


Figure 3.14: Bottleneck detection

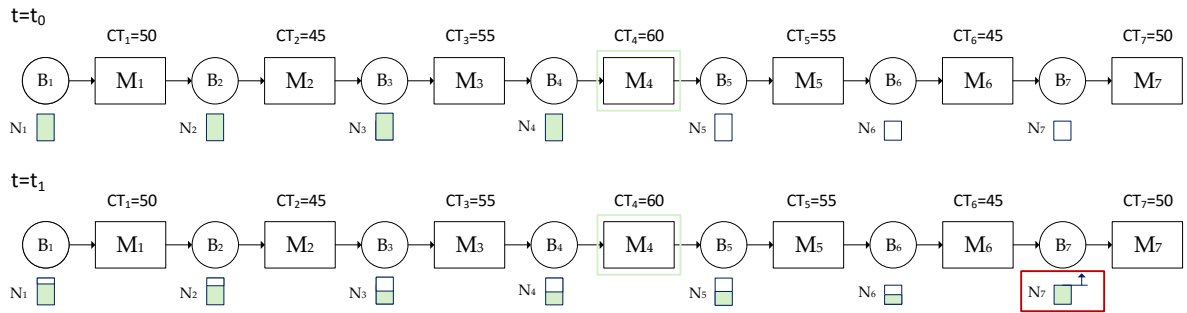


Figure 3.15: Bottleneck influence

Another influence on the bottleneck is possible with a cycle time constellation of $CT_{i-1} > CT_i < CT_{i+1}$ previous to the bottleneck or $CT_{i-1} < CT_i > CT_{i+1}$ after it. Figure 3.16 shows this case. At time t_1 all buffer are full and for the machines M_1 and M_2 a flexible time window is set. At t_2 machine M_2 starts again and buffer B_2 goes empty. At time t_3 all machines work again. Through the fact, that the cycle time of machine M_2 is lower than from machine M_1 a new time window would be created as explained in the previous paragraph. The problem in this case is that the buffer-level of buffer B_3 is low. Hence many short time windows would be created for machine M_2 and the possibility of an influence on the throughput of the bottleneck increases. This cycle is illustrated with the time steps $t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow t_6$. To counteract this the time window of machine M_1 is reduced for single machines to the end of the time window from the faster machine. In this case machine M_1 would start to produce again with machine M_2 at point in time t_2 . For redundant machines the utilization degree μ is set to 0.7, to get acceptable time window durations with a small impact on the bottleneck.

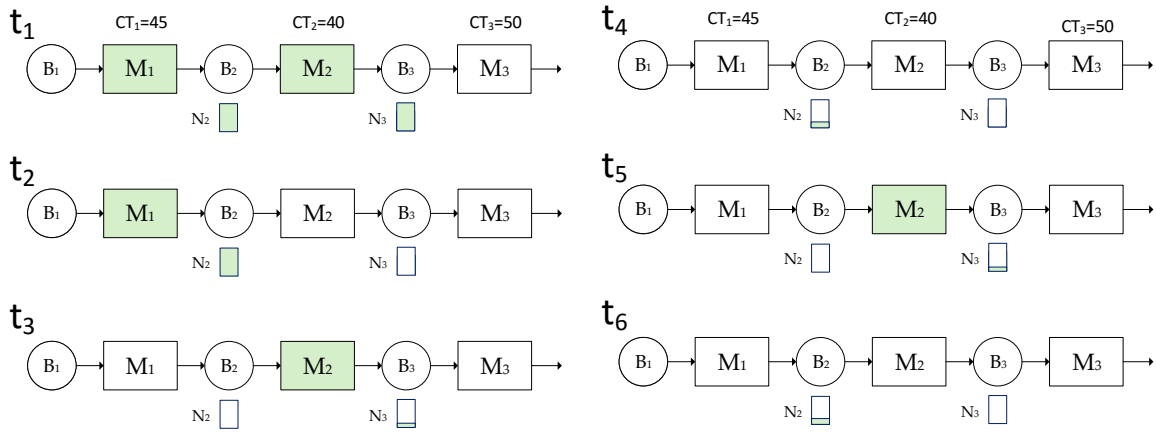


Figure 3.16: Bottleneck influence of cycle time

3.4.2 Task scheduler

In this section the second aspect of the problem definition is modelled with a linear model according to Dantzig (1966) (see chapter 2.2.2). The model has been designed according to the three step process by Dantzig (1966) as well as the theoretical framework of scheduling and optimization for maintenance.

Following the creation of flexible time windows the maintenance tasks have to be scheduled. A set of tasks J has to be scheduled on a set of machines I . Every task is specified by a due date te_j , an earliest start ts_j , a special machine i on which the process has to be fulfilled and a skill set R_{j_s} which is required to perform the task. Furthermore, every task has an optimal execution date with a cost function if it differs from it.

From the time window planner a set of possible flexible time windows K is defined. These windows are specified by a start time tws_k , an end time twe_k and a specified machine i on which the time window takes place. Moreover, the global planner sets some fixed maintenance windows in which all machines can be maintained. These fixed windows have higher costs than the flexible ones. This is considered with the window costs in the objective function.

To perform the tasks on the machines employees are needed. These employees are abbreviated by the index e . Every employee has a set of skills S_{e_s} which has to be higher than the required skill set of the task. Another constraint is that employees have specified shift plans according to which they are present at the company. Every employee has specific costs c_e which have to be considered.

Other resources like spare parts have not been considered because the tasks which have to be scheduled are mainly TPM tasks and therefore spare parts availability is not critical. (see page 44)

In figure 3.2 the function of the task scheduler is shown. The input is the task-list, flexible windows from the flexible window creator, fixed windows from the global planner and an employee list with shift plans. The output is supposed to be scheduled tasks to time windows with assigned employees.

Notation

Decision variables

x_{jk}	task j performed in time window k
per_{je}	employee e performs task j
emp_{jke}	task j is performed in window k from employee e

Task variables

$i = 1, \dots, m$	Machine
$j = 1, \dots, n$	Task
d_j	Duration of task j
te_j	Due date of task j on machine i
ts_j	Earliest starting point of TPM task j
c_j	Cost regarding to variation of time t to T_{opt} of task j
C_j	Gradient of cost function c_{ij}
T_{opt}	Optimal point in time for task j
Per_{ij}	Minimal needed persons for task j
$avail_{it}$	Available time windows for machine i
F_{ji}	Task machine matrix
A_{ik}	Task machine matrix
T_{jk}	Earliest start and due date of task fit to the time window
K_{jk}	Task j is possible to be performed in time window k
R_{js}	Skill set needed for task j $s \in S$

Employee variables

$e = 1, \dots, E$	employees in system
$s = 1, \dots, S$	Skill set (electric, mechanic)
S_{es}	Skill set of employee e $s \in S$
c_e	Cost of employee e

$labor_j$ Labour cost of job j
 $avail_e$ Availability of employee e

System variables

$k = 1, \dots, n$ System time windows
 K Set of time windows K
 tws_k Start time of time window k
 twe_k End time of time window k
 D_H Planning horizon

Figure 3.17 shows the notations for a scheduled task j in a time window k with the earliest start, the due date, the start of a/the window, the end of a/the window and the duration of a task.

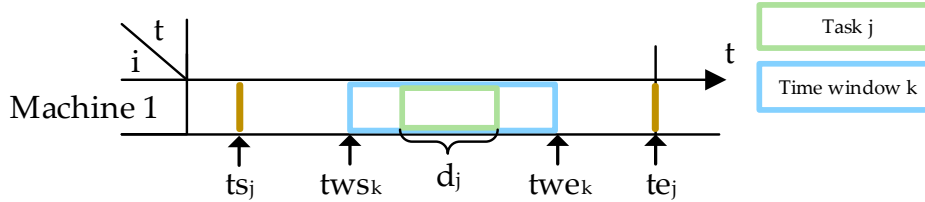


Figure 3.17: Timeline of scheduled task j in time window k

Decisional variables

For the linear model there are three binary 0-1 decision variables. The first variable x_{jk} defines if a task j is performed in the time window k .

$$x_{jk} = \begin{cases} 1 & \text{if task } j \text{ is performed in time window } k \\ 0 & \text{else} \end{cases}$$

The variable $per_{je} = 1$ if the employee e has to perform the task j .

$$per_{je} = \begin{cases} 1 & \text{if task } j \text{ is performed by employee } e \\ 0 & \text{else} \end{cases}$$

Variable emp_{jek} is the connection between the variables x_{jk} and per_{je} .

$$emp_{jek} = \begin{cases} 1 & \text{if task } j \text{ is performed by employee } e \text{ at time window } k \\ 0 & \text{else} \end{cases}$$

Constraints

The first constraints define the link between the decision variables.

$$emp_{j,e,k} \leq x_{jk} \quad \forall j, k \quad (3.13)$$

$$emp_{j,e,k} \leq per_{je} \quad \forall j, k \quad (3.14)$$

$$\sum_e \sum_k emp_{j,e,k} = \sum_e per_{je} \quad \forall j \quad (3.15)$$

Time constraints

The first time constraint is that all tasks have to be scheduled in the time period. The sum of the time windows also includes that the task can only be assigned to one time window.

$$\sum_{k \in K} x_{jk} = 1 \quad \forall j \quad (3.16)$$

The availability of a task for a time window is calculated by the task-machine matrix and the window-machine matrix.

$$F_{ji} = \begin{cases} 1 & \text{If task } j \text{ is done at machine } i \\ 0 & \text{else} \end{cases} \quad (3.17)$$

$$A_{ik} = \begin{cases} 1 & \text{If window } k \text{ is available on machine } i \\ 0 & \text{else} \end{cases} \quad (3.18)$$

$$T_{jk} = \begin{cases} 1 & \text{If } tws_k \geq ts_j \text{ and } twe_k \leq te_j \text{ (fig.3.17)} \\ 0 & \text{else} \end{cases} \quad (3.19)$$

$$\mathbf{K} = (\mathbf{FA})\mathbf{T} \quad \forall j, k, i \quad (3.20)$$

In figure 3.18 the calculation of the matrix K_{jk} is shown. As illustrated every task has to be on a specific machine. In matrix A_{ik} the sum $\sum_{k=0}^o A_{ik}$ also has to be 1 $\forall i$. First the product of the matrix F_{ji} and A_{ik} is calculated if a time window is in generally available for a task. T_{jk} specifies if the earliest start time and the due date of the task fit in the time window. The result is the K_{jk} matrix which determines whether a time window is available for a task or not. The necessary constraint to the availability of the task to time window is the following:

$$x_{jk} \leq K_{jk} \quad \forall j, k \quad (3.21)$$

F_{ji}-matrix

i \ j	Machine1	Machine2	Machine3	Machine4	Machine5	Machine6	...	i=m	sum
Task1	1	0	0	0	0	0	...	0	1
Task2	0	0	0	1	0	0	...	0	1
Task3	1	0	0	0	0	0	...	0	1
Task4	0	0	1	0	0	0	...	0	1
Task5	0	0	0	0	0	1	...	0	1
Task6	1	0	0	0	0	0	...	0	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
j=n	0	0	0	0	1	0	...	0	1
sum	5	9	3	5	2	1	...	0	

A_{ik}-matrix

i \ k	Window1	Window2	Window3	Window4	Window5	Window6	...	k=0	sum
Machine1	0	0	0	1	0	0	...	0	5
Machine2	0	0	1	0	0	0	...	0	2
Machine3	0	1	0	0	0	0	...	0	3
Machine4	0	0	0	0	1	0	...	0	4
Machine5	0	0	0	0	0	1	...	0	1
Machine6	1	0	0	0	0	0	...	1	7
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
i=m	0	0	0	0	0	0	...	0	8
sum	1	1	1	1	1	1	...	1	

T_{jk}-matrix

j \ k	Window1	Window2	Window3	Window4	Window5	Window6	...	k=0
Task1	1	1	0	1	0	0	...	0
Task2	0	0	0	0	0	0	...	0
Task3	1	1	1	0	0	0	...	0
Task4	0	0	1	0	0	0	...	0
Task5	1	1	1	1	0	0	...	1
Task6	0	0	0	1	0	0	...	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
j=n	0	1	1	1	0	1	...	0

\times

K_{jk}-matrix

j \ k	Window1	Window2	Window3	Window4	Window5	Window6	...	k=0	sum
Task1	0	0	0	1	0	0	...	0	3
Task2	0	0	0	0	0	0	...	0	4
Task3	0	1	0	0	0	0	...	0	2
Task4	0	0	1	0	0	0	...	0	1
Task5	1	0	0	0	0	0	...	1	5
Task6	0	0	0	1	0	0	...	0	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
j=n	0	0	0	0	0	1	...	0	1
sum	2	3	8	1	6	4	...	2	

$=$

Figure 3.18: Calculation availability

Another time constraint can be defined as the sum of all task durations which are assigned to a window k and which have to be smaller than the length of the window k .

$$d_k \geq \sum_{j=0}^n d_j x_{jk} \quad \forall k \tag{3.22}$$

Employee constraints

The first employee constraint is that the skill set of e has to be higher than the required skill set of j . The skill set is shown in figure 3.19 on the left-hand side. For every task a matrix exists.

$$per_{je} \leq able_{je} \quad \forall j, e \tag{3.23}$$

$$able_{je} = \begin{cases} 1 & \text{If employee } e \text{ is able to perform task } j \rightarrow S_{es} \geq R_{js} \\ 0 & \text{else} \end{cases}$$

The equation 3.24 defines that the number of employees working on the task has to be equal to the required number.

$$\sum_{e \in E} per_{je} = Per_j \quad \forall j \quad (3.24)$$

S_{es}-matrix

e \ s	Skill1	Skill2	Skill3	Skill4	...	s=q	sum
Employee1	1	0	1	0	...	0	5
Employee2	0	1	0	1	...	1	2
Employee3	1	0	0	0	...	0	3
Employee4	0	0	1	0	...	1	4
Employee5	0	0	0	0	...	0	1
Employee6	1	0	1	0	...	0	7
⋮	⋮	⋮	⋮	⋮	⋮	⋮	
e=p	0	1	0	1	...	0	8
sum	5	9	3	5		0	

A_{ek}-matrix

i \ k	Window1	Window2	Window3	Window4	Window5	Window6	...	k=0
Employee1	0,3	0	0	0,5	0	0,1	...	0
Employee2	0	0	1	0	0	0	...	0
Employee3	0	1	0	0	0	0	...	0
Employee4	0	0,3	0	0	1	0,5	...	0
Employee5	0	0	0	0	0	1	...	0
Employee6	1	0	0	0,2	0	0	...	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
e=p	0,9	0	0	0	0,1	0	...	0

