

Tobias Elmer

Dynamic task prioritization in maintenance management

Master Thesis

Graz University of Technology

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

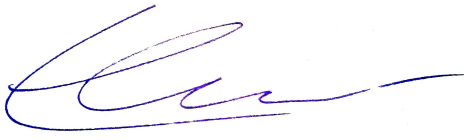
Supervisor:
Dipl.-Ing. Clemens Gutsch
Dipl.-Ing. Dr.techn. Nikolaus Furian

Graz, April 2017

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.

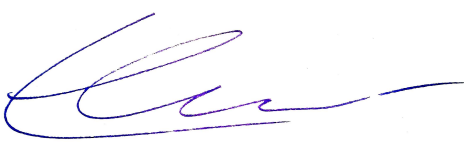
Graz, 13 April 2017
Date


Signature

Eidesstattliche Erklärung

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 entnommene Stellen als solche kenntlich gemacht habe.

Graz, am 13 April 2017
Datum


Unterschrift

Danksagung / Acknowledgement

Hiermit möchte ich all jenen danken, die mich im Rahmen meines Studiums sowie beim Verfassen dieser Masterarbeit unterstützt und begleitet haben. Ich bedanke mich bei meinen Betreuern Clemens Gutschi, Nikolaus Furian und Dietmar Neubacher für ihre fachliche und persönliche Unterstützung. Besonderer Dank gebührt außerdem meinen Eltern Bernhard und Margot, die mir durch ihre Unterstützung mein Studium ermöglicht haben.

I would like to express my gratitude to the people, who accompanied me during my studies and supported me while writing this thesis. I want to thank my supervisors Clemens Gutschi, Nikolaus Furian and Dietmar Neubacher for their professional and personal support. Finally, let me convey my special thanks to my parents Bernhard and Margot who made my graduation possible through their support.

Kurzfassung

Durch die zunehmende Globalisierung und dem damit verbundenen verschärften Wettbewerb, stehen produzierende Unternehmen unter ständig steigendem Kostendruck. Um diesem Kostendruck gerecht zu werden, versuchen Unternehmen lohnintensive Herstellungsprozesse durch solche zu ersetzen, welche aufgrund eines hohen Automatisierungsgrades einen geringeren Lohnanteil aufweisen. Der höhere Automatisierungsgrad impliziert wiederum eine zunehmende Komplexität und höhere Kosten der Produktionsanlagen, weshalb das Instandhaltungsmanagement zunehmend in den Fokus der Unternehmensführung rückt. Effiziente Planung von präventiven Instandhaltungsmaßnahmen, sowie eine rasche Behebung von Maschinenausfällen sind notwendig, um eine hohe Anlagenverfügbarkeit zu erzielen. Im Falle simultaner Maschinenausfälle und begrenzter Personalressourcen können Situationen entstehen, in welchen eine Priorisierung von Anlagen notwendig ist. Während diese Priorisierung heute meist im Rahmen der subjektiven Einschätzung des Wartungspersonals durchgeführt wird, soll im Rahmen dieser Arbeit eine objektive und echtzeitdatenbasierte Priorisierungsmethode entwickelt werden. Hierzu wurden zwei Bottleneck-Identifizierungsmethoden - die *Active Period Method* und die *Blockage & Starvation Probability Method* - hinsichtlich ihrer Validität und ihrer Eignung für die Priorisierung von Anlagen untersucht. Weiters wurde im Rahmen einer Simulationsstudie das Potential für die Priorisierung von reaktiven Instandhaltungsmaßnahmen ermittelt. Die besten Ergebnisse bezüglich Validität, Usability, wie auch Performance wurden mit der *Active Period Method* erzielt. Im Rahmen der Simulation einer komplexen Produktionslinie eines Motorenwerks konnte durch Priorisierung von reaktiven Wartungsaufgaben eine Steigerung der Produktionsleistung von 5,9 %, verglichen zu einer FIFO Strategie, erzielt werden.