Figure 3.19: Employee matrices

Figure 3.19 on the right-hand side illustrates the availability matrix of employee e for the time window k . A_{ek} is calculated by the shift plan of the employees and by the fact if an employee is available for a machine and therefore for a window. The duration d_{ek} in window k defines the time an employee is available in this window.

$$0 \leq A_{ek} \leq 1 \quad (3.25)$$

$$d_{ek} = A_{ek} d_k \quad (3.26)$$

$$d_{ek} \geq \sum_{j=0}^n \left(\sum_{k=0}^o (O_{kl} emp_{jke}) d_j \right) \quad \forall e, k \quad (3.27)$$

$$O_{kl} = \begin{cases} 0 & \text{If } t_{we_l} \leq t_{ws_k} \text{ or } t_{ws_l} \geq t_{we_k} \\ \frac{t_{we_l} - t_{ws_k}}{d_k} & \text{If } t_{ws_k} \leq t_{we_l} \leq t_{we_k} \text{ and } t_{ws_l} \leq t_{ws_k} \\ \frac{t_{we_k} - t_{ws_l}}{d_k} & \text{If } t_{ws_k} \leq t_{ws_l} \leq t_{we_k} \text{ and } t_{we_l} \geq t_{we_k} \\ \frac{d_l}{d_k} & \text{If } t_{ws_k} \leq t_{we_l} \leq t_{we_k} \text{ and } t_{ws_k} \leq t_{ws_l} \leq t_{we_k} \\ 1 & \text{else} \end{cases} \quad (3.28)$$

$$\forall k, l \in K$$

The constraint 3.27 defines the maximum availability of an employee on a time window by considering the overlapping windows. Figure 3.20 shows the overlapping possibilities. Due to the fact that time windows could overlap, employees could be scheduled to tasks on different windows at the same time. This should be avoided by the O_{kl} factor. A certain risk of overlapping tasks with the same assigned employee still exists. This arises because all tasks are assigned to time windows but not scheduled to specific starting times and end times. The positive effect is that the computation time decreases and the maintenance engineer at the production line has still a margin for changes. The negative aspect is, if time windows overlap and the utilization of these windows is on maximum a non feasible solution regarding to employees could occur. To counteract this, a possibility would be to reduce the flexible time window duration to a specific degree at the input and therefore set a safety time buffer.

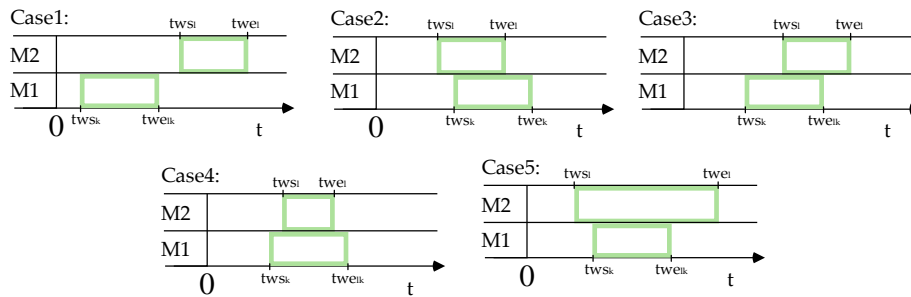


Figure 3.20: Overlapping matrix O_{kl}

Objective Function

The objective function consists of three different terms. The first one defines penalty costs if a task is performed in a fixed window. This should force the solver to fill up the flexible time windows. The second term is a penalty function for a different execution than the optimal maintenance time. Figure 3.21 shows a linear penalty function. The third term considers different costs of the employees if they execute the task. Therefore over qualification should be avoided.

$cfix_j \implies$ Penalty function through execution in fixed time window

$c_j \implies$ Penalty function through different execution date then $topt$ (figure 3.21)

$clabor_j \implies$ Labor costs

$$\min z = \sum_{j \in J} (\alpha \cdot cfix_j + \beta \cdot c_j + \gamma \cdot clabor_j) \quad (3.29)$$

$$cfix_j = \sum_{k \in K} x_{jk} c_k \quad \forall j \quad (3.30)$$

$$c_k = \begin{cases} 1 & \text{If window is a fixed window} \\ 0 & \text{else} \end{cases}$$

$$c_j = \sum_{k \in K} C_j x_{jk} |(Topt_j - Tws_k)| \quad \forall j, k \quad (3.31)$$

$$clabor_j = \sum_{e \in E} per_{je} c_e d_j \quad \forall j \quad (3.32)$$

α, β, γ are the weighting factors of the objective function terms. Due to the fact that the cost for deviation to the optimal execution date and the cost of execution in a fixed time window are difficult to quantify, these weighting factors are needed. To set this factors experimental runs of the algorithm on production lines are required.

$$\min z = \sum_{j \in J} \sum_{k \in K} \alpha \cdot x_{jk} \cdot c_k + \sum_{j \in J} \sum_{k \in K} \beta \cdot C_j \cdot x_{jk} \cdot |(Topt_j - Tws_k)| + \sum_{j \in J} \sum_{e \in E} \sum_{k \in K} \gamma \cdot x_{jek} \cdot c_e \cdot d_j \quad (3.33)$$

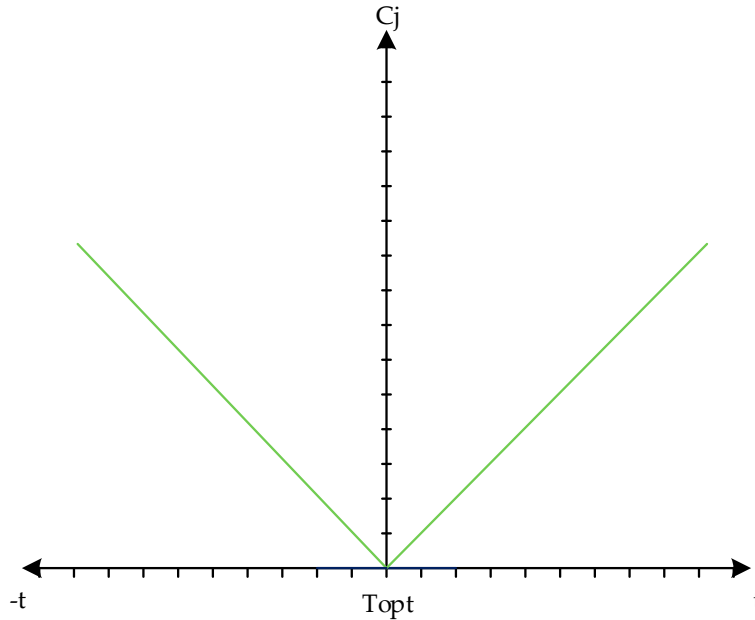


Figure 3.21: Penalty-Function

Chapter 4

Implementation

This chapter shows the transfer from a mathematical model to a computer readable model and the according program. Robinson (2013) defined this steps as the transfer from the model design to the computer model through coding (see page 28). The second section discusses the validation of the developed algorithm with three different production lines.

4.1 Algorithm

This section deals with the model translation from the mathematical model to a computer model. The entire implementation was done by using C# in a DotNet Framework. A pseudo-code is used to illustrate the major algorithm used during this implementation phase. This section is also divided into two parts, the time window planer and the task scheduler. In the entity–relationship model (see Appendix B) the data model and interfaces to the global planner can be seen.

4.1.1 Time Window Planner

Algorithm 3 shows the initialization of the time window planner. Input data are the machine-list with redundancies of the production line and the buffer-list. The first step of the algorithm is to combine the redundant machines with calculating the cycle time C_{High} for these machine combinations. The second step is to determine if the buffer capacity is zero. If this is the case, the cycle time for this step in process will be forwarded

to the next one in the machine-buffer-machine system. Afterwards a list of the existing bottlenecks of the production line is created. Bottlenecks are, as stated in chapter 3.4.1, the process steps with the highest cycle times. These first three steps are the preprocessing of the planner. The main algorithm starts with the request whether the start of the algorithm can be related to the breakdown of a machine or not. If this condition can be verified, the expected breakdown duration and the ID of the broke machine (Break ID) have to be queried. Subsequently a time window is created, in advance to the activity scanning for the broken machine. The last step of this basis algorithm is to start the sub program, the time window calculator. If the condition of a breakdown can be falsified, the time window calculator is started directly.

Algorithm 3: Time window planner

Input: machinelist, redundancies, bufferlist
 Combine redundant machines with redundancy matrix;
 Change CT if buffer capacity==0;
 Find bottlenecks;
if *start regarded to breakdown==true* **then**
 | read breakID, breakDuration;
 | create timewindow for breakID;
 | **start Time window calculator algorithm 4;**
else
 | **start Time window calculator algorithm 4;**
end

The data structure of the input list machine and the input list buffer is shown in the tables 4.1 and 4.2. The list machine contains the attributes of the cycle time, the redundancies to other machines, the capacity of the machine and the system ID. Starting from the algorithm, a primary ID is assigned to avoid a mix-up if redundant machines are combined. Another attribute of the program is the calculation of *CThigh* for redundant machines. *CThigh* is the cycle time a redundant machine combination holds if one machine is shut down. All the information, which is needed from the buffer, is the start level and the maximal capacity. The redundancies at table 4.1 are the link between the working operation and the machines. Through the merge of the redundant machines in Algorithm 3, the working operations correspond to the number of machines.

This paragraph defines the parameters of the discrete process according to Banks et al. (2004, p.33). The **system** is defined by a production line. **Entities** within this system are machines and buffers which are connected in a defined way. The boundary of the

Table 4.1: Input Machine

	Attributes	Data type	Description
Machine	ID	int	Identification number of machine
	CT	double	Cycle Time of machine
	CThigh	double	Cycle Time for a redundant machine combination if one machine is down
	Redundancies	List(int)	List of redundant machines
	Capacity	int	Number of parts which can be operated
	SystemID	string	ID from BDE system

Table 4.2: Input Buffer

	Attributes	Data type	Description
Buffer	ID	int	Identification number of machine
	Capacity	int	Maximum possible number of parts in buffer
	Level	int	Actual buffer level

system is assumed to be an infinite source of raw parts to the production process and an infinite sink of finished parts at the end of the process. The mathematical **model** is discussed in chapter 3.4.1. The **system state** is defined by the buffer levels and the machine cycle times at a specific point in time. **Attributes** of the entities are shown in the tables 4.1 and 4.2. **Events** which occur are flexible maintenance windows and consequently cycle time changes. The **event notices** of the time windows are defined by a specific point in time and a cycle time. The **activity** is the state of the machine if the cycle time changes. There are the activities normal production (standard CT), complete shut off ($CT = \infty$) or production with a *CThigh*.

As stated in chapter 3.4.1, the approach for the simulation based calculation of the time windows is activity scanning (explanation on page 35). The following section defines the activities, conditions and actions of the algorithm. Algorithm 4 illustrates the process of activity scanning.

At the beginning the start time is set to zero. Afterwards a while loop until the end of the planning horizon is carried out. The three activities in the loop are "produce with individual rate", "produce with standard rate" and "flexible time window". The first activity "produce with individual rate" is triggered by the action "set new production

Algorithm 4: Program sequence

Input: utilization buffer, tstep, bufferlist, StartTime
 $t = 0$ **Initialization;**
while $t \leq tend$ *Terminate simulation?* **do**
 Activity Produce with individual rate;
 Activity Produce with standard rate;
 Update bufferlevels;
 for *Machines upstream to first bottleneck machine; $i=bottlenecks; i \geq 0; --i$* **do**
 | **Activity** Flexible time window
 end
 for *Between bottleneck machiens; $i=Firstbottleneck+1; i \leq Lastbottleneck; ++i$*
 do
 | **Activity** Flexible time window
 end
 for *Machines downstream to last bottleneck machine; $i=Lastbottleneck; i \leq$*
 LastMachine; ++i **do**
 | **Activity** Flexible time window
 end
 $t = t + tstep$ **Time Flow Mechanism;**
end

rate” and only occurs at redundant machines if a time window already exists and the cycle time has to be aligned to another $CThigh$. The action ”reset production rate to standard” starts the second activity and occurs if a time window ends and the cycle time is set back to standard cycle time. The last activity is started by the action ”set flexible time window” and occurs through a empty or full buffer. After all conditions of the activities have been examined, the time is set to the actual time plus the time step by the Time Flow Mechanism. The decision point ”Any other condition satisfied” in figure 2.4.2 can be skipped because of the sequence of the queries. As stated algorithm 4 defines the process of activity scanning. The algorithms 5-7 are subroutines which illustrate the stated actions. The according tables should give an overview of the single activities. The name of the algorithms is according to the action of the activity.

The first activity ”Produce with individual rate” is illustrated in algorithm 5 and table 4.3. The algorithm checks if a cycle time change event occurred in the last time interval. This signifies that the machine is already in a flexible maintenance window and changes, for example, the cycle time from ∞ to $CThigh$. These cycle time changes are a list of events per time window, set at the calculation of the windows, which contain a point in

time once a cycle time will be changed to a different one. At every machine the last time window is checked.

Table 4.3: Activity: Produce with individual rate

Activity	Produce with individual rate
Condition	$t_{act} - t_{step} < t_{wc_i} \leq t_{act} \quad \forall i$
Description	Production with an individual production rate. Can only occur at redundant machines.
Action	Algorithm 5 Set new production rate

Algorithm 5: Set new production rate

Input: machinelist, Event list of cycle time changes

foreach *machine* **do**

if *actual time - time step* < *Time of next CT change* < *Actual time* **then**
 | CT=newCT
else
 | break
end

end

The second activity is called: "Produce with standard rate". At the end of the time window the production rate of the machine is set back to the standard production rate. As shown in algorithm 6, the last interval is scanned for a time window end event of a machine. If the condition can be verified, the cycle time is set back to the start cycle time. The activity can only occur if the machine has an actual activity the activity of "Flexible time window" or "Produce with individual production rate". The "Flexible time window" could be at a single machine or a redundant machine combination.

Table 4.4: Activity: Produce with standard rate

Activity	Produce with standard rate
Condition	$t_{act} - t_{step} < t_{we_k} \leq t_{act} \quad \forall i$
Description	Production with standard production rate. The machine or machine combination has no active time window.
Action	Algorithm 6 Reset production rate to standard

Algorithm 6: Reset production rate to standard

Input: machinelist, Time window list

```

foreach machine do
  | if actual time - time step < Time window end of last time window < Actual time
  |   then
  |   | CT=startCT
  |   else
  |   | break
  |   end
end

```

The condition for the activity flexible time window checks if a buffer level is lower than zero or higher than the maximum capacity. If this is true, the algorithm 8 is started. This algorithm calculates the time window duration, the start cycle time and cycle time changes for the machine as explained in chapter 3.4.1.

Table 4.5: Activity: Flexible time window

Activity	Flexible time window
Condition	Buffer check $L_i < 0$ or $L_i > C_i \quad \forall i$
Description	Creates time window if Buffer is full or empty
Action	Algorithm 7 Set flexible time window

Algorithm 7: Set flexible time window

Input: machinelist, Buffer list, Actual time

```

foreach buffer do
  | if bufferlevel < 0 or bufferlevel > Capacity then
  |   Create Time Window
  | else
  |   | break
  |   end
end

```

The first IF-statement examines if the machine is a bottleneck machine. If this is the case, the duration of the time window depends on the window of the previous machine or on the window next to this machine. If the machine is not a bottleneck, the standard time window calculation (chapter 3.4.1) is executed. As already pointed out in chapter 3.4.1, the position of the machine and the start cycle times have to be checked. The output of

algorithm 8 is a time window duration and the associated cycle time changes.

Algorithm 8: Create time window

```

if bufferlevel < 0 then
  | if machine==bottleneck then
  | | Time window duration= Time window end of previous machine - Actual
  | | Time;
  | | Set new cycle time and time window end
  | else
  | | Window1:Calculation Time window according to chapter 3.4.1 page 55
  | end
  | if Machine is previous to bottleneck then
  | | According to chapter 3.4.1 page 57;
  | | Window2: Calculation Time window for buffer after machine;
  | | Add Window2 if duration is less than duration from Window1
  | else
  | | Check CT of previous machine and calculate a new time window end if
  | | necessary according to page 3.4.1
  | end
else
  | if machine==bottleneck then
  | | Time window duration= Time window end of next machine - Actual Time;
  | | Set new cycle time and time window end
  | else
  | | Window1:Calculation Time window according to chapter 3.4.1 page 55
  | end
  | if Machine is after bottleneck then
  | | According to chapter 3.4.1 page 57;
  | | Window2: Calculation Time window for buffer previous to machine;
  | | Add Window2 if duration is less than duration from Window1
  | else
  | | Check CT of next machine and calculate a new time window end if
  | | necessary according to chapter 3.4.1
  | end
end

```

The overall output of the time window planner is a list of available opportunity windows for every machine. The data structure is shown in table 4.6. The windows contain a start time and an end time of the maintenance window. The variable window cost defines if a

window is a fixed or a flexible window. This factor is a control variable of the algorithm and defines allocation to flexible windows. For further investigations, this cost factor is set to 0,5 in order to balance the utilization of both - flexible and fixed - time windows. Cycle time changes are no output variable any more, because they are only needed for calculating the time window duration. The output of the program for the window planer can be seen in figure 4.1. The Gantt Chart shows the possible time windows for all machines. The green bars show a fixed time window at the beginning of the planning horizon. The red bars display the flexible maintenance opportunities.

Table 4.6: Output Time Window

	Attributes	Data type	Description
Time Window	ID	int	Identification number of window k
	Start time	DateTime	Start of the time window
	End time	DateTime	End of time window k
	Duration	TimeSpan	End time-Start time
	Machine	int	ID of belonging machine
	Window costs	double	Cost factor of the time window k

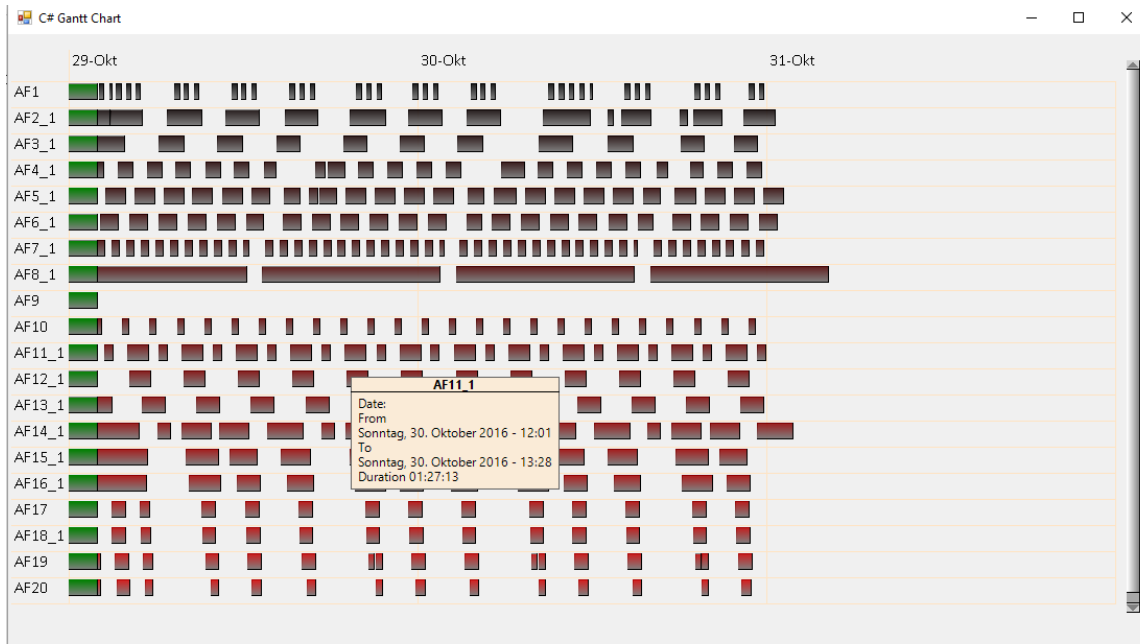


Figure 4.1: Gantt Chart of the time windows

4.1.2 Task scheduler

Algorithm 9 shows the process of the task scheduler. At the beginning all input data are imported. The next step is the pre-processing of the availability and the determination of the ability matrix. Subsequently the linear integer program, according to chapter 3.4.2, is defined. As optimisation software Gurobi is chosen, which has an object-oriented interface to C#. After the optimization, a Gantt-Chart with the flexible/fixed window, a task-window list and a task-employee list are created.

Algorithm 9: Task scheduler

Input: Task list, Employee list, Flexible time window, Fixed time window

Output: Gantt chart, Task list per employee, Task list per window

Read task list;

Read employee list;

Read flexible and fixed time windows;

Calculate availability of employees to time windows with shift plan and employee-machine matrix;

Calculate ability matrix with skills of employees and skill requirements of tasks;

Define linear integer program according to chapter 3.4.2;

Optimization with Gurobi;

Create output lists;

The input data of the time windows are the same as shown in table 4.6. Flexible windows have a cost factor of 0.5, whereas fixed time windows hold a factor of 1. The input of the task and employee list is shown in table 4.8 and table 4.7 on the next page. The output of the overall program is illustrated in the figures 4.2 and 4.3. In the first one the allocation of tasks and time windows to employees can be seen. With a combo-box the required employee can be chosen. The second one shows the allocation of task to time window. In the Appendix C the menu of the program can be seen.

Table 4.7: Input Employee

	Attributes	Data type	Description
Employee	ID	int	Identification number of employee e
	Cost	int	Labour cost of employee
	Skill	List(int)	Skill set of employee
	Shift plan	List(DateTime)	Start and End times of shift plan
	Machine	List(Machine)	Special machine allocation

Table 4.8: Input Task

	Attributes	Data type	Description
Task	ID	int	Identification number of task j
	Machine	int	Belonging machine ID of task
	Duration	TimeSpan	Duration of task
	Earliest Start	DateTime	Earliest start of task
	Due date	DateTime	Due date of task
	Optimal time	DateTime	Optimal point of time for maintenance
	Persons	int	Needed Persons for task
	Skill	List(int)	Required skill set of task
	Cost factor	int	Cost factor for deviation of optimal maintenance point

Task_Employee_List

Maschinenbau- und Betriebsinformatik **mbi** **TPM4.0**

Choose Employee:

	Employee	Task	Window Start	Window End
▶	Bence	87	16.11.2016 13:53	16.11.2016 13:59
	Bence	90	16.11.2016 11:21	16.11.2016 11:27
	Bence	102	16.11.2016 08:50	16.11.2016 08:56
	Bence	103	16.11.2016 10:05	16.11.2016 10:11
	Bence	107	16.11.2016 15:09	16.11.2016 15:15
	Bence	112	16.11.2016 16:25	16.11.2016 16:31
	Bence	116	16.11.2016 12:37	16.11.2016 12:43
	Bence	127	16.11.2016 13:31	16.11.2016 13:48
	Bence	129	16.11.2016 13:31	16.11.2016 13:48
	Bence	137	16.11.2016 13:31	16.11.2016 13:48
*				

Figure 4.2: Output Employee-Task List

Task	Machine	Window Start	Window End
13	AF220_1+1ZE a...	10.11.2016 02:51	10.11.2016 03:04
14	AF220_1+1ZE a...	10.11.2016 02:51	10.11.2016 03:04
15	AF220_1+1ZE a...	10.11.2016 02:51	10.11.2016 03:04
16	AF230	10.11.2016 02:42	10.11.2016 03:04
17	AF230	10.11.2016 02:42	10.11.2016 03:04
18	AF240	10.11.2016 05:32	10.11.2016 06:59
19	AF240	10.11.2016 05:32	10.11.2016 06:59
20	AF240	10.11.2016 05:32	10.11.2016 06:59
21	AF240	10.11.2016 05:32	10.11.2016 06:59
22	AF240	10.11.2016 05:32	10.11.2016 06:59
22	AF240	10.11.2016 05:32	10.11.2016 06:59

Figure 4.3: Output Task-Window List

4.2 Validation

This section focuses on the validation of the time window planner on a virtual production line, which was implemented in the simulation software Plant Simulation[®]. Therefore, the experimental design is explained in the following sections and the three scenarios that are used to validate are introduced. Finally the obtained results are presented and discussed.

4.2.1 Validation design

According to Banks et al. (2004, p. 310), the first objective of validation is to find out if the model is correct. The second objective is to obtain an accurate abstraction of the real world. The validation of the time window planner algorithm is done with a DES using the software Plant Simulation[®]. In figure 4.4, on the right-hand side, the abstraction levels of the three systems can be seen. At the bottom the real system is displayed. With the help of Plant Simulation[®] the discrete event simulation abstracts the real problem to a specific degree. The Plant Simulation[®] model is abstracted by the developed algorithm

in a certain way. Therefore, the highest degree of abstraction can be detected through the model. To achieve the objective of validation three different line formations were analysed to include all possible set ups. The validation process can be seen in figure 4.4. After the modeling of the three production lines in Plant Simulation[®], the first step was to simulate the lines for two days to get a stable production line. After these two days the buffer levels, based on the Plant Simulation[®] output, were taken as input for the time window calculation. Afterwards the program calculated buffer changes and possible maintenance opportunities windows for another two days. These flexible time windows were implement into Plant Simulation[®] and the machines were shut down accordingly for the following two days. As a result deviations on the throughput of the bottleneck were determined. Furthermore blocking and waiting times of these machines - due to the time windows - were used for the validation of the algorithm. Buffer changes over the time were tracked in a minute interval for both Algorithm and Plant Simulation[®]. The next step was to compare and evaluate them.

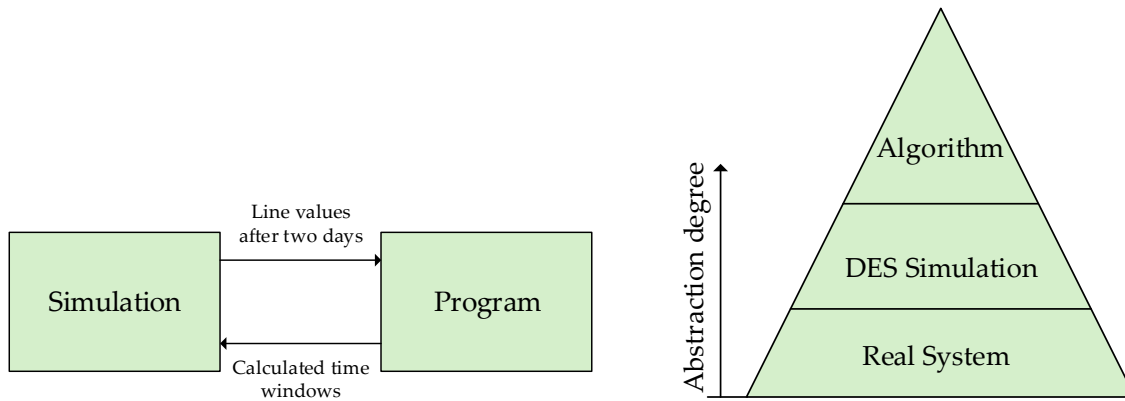


Figure 4.4: Validation process and abstraction levels

The software uses discrete event simulation to optimize and analyse the performance of the material flow, the resource utilization and all levels of logistics. It is possible to insert different objects, like machines or buffers, and program a specific behaviour. In figure D1 the basic objects of the plant simulation model can be seen. The first object "Quelle" is the source of the production line. Objects with a "B" define a buffer, and objects with an "AF" a machine. The last object "Senke" defines the sink of the line.

To validate the different possibilities of the line structure three different production lines were set up: a single machine, a redundant and a mixed production line. For each of these three lines two different cases were simulated, one with increasing cycle times to the bottleneck and decreasing after it and one with mixed cycle times. In order to be able to compare the lines with one another, the buffer capacities were the same in both

of the previously mentioned cases. Table 4.9 shows the cycle times and buffer capacities previous to the bottleneck machine, table 4.10 shows the same but sub-sequential to the bottleneck. The abbreviations for the cases are the following:

- Single production line case1 → Single1
- Single production line case2 → Single2
- Redundant production line case1 → Redundant1
- Redundant production line case2 → Redundant2
- Mixed production line case1 → Mixed1
- Mixed production line case2 → Mixed2

Table 4.9: Cycle times and buffer capacity previous to bottleneck

Case	AF1	AF2	AF3	AF4	AF5	AF6	AF7	AF8
1	38	39	40	42	44	45	48	50
2	45	39	40	45	41	46	48	50
Buffer	B1	B2	B3	B4	B5	B6	B7	B8
	30	20	40	25	10	15	30	10

Table 4.10: Cycle times and buffer capacity after bottleneck

Case	AF8	AF9	AF10	AF11	AF12	AF13	AF14	AF15
1	50	49	47	46	44	42	41	40
2	50	49	40	45	44	46	38	42
Buffer	B8	B9	B10	B11	B12	B13	B14	B15
	10	15	30	40	20	40	20	30

Fifteen machines were chosen to show the influence of a long linkage from the first machine to the bottleneck. Another reason was to validate all the possible line dynamics discussed in chapter 3.4.1. Different cycle time constellations were modelled in case2. AF1 → AF3 and AF4 → AF6 show the case of a $CT_{i-1} > CT_i < CT_{i+1}$ previous to the bottleneck. AF10 → AF12 and AF12 → AF14 show this constellation sub-sequential to the bottleneck. As bottleneck working operation AF8 was chosen in each case. With the two cases and three different types the possibilities and limits of the algorithm are evaluated in the following section.