Abstract

In today's globalized markets, manufacturing companies have to increase their operational effectiveness and revenues while reducing their operating costs in order to compete successfully. This has triggered a development within manufacturing industries, where companies try to substitute labour-intensive production processes through automated processes. An increasing degree of automation implies machines with greater complexities which in turn require more maintenance than conventional machines. Due to these factors, maintenance has become a prime focus of industries. Efficient planning of preventive maintenance tasks, and a quick respond in the case of a machine breakdown are crucial tasks in maintenance management. In the field of corrective maintenance, situations arise, where there are more machine breakdowns, than there are maintenance workers available. In most companies, the prioritization of machines is done based on the subjective assessment of the maintenance staff. This thesis aims for providing a real-time data-driven prioritization method which can then be used as a decision support in the case of simultaneous breakdowns of machines. Therefore, two different bottleneck detection algorithms were applied, validated and assessed concerning their potential for prioritization in corrective maintenance. The two algorithms are based on the *Active Period Method* and the *Blockage & Starvation Probability Method*. The validation showed, that the Active Period Method was capable of identifying the primary, secondary and tertiary bottleneck of a system reliably, whereas the Blockage & Starvation Probability Method could only detect the primary bottleneck. Furthermore, the Active Period Method algorithm was modified for the usage as a prioritization method for corrective maintenance tasks. The performance of the algorithm was evaluated on a highly automated production line using discrete event simulation. Compared to a FIFO service policy, the prioritization policy using the Active Period Method brought a throughput increment of 5.9%.

Table of contents

1	Introduction	1
1.1	Problem Definition	2
1.2	Goals	3
2	Theory	4
2.1	Maintenance	4
2.1.1	Terms and Definitions	4
2.1.2	Objectives and Costs of Maintenance	6
2.1.3	Key Performance Indicators in Maintenance Management	8
2.1.4	Maintenance Strategies	9
2.1.5	Total Productive Maintenance	13
2.1.6	IT-Systems in Maintenance Management	16
2.2	Prioritization policies	18
2.2.1	First-In, First-Out	19
2.2.2	Heuristic Prioritization Policies	19
2.2.2.1	Part-Out-Part-Out Time	20
2.2.2.2	Availability	21
2.2.2.3	Redundancy	22
2.2.2.4	Comparison of the Heuristic Methods	23
2.2.3	Bottleneck Prioritization Policies	24
2.2.3.1	Blocking & Starvation Probability	24
2.2.3.2	Active period method	31
2.2.3.3	System Sensitivity Analysis	35
2.2.4	Comparison of Bottleneck Detection Methods	36
2.3	Simulation	39
2.3.1	Introduction to Simulation	40
2.3.2	Simulation Modelling	42

3	Methods & Implementation	46
3.1	Bottleneck Detection Algorithms	46
3.1.1	Active Period Method	46
3.1.2	Blocking & Starvation Probability	49
3.2	Simulation	50
3.2.1	Development of the Simulation Study	50
3.2.2	Simulation Scenarios	57
4	Use Cases	59
4.1	Demo Use Case	60
4.1.1	System Sensitivity Analysis	61
4.1.2	Validation of the Active Period Method	63
4.1.3	Validation of the Blocking & Starvation Probability Method	65
4.1.4	Comparison of the Bottleneck Detection Methods	67
4.2	Industrial Use Case 1	68
4.2.1	Heuristic Prioritization	70
4.2.1.1	Prioritizing using Part-Out-Part-Out Times	70
4.2.1.2	Prioritizing using Availabilities	72
4.2.1.3	Prioritizing using Redundancy & Availability	74
4.2.1.4	Comparison of Heuristic Methods	75
4.2.2	Bottleneck Prioritization	76
4.2.2.1	Active Period Method	76
4.2.2.2	Blocking & Starvation Probability Method	83
4.2.2.3	Conclusion – Blockage & Starvation Probability Method	88
4.3	Industrial Use Case 2	88
4.3.1	Variation of Input Parameters	90
5	Discussion	93
5.1	Comparison of Prioritization Policies	93
5.2	Limitations for Prioritization in Corrective Maintenance	96
5.3	Conclusion	97

6 Outlook	98
References	100

List of Figures

Figure 1: Situation on the plant floor which requires prioritization	2
Figure 2 Optimum for total maintenance costs	7
Figure 3: Excerpt of KPIs for maintenance management	8
Figure 4: Maintenance strategies	9
Figure 5: Activities during a machine failure	10
Figure 6: Calculation of the OEE	14
Figure 7: The five pillar model of TPM	14
Figure 8: Histogram of measured part-out-part-out times	21
Figure 9: Failure sensitivity simulation setup	22
Figure 10: Failure sensitivity of redundant machines	23
Figure 11: Bottleneck identification using BSP	25
Figure 12: Examples of multiple bottlenecks	26
Figure 13: Trend of blocking and starvation in a serial production line	28
Figure 14: Transformation of a concurrent station into a virtual machine	29
Figure 15: Feedback loop in a serial line production	30
Figure 16: Branch in a serial production line	31
Figure 17 Active Periods of a machine	32
Figure 18: Shifting bottlenecks	34
Figure 19: Average bottleneck over a period of time	34
Figure 20: Methods to study a system	40
Figure 21: Major approaches in simulation modelling	43
Figure 22: Main elements of the HCCM framework	51
Figure 23: Time to repair in context of the simulation study	55
Figure 24: Introduction of a Priority Increase Factor	56
Figure 25: Structure of the Demo production line	60
Figure 26: Bottleneck ranking of the Demo line - System sensitivity analysis	62
Figure 27: Shifting bottlenecks of the Demo production line - APM	64

Figure 28: Comparison of the priority ranking of the active period method and the system sensitivity analysis _____	64
Figure 29: Shifting bottlenecks using the BSP method _____	66
Figure 30: Comparison of the BSP method to the system sensitivity method _____	66
Figure 31: Bottleneck ranking of the Demo line using three different detection methods _____	67
Figure 32: Structure of Line 1 _____	68
Figure 33: Priority ranking using Part-Out-Part-Out times _____	71
Figure 34: Average throughput per day - FIFO vs. Part-Out-Part-Out Time _____	71
Figure 35: Priority ranking using availability values _____	73
Figure 36: Average throughput per day - FIFO vs. Availability _____	73
Figure 37: Priority ranking using availability values with decreased priorities for redundant machines _____	74
Figure 38: Average throughput per day - FIFO vs. Availability & Redundancy _____	75
Figure 39: Average throughput increment of heuristic methods compared to a FIFO policy _____	75
Figure 40: Excerpt of the priorities over a period of 3 days using the AAP method for Line 1 _____	77
Figure 41: Average throughput per day - FIFO vs. APM (48 Hours) _____	78
Figure 42: Average throughput increment for different periods of observation _____	79
Figure 43: Static AAP priorities before redundant machines priorities are reduced _____	80
Figure 44: Static AAP priorities after redundant machines priorities are reduced _____	81
Figure 45: Average throughput per day FIFO vs. Static APM _____	81
Figure 46: Average throughput increment of different <i>pinc</i> using a static AAP priority ranking _____	82
Figure 47: Average throughput increment for different input parameters on Line 1 compared to a FIFO service policy _____	83
Figure 48: Excerpt of the priority over time using the BSP method _____	84
Figure 49: Average throughput per day - FIFO vs. BSP (7 Days) _____	85
Figure 50: Static priorities of the BSP method before redundancies are reduced _____	86

Figure 51: Static priorities of the BSP method after redundant machine's priorities are reduced _____	87
Figure 52: Average throughput increment and 95% confidence intervals for different periods of observation for the BSP method compared to a FIFO service policy _____	87
Figure 53: Structure of Line 2 _____	88
Figure 54: Average throughput increment for different input parameters on Line 2 compared to a FIFO service policy _____	91
Figure 55: Static AAP priority ranking for Line 2 after redundant machine's priority was reduced _____	91
Figure 56: Simultaneous breakdowns Line 1 - Line 2 _____	92
Figure 57: Performance of each prioritization policy for Line 1 compared to a FIFO policy including a 95 % confidence interval _____	94
Figure 58: Potential for throughput increment on Line 1 depending on amount of repair workers _____	96

List of tables

Table 1: States of different system entities adapted from _____	32
Table 2: Comparison of bottleneck detection methods _____	39
Table 3: Example of production data for a period of 9 time steps _____	47
Table 4: Overview of conducted use cases within the simulation study _____	59
Table 5: Input data for the Demo Use Case _____	61
Table 6: Settings for the System Sensitivity Analysis _____	61
Table 7: Settings for the validation of the APM _____	63
Table 8: Settings for the validation of the BSP method _____	65
Table 9: Simulation input parameters for Industrial Use Case 1 _____	69
Table 10: Settings for prioritizing using Part-Out-Part-Out Times _____	70
Table 11: Settings for prioritizing using availabilities _____	72
Table 12: Settings for prioritizing using availabilities and redundancies _____	74
Table 13: Settings for the initial experiment – APM _____	77
Table 14: Settings for the variation of <i>Tobs</i> - APM _____	78
Table 15: Settings for the static bottleneck detection - APM _____	80
Table 16: Settings for the variation of <i>pinc</i> - APM _____	82
Table 17: Settings for the initial experiment - BSP _____	84
Table 18: Settings for the variation of <i>Tobs</i> - BSP _____	86
Table 19: Simulation input parameters for Industrial Use Case 2 _____	89
Table 20: Settings for the initial experiment – APM _____	90