Figure 4.5 shows the simulated single machine production line including fifteen working operations. A larger illustration can be found in the Appendix D on page 99. Also the redundant production line and the mixed production line can be found in the Appendix.

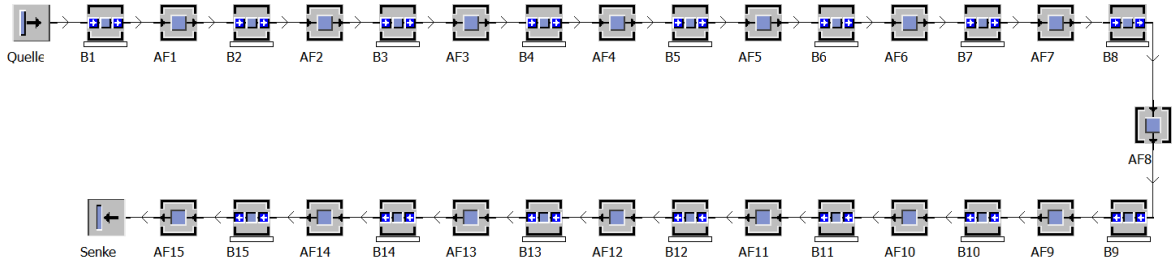


Figure 4.5: Single machine production line in Plant Simulation[®]

4.2.2 Results

Table 4.11 shows the results of the different simulation runs. The stated values refer to the bottleneck, because it defines the overall output of the line. Within two days the standard throughput reached the amount of 3456 parts, not taking flexible time windows into account. The cases Redundant1 and Mixed1 are closest to this result. The amount of produced parts is reduced by seven units in these two days. This result may be explained by the fact that at production lines with redundant machines the production rate differences are smaller, because just one machine is switched off. Finally, the cases one and two of the single machine production line have the third and fourth best throughput. Higher production losses can be seen in case2 of the redundant and mixed production line. It can be concluded that differences in the cycle time, from an increase behaviour previous to the bottleneck and an a decrease behaviour subsequent to it, can have an influence on the throughput at production lines with redundant machine combinations. Nevertheless, it has to be considered that the production rate of the line is in each case over 99.15% and therefore only a small influence of the algorithm can be seen. This means that the algorithm detects the opportunity time windows almost perfect, independent from the structure of the production line. Chapter 5 includes an example of the benefit from the overall algorithm. Another important finding is that the blocking percentage of the bottleneck is negligible. This means that the machines succeeding to the bottleneck are responsible for production losses primarily.

Table 4.11: Utilization statistic of bottleneck

	Single1	Single2	Redundant1	Redundant2	Mixed1	Mixed2	Standard
Throughput	3445	3447	3449	3427	3449	3429	3456
Waiting	0.3%	0.26%	0.18%	0.82%	0.17%	0.78%	0.00%
Blocked	0%	0%	0.02%	0.03%	0.01%	0.01%	0.00%
Production	99.7%	99.74%	99.80%	99.15%	99.82%	99.21%	100%

On the left-hand side, figure 4.6 provides the accumulated durations of waiting and blocking times at the bottleneck machine. The highest accumulated waiting time can be detected in case Redundant2 (23.5 minutes within two days). As also stated in the previous paragraph, blocking times only constitute a small fraction of the production losses. On the right-hand side the average durations of waiting and blockage interruptions are displayed. The deviations for the different cases are smaller compared to the accumulated duration. What is interesting concerning these data is the comparison of the Single1 and Single2 case. Through the lower window duration (figure 4.7) in case Single2 also the average waiting time is 40% less compared to case Single1. This also leads to the higher throughput of case2. The reason for these shorter time windows is the modeling step discussed in chapter 3.4.1.

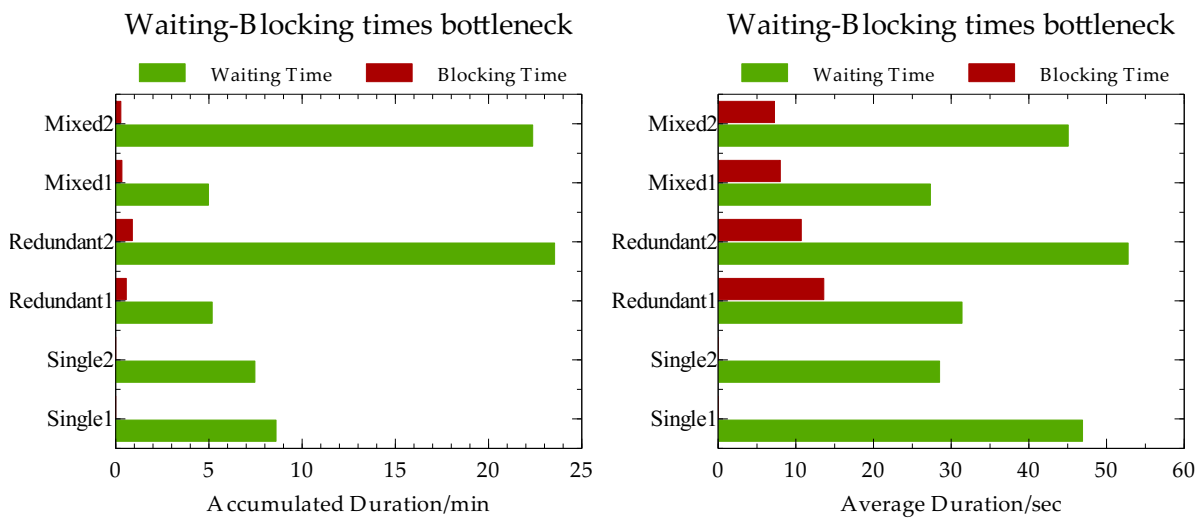


Figure 4.6: Waiting and blocking times of bottleneck for two days

Figure 4.7 shows the comparison of the different production lines regarding average time window durations. The green bar illustrates the average accumulated duration per working operation. This value is calculated by accumulating of all time windows for every machine. Then the average of all machines, except for the bottleneck, is used. No flexible time windows can occur at the bottleneck machine and therefore it is not included and would distort the result. As illustrated the accumulated duration increases with the number of redundant machine combinations. The cases Redundant1 and Redundant2 have the highest values of all cases. However, these numbers provide little information if the number of the machines in the production line is not known. Therefore the red bar displays the accumulated maintenance duration per machine. To calculate this value the accumulated window duration is divided by the number of machines for the process step.

Again, the average of the machines, except for the bottleneck, is used. The green and red bar are equal concerning the single machine production line, because every working step only has one machine. An interesting finding is that the average accumulated duration per machine is a bit lower at redundant and mixed production lines. However, at these lines the average window duration of the time windows is more than the double at best compared to the single production lines. Therefore, tasks with long durations can be scheduled more easily. The influence of the model, described in chapter 3.4.1, can be seen at the blue bar in the case Single2. Based on the fact that at specific cycle time constellations the time window duration is reduced to avoid the influence on the bottleneck, the average time window duration declines compared to case Single1.

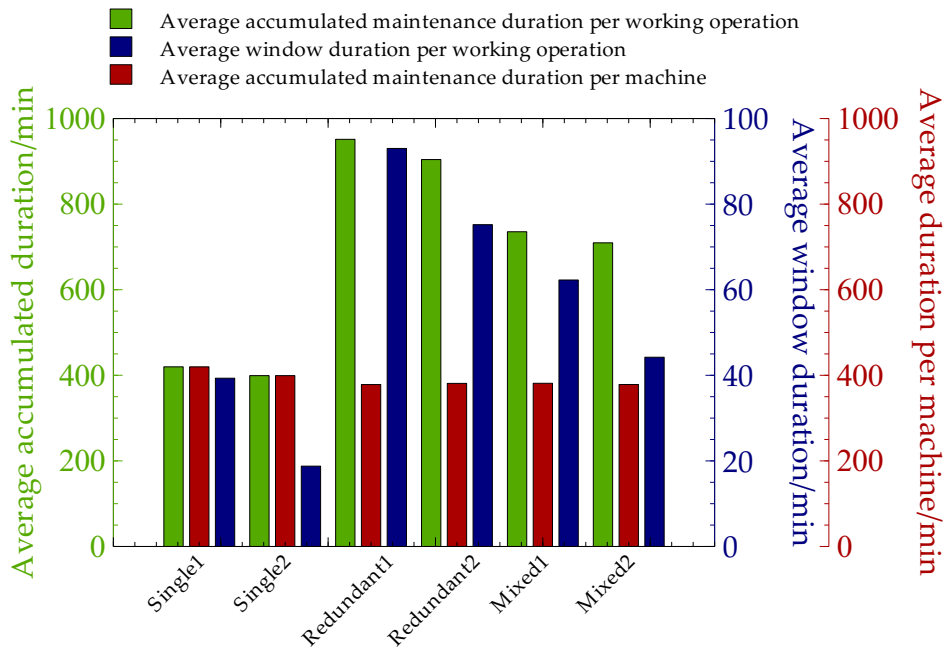


Figure 4.7: Comparison of average times of the production lines

Figure 4.8 shows the detailed comparison between the accumulated and average window duration for each machine and each case. The influence factors on these two values are the buffer size next to the machine, cycle times and whether the machine is redundant or not. The buffer capacity influence can be seen in the top left diagram of figure 4.8. As illustrated the buffer size of buffer B_5 is rather low compared to the other ones. Also the average window duration decreases in this case from working operation "AF5" to "AF6". The influence of the number of machines per working operation on the average window duration is, in the cases Redundant2 and Mixed1, higher compared to Redundant1 and Mixed1. The working operation "AF12", for example, has in case Redundant2, four

machines. In Redundant1 it has the fourth highest average window duration, in case Redundant2 the second highest. This is due to the fact that at production lines with mixed cycle times the flexible window does not strongly overlap through the machines. The influence of the cycle time differences and constellations can be seen in the diagram at the top right. AF13 has a cycle time of 46 seconds and AF14 of 38 seconds. As illustrated the average window durations is at the working operation AF14 somewhat higher.

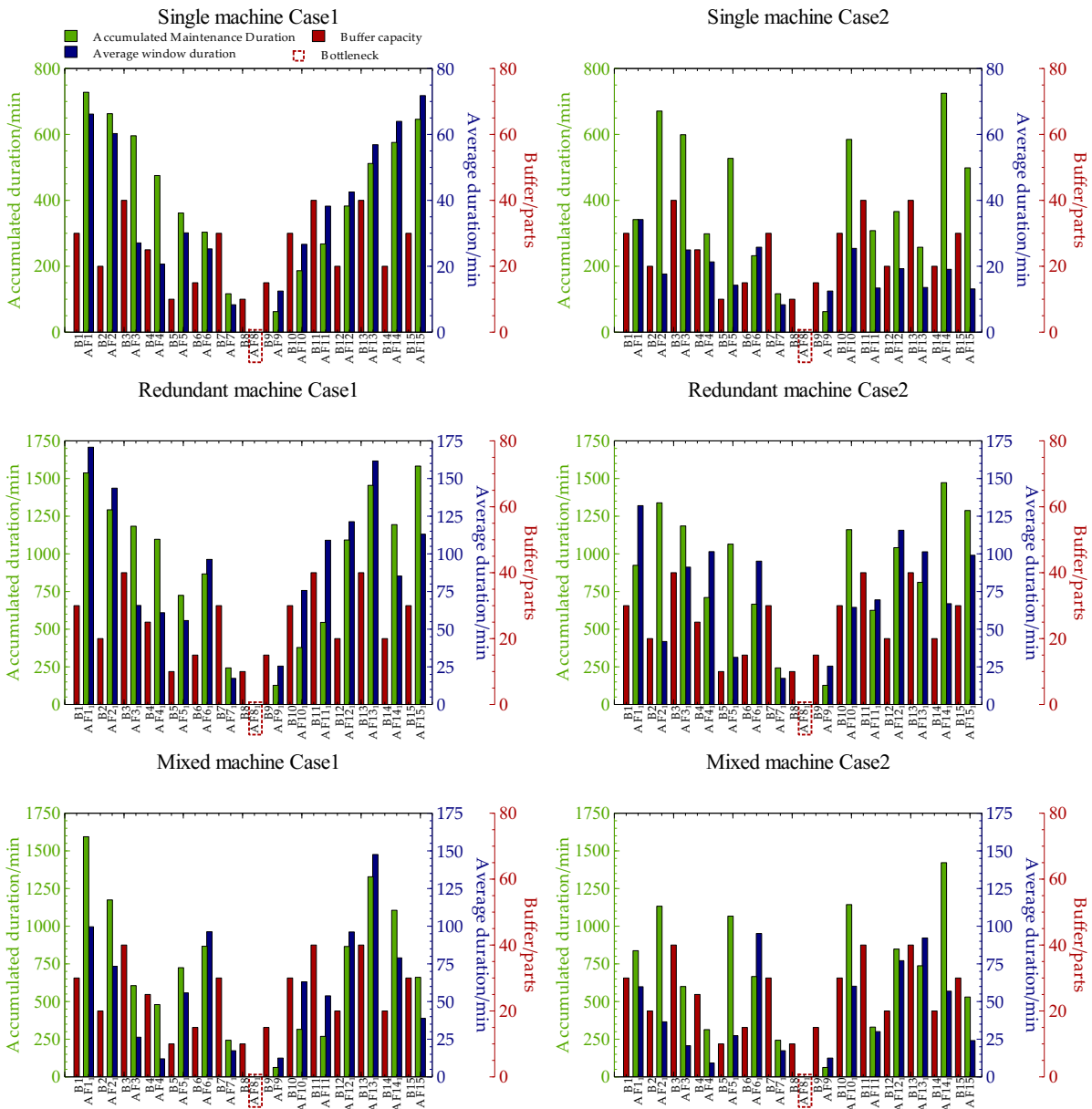


Figure 4.8: Comparison accumulated and average window durations

Figure 4.9 shows the production statistic of the case Mixed2. As shown at the redundant machine combinations, for example $AF12_1 \rightarrow AF12_4$, the algorithm calculates the time window for only one machine in the redundant combination. Corresponding to that just one machine is shut down for maintenance activities per point in time in a redundant combination and therefore only one machine of a combination in figure 4.9 contains a blue bar. At the other machines of the combinations blocking and waiting times. The green bar shows the production percentage on the time of every machine. The yellow bar is the blocking time and the red one the starving or waiting time. Generally speaking, prior to the bottleneck, only blocking times should occur and afterwards to it only waiting times should occur. Through different cycle time constellations this could change as shown in figure 4.9.