List of abbreviations

AAP	Average Active Period
APM	Active Period Method
BSP	Blocking and Starvation Probability
C/T	Cycle Time
KPI	Key Performance Indicator
MTBF	Mean Time between Failures
MTTR	Mean Time to Repair
OEE	Overall Equipment Effectiveness
TP	Throughput
TPM	Total Productive Maintenance
WO	Work Order

1 Introduction

Maintenance has become increasingly important since companies are more and more subject to a tough competition on international markets. To be able to compete successfully on these markets, manufacturing companies have to increase their operational effectiveness and revenues while reducing their operating costs. Since a big share of the operating costs of a manufacturing company are the labour costs, some companies move their production in low-wage countries in order to stay competitive. But in the long term and regardless the location, labour intensive manufacturing processes will be substituted more and more by automated processes. The high degree of automation requires production facilities which are usually more complex and more expensive than non-automated facilities. Because of that, two important conclusions can be drawn about the potential for cost reduction initiatives in the field of maintenance management. First, the increased complexity requires less skilled production workers, but more skilled maintenance workers. This results in an increasing share of maintenance labour costs compared to other labour-intensive activities and brings maintenance more in the focus of a company's management. The second point is, that more expensive production facilities imply also higher downtime costs. In order to reduce the downtime costs due to machine breakdowns, modern maintenance strategies evolved towards preventive maintenance concepts. But since exceeding preventive maintenance would also increase the downtime of production facilities, a *Zero Breakdown* strategy cannot be realized yet cost-efficiently in all industries. Therefore, machine breakdowns and thus also corrective maintenance is still an important aspect in the field of maintenance management (Biedermann, 2008, p. 9; Leidinger, 2014, pp. 1).

This thesis is aiming to improve the efficiency of workforce distribution in a plant concerning corrective maintenance tasks. As a part of a research project of the Technical University of Graz with a German car manufacturer, the potential for throughput improvement through dynamic prioritization of corrective maintenance tasks will be evaluated.

1.1 Problem Definition

The mentioned research project deals with the usage of *Internet of Things* in the field of maintenance management. Besides Big Data Analysis for the purpose of predictive maintenance and the related scheduling of maintenance tasks, one aspect of the project is the evaluation of the organizational structures within the plant of the project partner. Whereas the maintenance staff is today organized in a decentralized structure, the findings of the research project recommend a centralized maintenance department for the whole plant. In that case, one central authority has to distribute the maintenance workforce on the different production lines within the plant. In order to support this central authority in how to distribute the workforce most efficiently, a prioritization method for machines shall be developed. The focus of this thesis is on the prioritization of corrective maintenance tasks for the case of multiple machine breakdowns. Figure 1 is a simplified example of a situation where a prioritization of maintenance tasks is required.

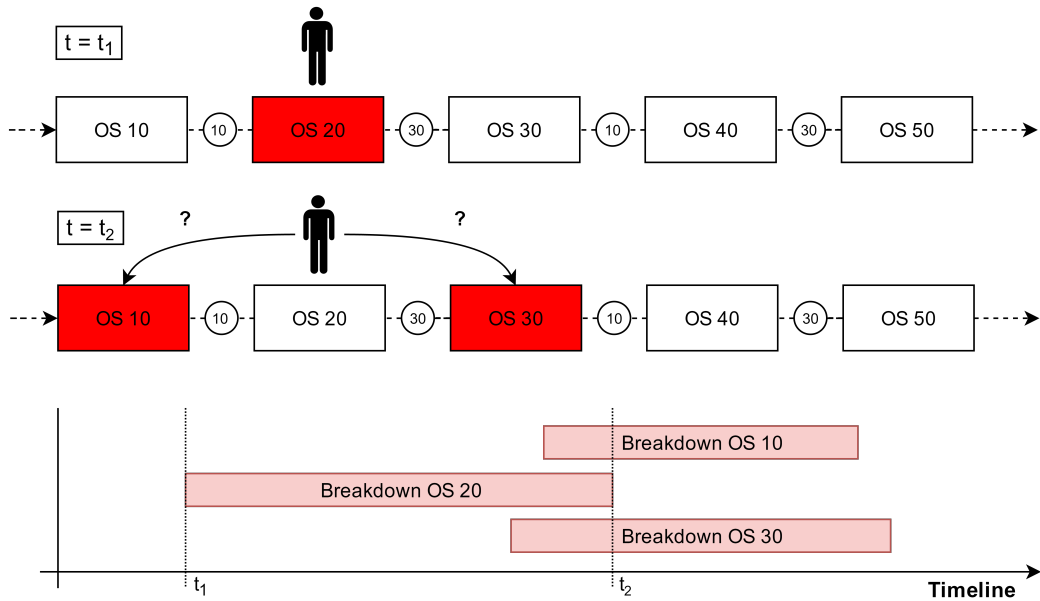


Figure 1: Situation on the plant floor which requires a prioritization of machines

The red machines symbolize a machine breakdown which requires a maintenance worker. At time t_1 , the maintenance worker can maintain OS20 immediately, since there are no other breakdowns. No prioritization is required. During OS20 is maintained, OS10 and OS30 breakdown, wherefore the maintenance worker has to prioritize which machine to maintain first at time t_2 . Currently, the prioritization of machines is done based on the subjective assessment of the maintenance worker and the line manager.

1.2 Goals

The aim of this thesis is to develop a data-driven method for the prioritization of corrective maintenance tasks on machines within a production line. The main idea, is to prioritize bottleneck machines over non-bottleneck machines to generate a throughput improvement without increasing the available personnel resources. Therefore, the first task is to develop a bottleneck detection algorithm which is capable of identifying dynamic bottlenecks by analysing data from the production data acquisition system. The output of the algorithm is a priority ranking, which can later be used by the maintenance department as a decision support for the workforce distribution in case of simultaneous breakdowns. In order to evaluate the potential of the developed prioritization method, a simulation study is conducted on two different production lines of the mentioned plant.

2 Theory

2.1 Maintenance

Due to the advancing globalization and the associated intensification of competition, companies have to provide high-quality products to low costs. One way to achieve this goal is the increasing usage of automatization technologies in the manufacturing processes. This brings new challenges for maintenance management. The increasing complexity of highly interconnected production facilities requires more efficient maintenance in order to achieve a high reliability of the overall production system. On the other hand the usage of automatization increases the amount of maintenance workers, compared to the other workers in the production process. This brings maintenance more and more in the focus of a company's management (Pawellek, 2016, p. 1; Leidinger, 2014, p. 1).

2.1.1 Terms and Definitions

The maintenance-related terms used within this thesis are defined in EN 13306:2010-10-01:

- **Maintenance**

“Combination of all technical, administrative and managerial actions during the life cycle of an item intended to retain it in, or restore it to, a state in which it can perform the required function”

- **Maintenance Management**

“All activities of the management that determine the maintenance objectives, strategies and responsibilities, and implementation of them

by such means as maintenance planning, maintenance control, and the improvement of maintenance activities and economics”

- **Reliability**

“Ability of an item to perform a required function under given conditions for a given time interval”

- **Redundancy**

“Existence of more than one unit to fulfil a required function”

- **Failure**

“Termination of the ability of an item to perform a required function”

- **Preventive maintenance**

“Maintenance carried out in accordance with established intervals of time or number of units of use but without previous condition investigation”

- **Corrective maintenance**

“Maintenance carried out after fault recognition and intended to put an item into a state in which it can perform a required function”

Further terms of maintenance used within this thesis are:

- **Availability**

Availability is the ability to perform under the assumption that all external resources are given. Breaking down the planned production time of a machine into the 4 states of “Produce”, “Blocked”, “Starved” and “Machine failure”, the availability can be calculated as follows:

$$Availability = \frac{Produce + Blocked + Starved}{Planned Production Time} \quad (1)$$