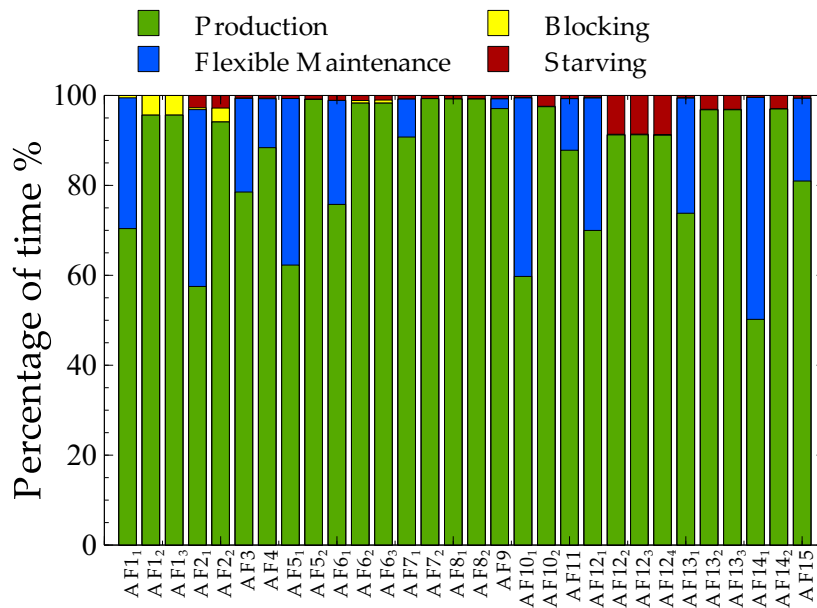


Figure 4.9: Utilization statistic single machine production line case2

A comparison of all cases regarding the production statistics is shown in figure 4.10. Horizontally, the diagrams compare the differences of the cycle time. Vertically, the differences of the different production lines can be compared. The top left corner of the diagram clearly shows that the time windows percentage increases according to the distance of the bottleneck. This is due to the fact that machines are shut down for maintenance in sequence of the bottleneck. Therefore, the window durations accumulate. The aforementioned case does not apply to the top right corner of this diagram, though. The reason for that circumstance is the different cycle time constellation. At the redundant machines the higher accumulated window duration can be seen with the number of redundant machines in combination. The diagram also clearly indicates that waiting or

blocking times at maintainable machines, in case of occurrence, directly have influence on the throughput. If no maintaining window is on a machine, starving and blocking times are regarded as a complete shut down of surrounding machines. This can be especially seen at the mixed production line where redundant combinations and single machines alternate since single machines can only produce on a standard production rate or not at all.

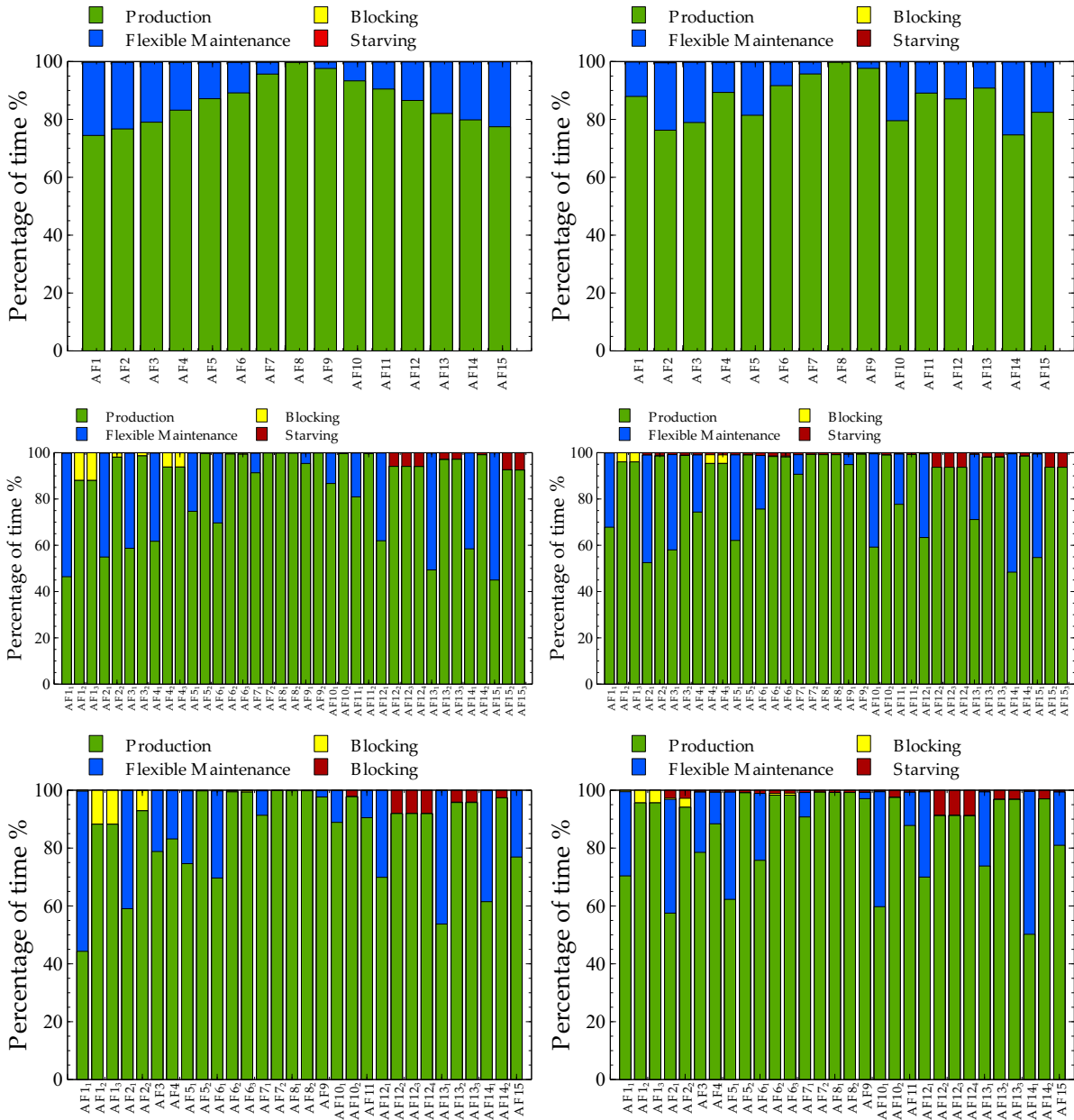


Figure 4.10: Comparison of production lines and cases regarding the production statistic

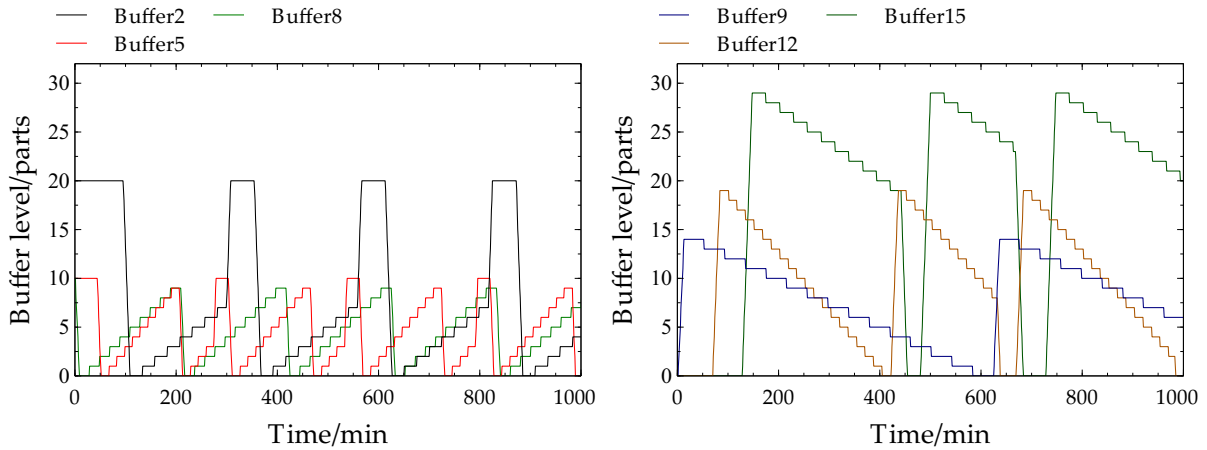


Figure 4.11: Buffer comparison

To see influences of buffer changes and how time windows occur over time, buffer levels were plotted at the algorithm and Plant Simulation[®] every minute. On the left-hand side figure 4.11 shows the buffers 2,5 and 8 previous to the bottleneck machine and on the right-hand side it shows buffers 9,12 and 15 subsequent to the bottleneck machine from the case Single1. The buffers eight and nine are next to the bottleneck machine. The graph shows that these two buffers are directly influenced at the beginning. Buffer eight drops to zero and buffer nine reaches its capacity. Based on the fact that the time windows of these two buffers are only influenced by the bottleneck machine, a constant cycle of time window creation occurs for the machines next to the bottleneck. A different behaviour can be seen at the other buffers. Buffer 5, for example, has two different time window creation cycles. One is shown from the minutes 50 to 220 and the other one from 220 to 320. These differences especially depend on the line constellation. Furthermore, the influence of the buffer capacity can be noticed. A high buffer capacity leads to long filling or draining cycles and therefore to some long flexible time windows. Small buffer capacities lead to the opposite, i.e. many short time windows.

Figure 4.12 illustrates the deviations of the buffer levels from the program compared to the plant simulation output. On the left hand side the case with the smallest buffer deviation is shown. This case occurred at the single machine production line case1 at buffer3. The mean deviation, observed for over two days, are 0.28 parts. As illustrated the program and Plant Simulation[®] output almost follow the same behaviour. The red differences shown on the left hand side arise by the discrete changes in Plant Simulation[®] over time and the recording of the data per minute. In the program the trend is also recorded per minute but the change is calculated continuously and rounded afterwards. On the right hand side the largest deviation is shown. This occurred at the mixed production

line with case2, also at buffer 3. At this case bigger differences at the trend are visible. These differences especially arise through the cycle times variations and the redundant machine combinations. Table 4.12 shows the buffer level deviations - in average over two days- for each buffer and each simulated production line. The last row shows the average for every case. It should be noted that the highest deviations are also at the cases with the lowest throughput. It may be concluded that these deviations have a significant influence on the overall output.

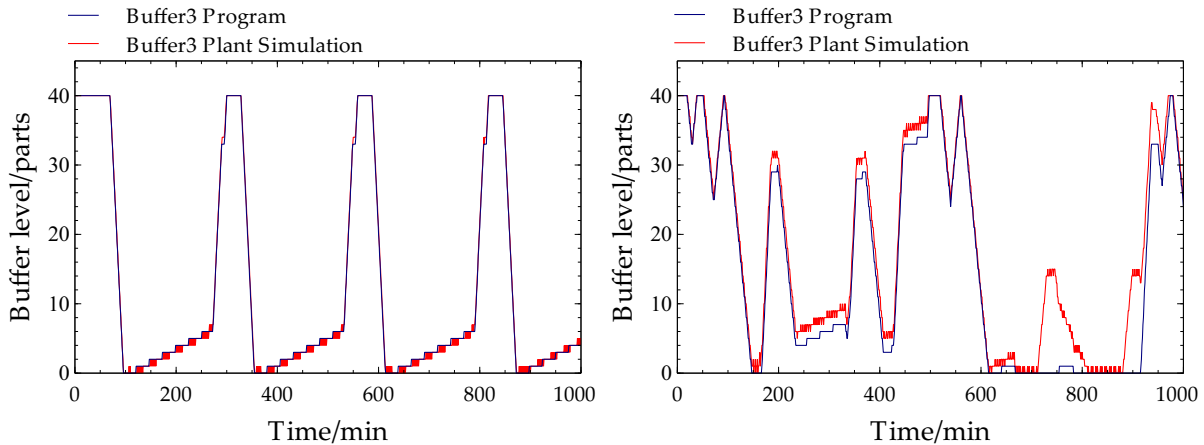


Figure 4.12: Buffer comparison program- Plant Simulation[®]

Finally, these results show that the algorithm is a valid option to abstract the problem. The production percentage amount is over 99% in every case and therefore only a small influence on the bottleneck are detected. Furthermore, the utilization of the buffer capacity μ was set to 100% in all cases. A decrease of this value would result in a decrease of the time window durations and in an increase of the production percentage. Moreover the negative impact of different cycle time constellations was shown. An increasing of cycle time in front of the bottleneck and a decreasing after it would be perfect for the algorithm. The fifteen working operations had no big influence on the algorithm. The lesser the number the better the output of the algorithm. The computation time for all production lines was with twelve seconds for two days nearly similar. If the planning horizon rises to one week the computation takes 30 seconds at the Single1 case.

Table 4.12: Buffer level mean deviation over two days

	Single1	Single2	Redundant1	Redundant2	Mixed1	Mixed2
B2	0.34	0.45	0.47	0.40	0.54	1.66
B3	0.28	0.33	0.49	0.49	0.69	2.79
B4	1.013	0.66	0.78	1.11	1.27	1.16
B5	0.42	0.35	0.53	0.48	0.52	0.83
B6	0.44	0.51	0.45	1.65	0.45	1.11
B7	0.69	0.74	0.60	0.98	0.53	1.63
B8	0.46	0.52	0.51	1.39	0.47	0.86
B9	0.84	0.90	0.55	1.70	0.49	1.18
B10	0.59	0.45	0.66	0.92	0.56	0.98
B11	0.56	1.02	0.57	0.89	0.64	0.91
B12	0.827	0.54	0.44	0.58	0.36	1.40
B13	1.195	1.18	0.72	0.81	0.77	1.46
B14	0.464	0.69	0.56	0.60	0.53	0.624
B15	1.39	0.56	0.36	0.52	0.68	0.86
Average	0.680	0.64	0.566	0.89	0.61	1.25

Chapter 5

Conclusion

According to the survey of Güntner et al. (2015), maintenance will change in the next decades, due to the new developments in the field of Industry 4.0. A paradigm shift from the traditional technical focused maintenance to a holistic maintenance will occur. Due to this trend, more researchers contribute to different topics of maintenance.

One of these fields of research is the opportunistic maintenance planning, which determines opportunities to carry out maintenance tasks during production shifts. This leads to an increase of the throughput. Chang et al. (2007) were among the first scientists who conducted investigations on this topic. They developed a model to calculate maintenance time windows during the production process with real time data. This model calculates a time window immediately from the time of the inquiry. No calculations of future opportunities, for example for one week, are made. The investigations on the topic of opportunity windows are currently reduced to a few papers.

The aim of this master's thesis was to develop an algorithm which enables flexible maintenance scheduling over a longer period of time. The algorithm has to fulfil two major tasks, the determination of opportunities, as stated, and the scheduling of predefined maintenance tasks on the windows. The two basic triggers for such a maintenance opportunity are a breakdown of a machine or a buffer level between two machines. The prediction of these time windows for a period of time has the benefit to enhance the previous planning of maintenance resources and therefore to enable the implementation of the algorithm on a real production line.

Based on the objective, the developed algorithm was split into two basic tasks, which were executed separately. The first one was the determination of the opportunity windows for every machine in the production line. For this part the machine-buffer-machine model

from Chang et al. (2007) was applied. With this basic formulation an algorithm was developed based on the simulation method of activity scanning. The second task was to schedule maintenance jobs on the determined flexible opportunity windows. An integer model of this problem was formulated and subsequently solved with the optimization software Gurobi. The necessary input parameters for the opportunity time window calculator were the buffer level, the buffer capacity, the cycle times and the machine structure. For the task scheduler a task list, time windows and employee data were required. The developed algorithm was restricted to a flow job production. Production lines with single and redundant machine combinations can be considered. The reliability of machines was not taken into account and therefore set to 100%. Point in times like due dates or start times were assumed to be deterministic and stochastic distributions were not considered.

A validation for the opportunity time window calculator was done with the software Plant Simulation[®]. Three different production lines, in each case with fifteen working operations, were set up to analyse the algorithm. In all cases the production rate of the bottleneck was higher than 99.15%. The determined average accumulated time window duration per machine amounted to approximately 400 minutes in two days. Although the cycle time differences and buffer capacities were chosen relatively high, an accumulated time window duration of one whole shift with lower parameters over one week is realistic. If all maintenance jobs were scheduled during these flexible time windows, the uptime and availability will increase significantly and consequently the throughput will be increased as well. Due to the fact that companies often have fixed maintenance shifts per week the benefit of the algorithm is that this period might be reduced to the duration of necessary maintenance efforts on the bottleneck machine. A brief case in the next section shows the benefit of the algorithm.

5.1 Exemplification Case

To illustrate the quantified benefit of the algorithm a brief case is calculated. As production line the case Single2 from chapter 4.2 is taken. It is assumed that all machines of the line require a maintenance effort of one hour per week. Half of the tasks require the skill level of the worker on the machine and the other half the skill level of the maintenance worker at the production line. The different throughputs per week can be seen in table 5.1. The cycle time of the bottleneck machine is 50 seconds per part. The throughput over one week without maintenance is 12096 for three shifts per day. However, maintenance is important to avoid unplanned breakdowns and a corresponding throughput loss. The

unplanned breakdowns are assumed to be zero. Usually, the maintenance is done in a fixed shift, for example on Monday morning. Due to the fact that the maintenance employee has to do half of all tasks at the machines, this would result in a downtime of 7.5 hours of the production line per week. As a result, the throughput decreases to around 95.5% and only 11556 parts/week could be produced. Nevertheless, the average time window duration and the lowest accumulated window duration is approximately 63 minutes. Therefore it can be assumed that all tasks of the production line, except the ones from the bottleneck, can be performed during these flexible time windows. This results in a maintenance duration for the fixed maintenance shift of one hour for the bottleneck machine. Consequently, the throughput is only reduced to 99.4% and 12024 parts/week can be produced. This means, that the utilization of these maintenance windows could increase the throughput by 4%. This leads to a significant performance increase while machine reliability stays at least the same. Furthermore, it gives a higher degree of flexibility to the maintenance workers to perform autonomous maintenance tasks, as claimed by the TPM concept.

Table 5.1: Throughput calculation

Planning technique	Fixed window duration/hours	Throughput/parts
No maintenance	0	12096
Fixed maintenance	7.5	11556
Flexible maintenance	1	12024

With this brief case the benefit is illustrated. In a field where every production loss should be avoided this algorithm is a possibility to increase the productivity. The last section discusses future possibilities of research and further developments of the algorithm.

5.2 Future Work

Further steps for the validation of the algorithm are experimental runs on a pilot line. The results should determine the performance of the algorithm in comparison to the real world. Throughput losses have to be plotted and it has to be examined which machine is responsible. The model could be extended by considering special logistic equipment or parallel line sections. Buffer levels can be tracked and set in contrast to the algorithm. Further steps would also take reliability, MTBF and MTTR of the machines into account. Production data in real time could be used to update the output, for example, per minute. As a result, a rolling horizon would arise which would precise the result.

Appendix

Appendix A

The figures A1, A2 and A3 explain the calculation of a time window for a redundant production line. As shown the line consists of one single machine in the front and three redundant machines next to it. The initial situation is in case one that the start cycle times, no machine is down, are in descending order $CT_{i-3} > CT_{start_{i-2}} > CT_{start_{i-1}} > CT_{start_i}$. Through that all redundant machines consists of two machines also the cycle times if one machine is down is descending from machine $i - 2$ to machine i $CThigh_{i-2} > CThigh_{i-1} > CThigh_i$. Case2 differs in the fact that machine $i - 1$ has a lower cycle time than machine i . This cases should show which influence differences in cycle times have on the duration Δt of a flexible window. The flexible window is calculated for machine i which is marked by the red frame.

The calculation of the time window for machine i starts through that buffer B_i gets empty at point time t_0 . As shown in figure A1 through the cycle times the machines $i - 2, 1$ and $i - 1, 1$ are already in a maintenance window from previous a point in time . As a consequence that the cycle time of machine $i - 1$ is higher than $CThigh_i$ machine i takes on $CThigh_{i-2}$. At point in time t_1 buffer B_{i-2} is full and the cycle time of $i - 2$ switch to $CT_{start_{i-2}}$ because the maintained machine $(i - 2), 1$ start again. The cycle times of machines $i - 1$ and i changes to $CThigh_{i-1}$ because $CThigh_{i-1} > CThigh_i$. At time point t_2 buffer B_{i-1} is full and machine $(i - 1), 1$ has to start again. Now machine i takes over its real $CThigh$ and runs till the buffer B_i is full at point in time t_c which is the same as the end of the time window twe_i for machine i .

The mathematical formulation for the duration of the time window is: $N(0) = 0; P_i = C_i\mu$

$$d_i = \frac{C_i\mu - \left(\frac{t_1-t_0}{CThigh_{i-2}} + \frac{t_2-t_1}{CThigh_{i-1}} - \frac{t_2}{CT_{start_{i-1}}} \right) + \frac{t_1-t_0}{CThigh_{i-2}} + \frac{t_2-t_1}{CThigh_{i-1}} - \frac{t_2}{CThigh_i}}{\frac{1}{CT_{start_{i-1}}} - \frac{1}{CThigh_i}}$$

Appendix

$$d_i = \frac{C_i\mu + t_2\left(\frac{1}{CT_{start_{i-1}}}\right) - \frac{1}{CT_{high_i}}}{\frac{1}{CT_{start_{i-1}}} - \frac{1}{CT_{high_i}}}$$

$$d_i = \frac{C_i\mu}{\dot{m}_{start_{i-1}} - \dot{m}_{high_i}} + t_2$$

In the second case the time window the time window duration of machine i differs to case one due to the lower CT_{high} of machine $i - 1$ compared to machine i . The first interval $t_0 \rightarrow t_1$ is the same as in case one. In the second one the machines $i - 1$ and i have already different cycle times, because $CT_{high_{i-1}} < CT_{high_i}$. Therefore the buffer level at buffer B_i starts to increase. At point in time t_c the buffer level is full and all machines produce again. The calculation of this case is the following:

$$d_i = \frac{C_i\mu - \left(\frac{t_1-t_0}{CT_{high_{i-2}}} + \frac{t_2-t_1}{CT_{high_{i-1}}} - \frac{t_2}{CT_{start_{i-1}}}\right) + \frac{t_1-t_0}{CT_{high_{i-2}}} + \frac{t_2-t_1}{CT_{high_i}} - \frac{t_2}{CT_{high_i}}}{\frac{1}{CT_{start_{i-1}}} - \frac{1}{CT_{high_i}}}$$

$$d_i = \frac{C_i\mu + (t_2 - t_1)\left(\frac{1}{CT_{high_i}} - \frac{1}{CT_{high_{i-1}}}\right) + t_2\left(\frac{1}{CT_{start_{i-1}}} - \frac{1}{CT_{high_i}}\right)}{\frac{1}{CT_{start_{i-1}}} - \frac{1}{CT_{high_i}}}$$

$$d_i = \frac{C_i\mu + (t_2 - t_1)(\dot{m}_{high_i} - \dot{m}_{high_i} - 1)}{\dot{m}_{start_{i-1}} - \dot{m}_{high_i}} + t_2$$

Through this equation the difference to case one can be seen. The available buffer level for the last interval decreases by the term $(t_2 - t_1)(\dot{m}_{high_i} - \dot{m}_{high_i} - 1)$ because $\dot{m}_{high_i} < \dot{m}_{high_i} - 1$. This change can also be seen in figure A3 by the comparison of the two cases.

Case1 of redundant production line

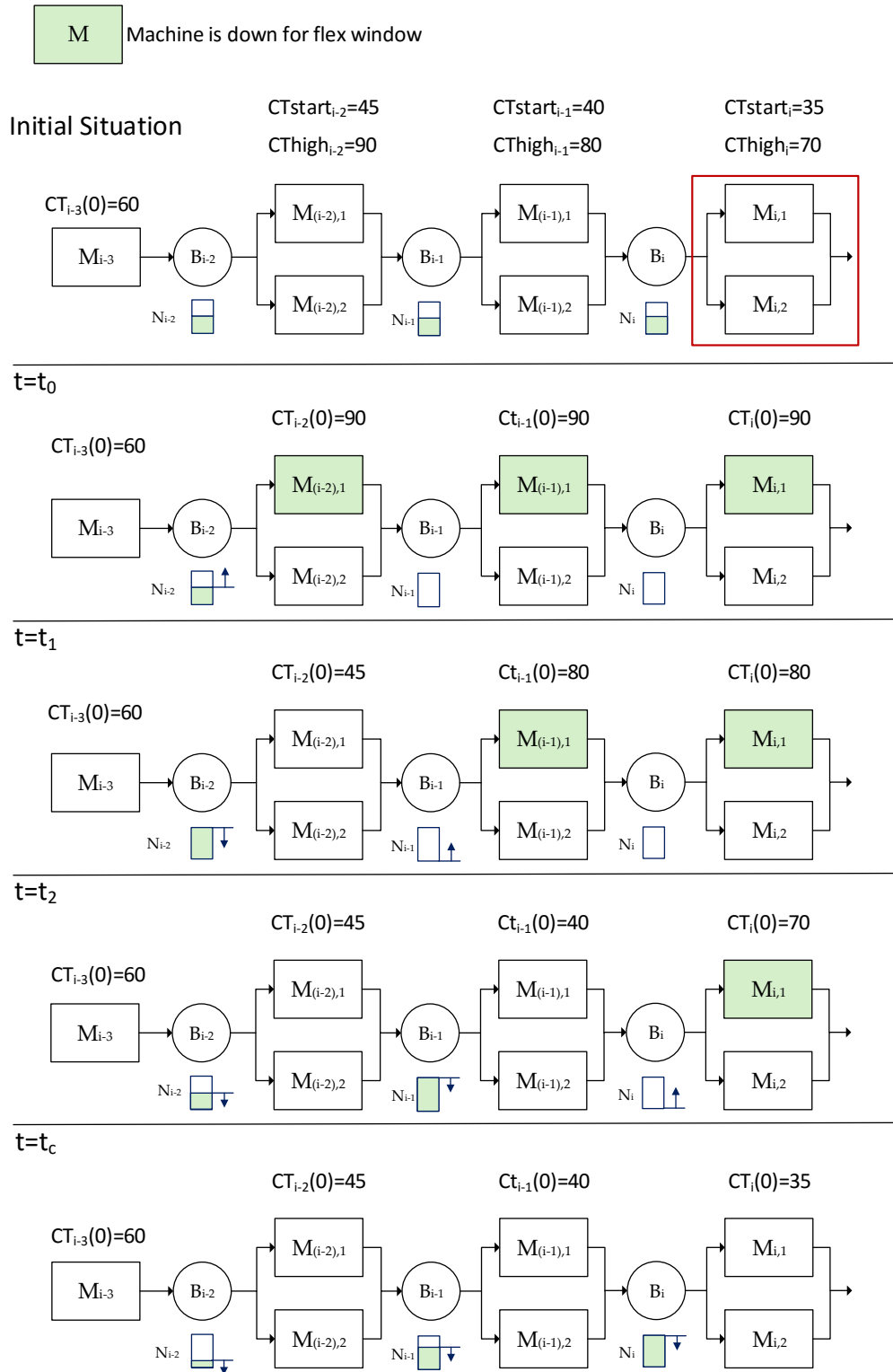


Figure A1: Redundant production line Case1

Case2 of redundant production line

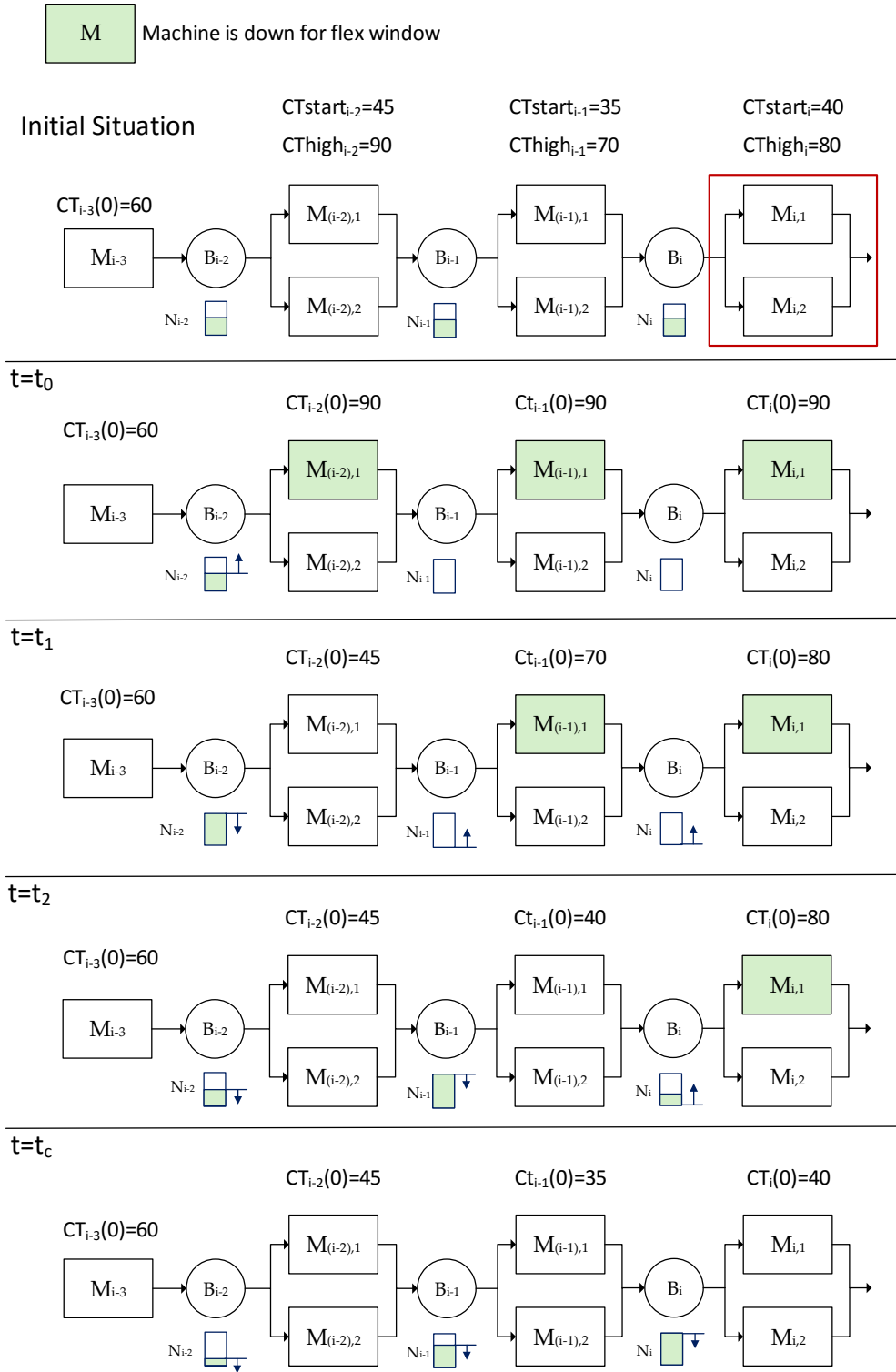


Figure A2: Redundant production line Case2

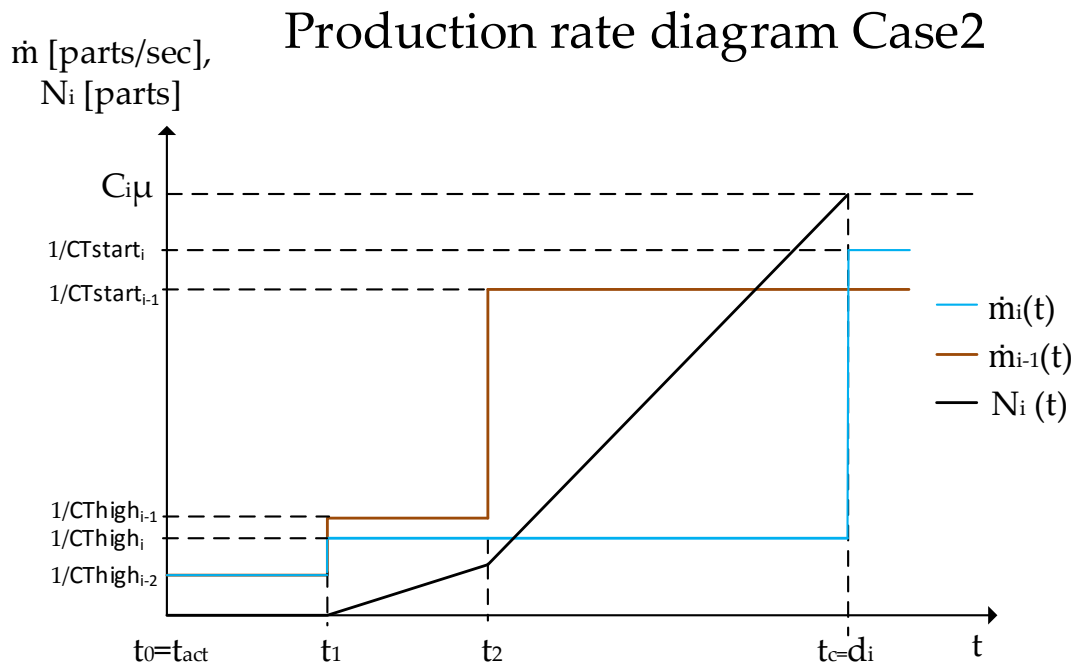
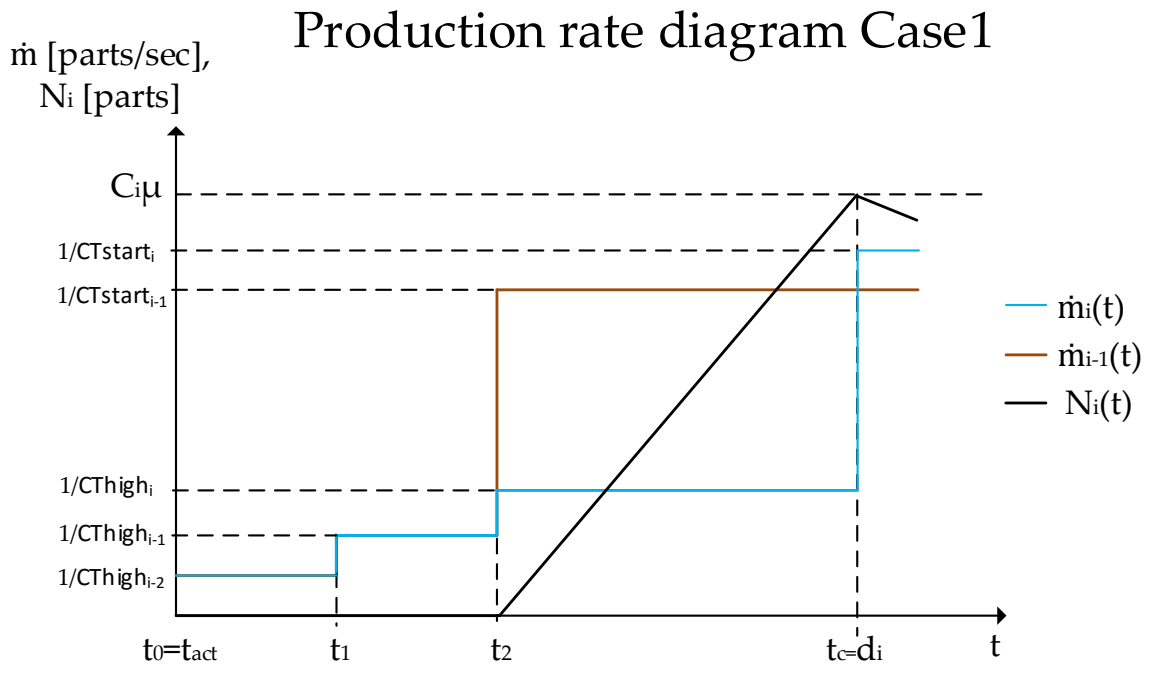


Figure A3: Production rate diagram for redundant production line cases

Appendix B
ERM-model

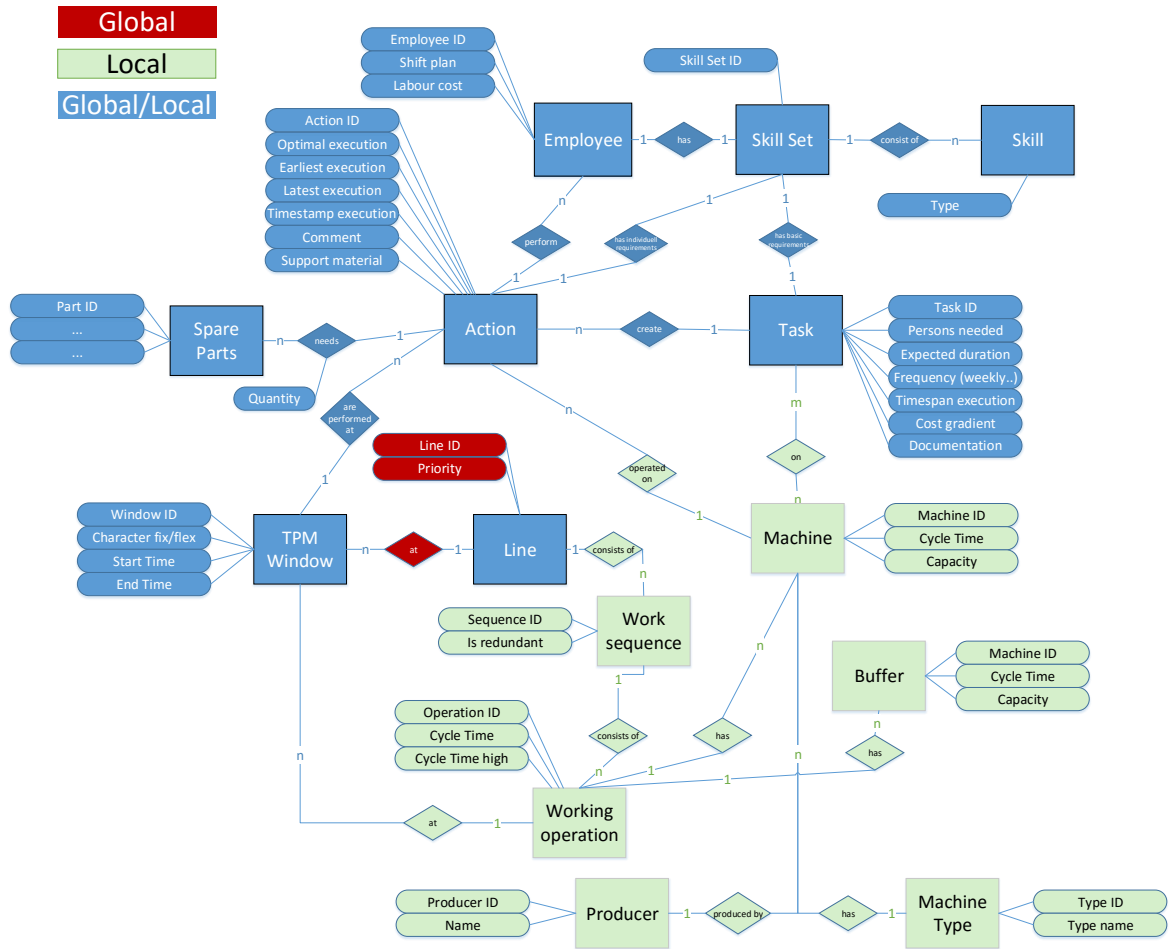


Figure B1: ERM model

Appendix C

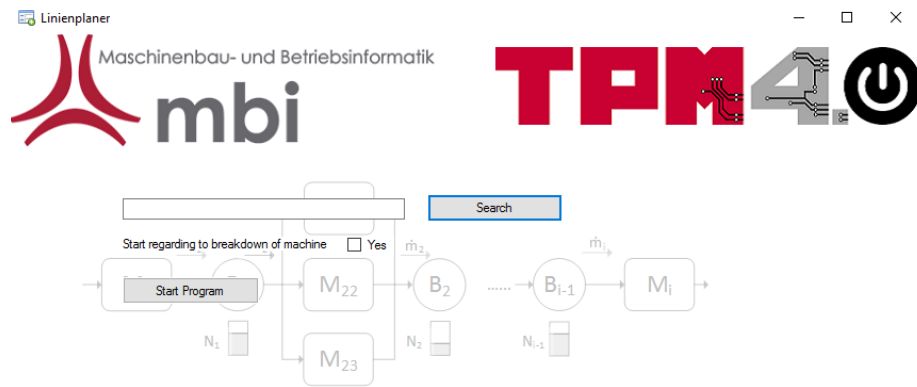


Figure C1: Main window program

Appendix D

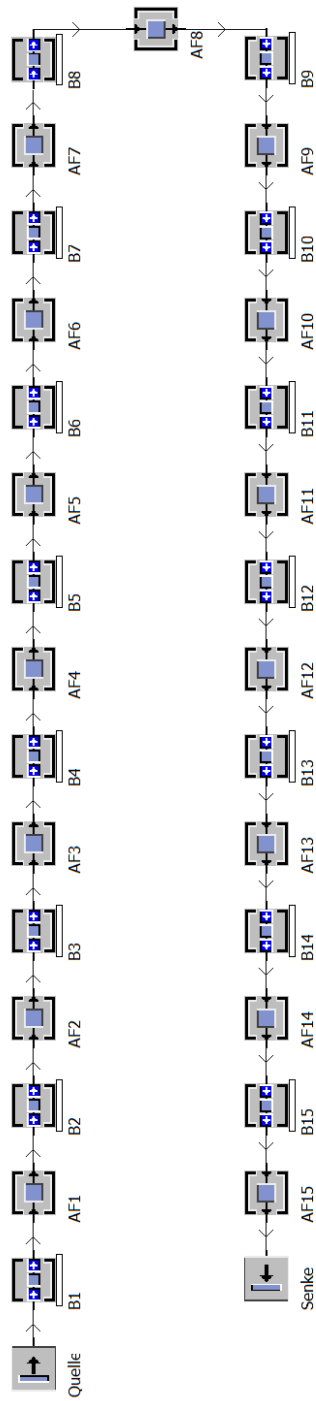


Figure D1: Single machine production line in Plant Simulation

Appendix

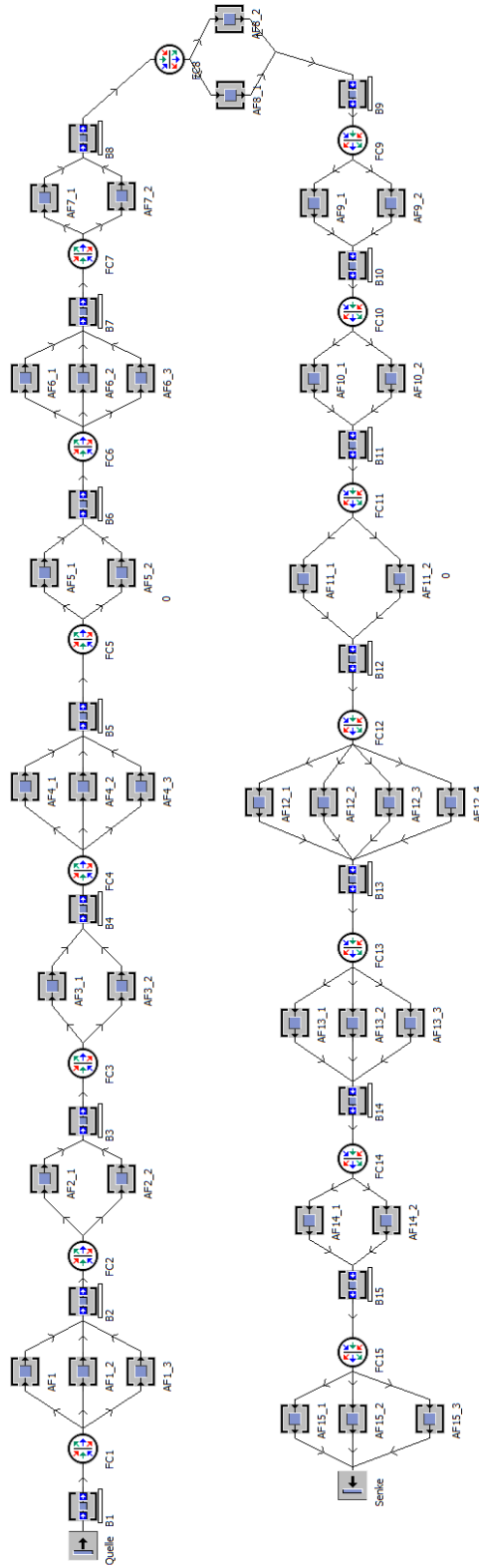


Figure D2: Parallel production line

Appendix

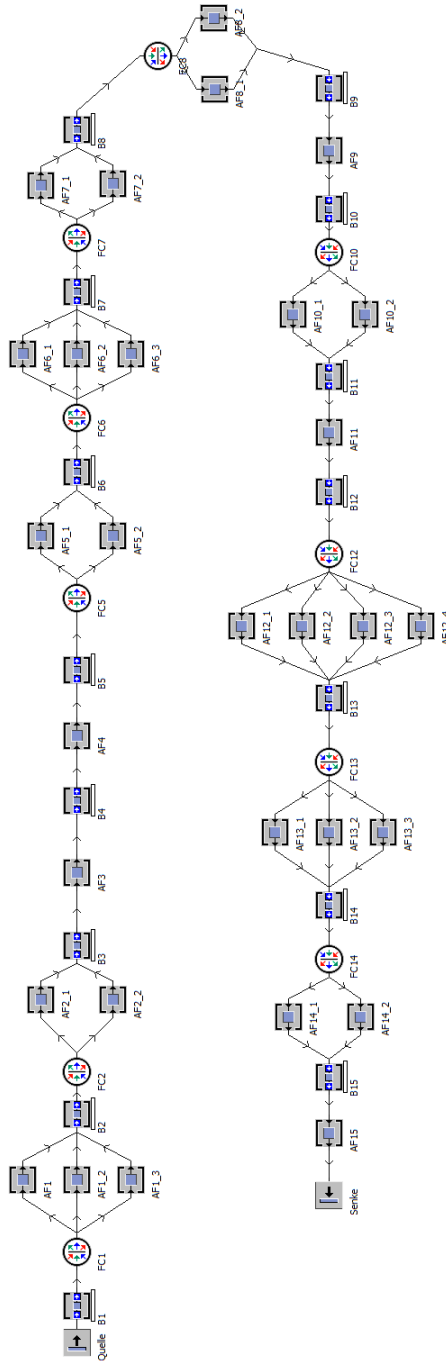


Figure D3: Mixed production line in Plant Simulation

Bibliography

- Ahuja, I.P.S. and J.S. Khamba (2008). “Total productive maintenance: literature review and directions”. In: *International Journal of Quality & Reliability Management* 25.7, pp. 709–756. DOI: 10.1108/02656710810890890 (cit. on pp. 14–16).
- Alrabghi, Abdullah and Ashutosh Tiwari (2015). “State of the art in simulation-based optimisation for maintenance systems”. In: *Computers & Industrial Engineering* 82, pp. 167–182. ISSN: 0360-8352 (cit. on p. 37).
- Baker, Kenneth R. and Dan Trietsch (2009). *Principles of Sequencing and Scheduling*. Hoboken, New Jersey: John Wiley & Sons, Inc. ISBN: 9780470391655 (cit. on p. 17).
- Balci, Osman (1988). “The Implementation of Four Conceptual Frameworks for Simulation Modeling in High-level Languages”. In: *Proceedings of the 20th Conference on Winter Simulation*. WSC '88. San Diego, California, USA: ACM, pp. 287–295. ISBN: 0-911801-42-1 (cit. on pp. 35, 36).
- Bandow, Gerhard and Friedrich-Wilhelm Schaefer (2009). “Ganzheitliche Instandhaltung: Strukturen und Strategien”. In: Früh, K. F. et al. *Handbuch der Prozessautomatisierung: Prozessleittechnik für verfahrenstechnische Anlagen*. 4th ed. München: Oldenbourg Industrieverlag. ISBN: 383563142X (cit. on p. 13).
- Banks, Jerry et al. (2004). *Discrete-Event System Simulation*. 4th ed. Upper Saddle River, New Jersey: Pearson Education, Incorporated. ISBN: 0131446797 (cit. on pp. 27, 28, 31, 33, 43, 68, 77).
- Bikash, Bhadury (1998). *Total Productive Maintenance*. Allied Publishers Ltd. ISBN: 81-7023-805-6 (cit. on p. 13).

Bibliography

- Birta, Louis G. and Gilbert Arbez (2013). *Modelling and Simulation: Exploring Dynamic System Behaviour*. 2nd ed. London: Springer-Verlag. ISBN: 978-1-4471-2782-2 (cit. on p. 29).
- Borshchev, Andrei and Alexei Filippov (2004). “From System Dynamics and Discrete Event to Practical Agent Based Modeling: Reasons, Techniques, Tools”. In: *22nd International Conference of the System Dynamics Society, 25-29 July 2004*, p. 45. ISSN: 07498063 (cit. on p. 32).
- Canto, Salvador Perez (2011). “Using 0/1 mixed integer linear programming to solve a reliability-centered problem of power plant preventive maintenance scheduling”. In: *Optimization and Engineering* 12.3, pp. 333–347. ISSN: 13894420 (cit. on p. 26).
- Cassady, C.R. et al. (2005). “Integrating Preventive Maintenance Planning and Production Scheduling for a Single Machine”. In: *IEEE Transactions on Reliability* 54.2, pp. 304–309. ISSN: 0018-9529 (cit. on p. 26).
- Chang, Qing et al. (2007). “Maintenance Opportunity Planning System”. In: *Journal of Manufacturing Science and Engineering* 129.3, pp. 661–668. ISSN: 1087-1357 (cit. on pp. 2, 38, 46, 53, 89, 90).
- Conway, Richard W. et al. (1967). *Theory of scheduling*. Reading: Addison-Wesley Publishing Company (cit. on p. 18).
- Dantzig, George B. (1966). *Lineare Programmierung und Erweiterungen*. Berlin Heidelberg: Springer-Verlag. ISBN: 978-3-642-87362-1 (cit. on pp. 19, 22, 59).
- Dantzig, George B. and Mukund N. Thapa (1997). *Linear Programming 1: Introduction*. New York: Springer-Verlag. ISBN: 978-0-387-22633-0 (cit. on pp. 20, 21).
- DIN31051:2012-09 (2012). *Grundlagen der Instandhaltung* (cit. on p. 6).
- Ebeling, Charles E. (1997). *An Introduction to Reliability and Maintainability Engineering*. Boston: McGraw-Hill Companies, Inc. ISBN: 0-07-018852-1 (cit. on pp. 8–10).
- Ebrahimipour, V. et al. (2013). “Multi-objective modeling for preventive maintenance scheduling in a multiple production line”. In: *Journal of Intelligent Manufacturing* 26.1, pp. 111–122. ISSN: 15728145 (cit. on p. 26).
- Eichler, Christian (1990). *Instandhaltungstechnik*. 5th ed. Berlin: Verlag Technik GmbH. ISBN: 3-341-00667-2 (cit. on pp. 11, 12).

Bibliography

- Gu, Xi et al. (2013). “Hidden maintenance opportunities in discrete and complex production lines”. In: *Expert Systems with Applications* 40.11, pp. 4353–4361. ISSN: 0957-4174 (cit. on p. 38).
- Güntner, Georg et al. (2015). *Roadmap der Instandhaltung 4.0*. URL: <http://www.salzburgresearch.at/wp-content/uploads/2015/05/IH40-Roadmap-final.pdf> (visited on 04/05/2016) (cit. on pp. 2, 89).
- Harper, Douglas (2016). URL: <http://www.etymonline.com/index.php?term=maintenance> (visited on 11/06/2016) (cit. on p. 5).
- Hedtstück, Ulrich (2013). *Simulation diskreter Prozesse: Methoden und Anwendungen*. Berlin Heidelberg: Springer. ISBN: 978-3-642-34871-6 (cit. on pp. 34, 35).
- Hiller, Frederick S. and Gerald J. Liebermann (2001). *Introduction to operations research*. 7th ed. McGraw-Hill. ISBN: 0072321695 (cit. on pp. 23, 24).
- Lee, Seungchul et al. (2013). “Stochastic maintenance opportunity windows for unreliable two-machine one-buffer system”. In: *Expert Systems with Applications* 40.13, pp. 5385–5394. ISSN: 0957-4174 (cit. on p. 38).
- Manzini, Riccardo et al. (2015). “The scheduling of maintenance. A resource-constraints mixed integer linear programming model”. In: *Computers & Industrial Engineering* 87, pp. 561–568. ISSN: 03608352 (cit. on p. 25).
- Mobley, R. Keith (2004). “1 - Impact of Maintenance”. In: *Maintenance Fundamentals (Second Edition)*. Ed. by R. Keith Mobley. Second Edition. Plant Engineering. Burlington: Butterworth-Heinemann, pp. 1–10. ISBN: 978-0-7506-7798-1 (cit. on p. 12).
- Moghaddam, Kamran S. and John S. Usher (2011). “Preventive maintenance and replacement scheduling for repairable and maintainable systems using dynamic programming”. In: *Computers and Industrial Engineering* 60.4, pp. 654–665. ISSN: 03608352 (cit. on p. 26).
- Nakajima, Seiichi (1988). *Introduction to TPM: Total Productive Maintenance*. Productivity Press, Inc. ISBN: 0-915299-23-2 (cit. on pp. 8, 9, 13–17).
- Nakajima, Seiichi (1995). *Management der Produktionseinrichtungen: Total Productive Maintenance*. German. Frankfurt, Main [u.a.]: Campus-Verlag. ISBN: 9783593351643 (cit. on p. 1).

Bibliography

- Nemhauser, George and Laurence Wolsey (1999). *Integer and Combinatorial Optimization*. John Wiley & Sons, Inc. ISBN: 0-471-82819-X (cit. on pp. 21, 24).
- Neubacher, D. et al. (2016). “A Hierarchical Control Simulation Model to Support Maintenance Planning in Flexible Production Systems”. In: *Proceedings of the 2016 European Simulation and Modelling Conference* (cit. on p. 38).
- Oxford Dictionary (2016a). *Definition scheduling*. URL: <http://www.oed.com/view/Entry/180009?redirectedFrom=Simulation#eid> (visited on 09/01/2016) (cit. on p. 27).
- Oxford Dictionary (2016b). *Definition simulation*. URL: <https://en.oxforddictionaries.com/definition/schedule> (visited on 09/03/2016) (cit. on p. 17).
- Pinedo, Michael (2002). *Scheduling: theory, algorithms, and systems*. English. 2. Upper Saddle River, New York: Prentice-Hall. ISBN: 0130281387 (cit. on p. 18).
- Plattform Industrie 4.0 (2016). *Definition Industry 4.0*. URL: <http://www.plattform-i40.de/I40/Navigation/DE/Industrie40/WasIndustrie40/was-ist-industrie-40.html> (visited on 10/30/2016) (cit. on p. 2).
- Renkes, Dieter (1993). *Instandhaltungsmanagement Fachmann*. Eschborn: Rationalisierungskuratorium der Deutschen Wirtschaft. ISBN: 3-929796-07-4 (cit. on p. 7).
- Robinson, Stewart (2013). “Conceptual Modeling For Simulation”. In: *Winter Simulation Conference*, pp. 342–353 (cit. on pp. 28, 43, 44, 67).
- Stamatis, D. H. (2010). *The OEE Primer: Understanding Overall Equipment Effectiveness, Reliability, and Maintainability*. New York: Taylor and Francis Group, LLC. ISBN: 978-1-4398-1406-2 (cit. on pp. 9, 10, 15, 16).
- Strunz, Matthias (2012). *Instandhaltung: Grundlagen - Strategien - Werkstätten*. Berlin Heidelberg: Springer-Verlag. ISBN: 9783642273896 (cit. on pp. 7, 11, 12).
- Van Horenbeek, Adriaan et al. (2010). “Maintenance optimization models and criteria”. In: *International Journal of Systems Assurance Engineering and Management* 1.3, pp. 189–200. ISSN: 09756809 (cit. on pp. 2, 25, 43).
- Wainer, Gabriel A. (2009). *Discrete-Event Modeling and Simulation: A Practitioner’s Approach*. Boca Raton: Taylor & Francis Group. ISBN: 978-1-4200-5336-4 (cit. on pp. 32, 33).

Bibliography

- Weißbach, Andreas (2012). “Verbundinstandhaltung bei Kleinstunternehmen, kleinen und mittleren Unternehmen (KMU) - Ein Konzept für neue Organisationsformen der Instandhaltung”. PhD thesis. Universitätsverlag Ilmenau (cit. on p. 1).
- Wildeman, R.E. et al. (1997). “A dynamic policy for grouping maintenance activities”. In: *European Journal of Operational Research* 99.3, pp. 530–551. ISSN: 03772217 (cit. on p. 26).
- Willmott Peter; McCarthy, Dennis (2001). *TPM - A Route to World-Class Performance*. Oxford: Butterworth-Heinemann. ISBN: 0 7506 4447 8 (cit. on p. 15).
- Zeigler, Bernard P. (1976). *Theory of modelling and simulation*. New York: Wiley Interscience. ISBN: 0471981524 (cit. on p. 34).