Another very common way to define the availability is the *Inherent availability* (Ebeling, 2010, p. 254):

$$Availability = \frac{MTBF}{MTBF + MTTR} \quad (2)$$

- **Mean time between failures (MTBF)** is the average of times between failures

$$MTBF = \frac{\Sigma \text{Operating time}}{\text{Number of machine failures}} \quad (3)$$

- **Mean time to repair (MTTR)** is the average of times to repair

$$MTTR = \frac{\Sigma \text{Repair time}}{\text{Number of repairs}} \quad (4)$$

- **Overall Equipment Effectiveness (OEE)** is a key performance indicator, especially in total productive maintenance, which enables an objective evaluation of production systems. The OEE is calculated by multiplying the following measures (Bellgran and Säfsten, 2010, p. 263):
 - Availability
 - Performance efficiency
 - Quality rate

A more detailed explanation on the calculation of the OEE will be given in chapter 2.1.5.

2.1.2 Objectives and Costs of Maintenance

DIN31051:2012-09 structures the tasks of maintenance into 4 basic activities:

1. **Service:** Activities to reduce the degradation of a unit
2. **Inspection:** Activities to determine the actual condition of a unit
3. **Repair:** Activities to restore the required function of a faulty unit
4. **Improvement:** Activities to increase the reliability, maintainability or safety of a unit without changing its original function

These basic actions are taken to achieve the following primary objectives of maintenance (Leidinger, 2014, p. 15):

- **Safety**
- **Availability**
- **Reliability**
- **Value retention**

The objectives concerning safety consist of health, safety, security and environmental issues which are mostly regulated by legal requirements on how to maintain production facilities. The goals of increasing availability, reliability and value retention are internal objectives of a company (Leidinger, 2014, p. 16). Besides these goals, maintenance management always aims for **reducing costs**. Figure 2 shows the two costs, which have the biggest influence on the total costs of maintenance. The blue line shows the costs for planned maintenance activities. Contrary to that there are the downtime costs in red, which occur due to lost profit when a machine is not available.

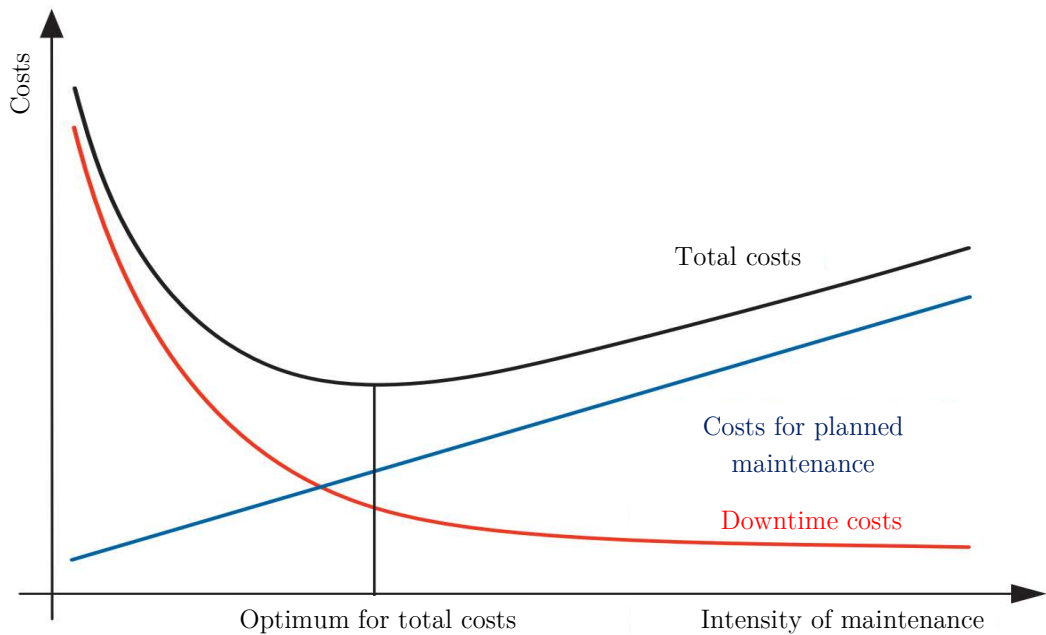


Figure 2 Optimum for total maintenance costs adapted from Matyas (2011, p. 41)

It is plausible that the more a company invests in planned maintenance activities, the less downtime costs occur. But the marginal downtime-cost-reduction of these activities is diminishing, whereas the planned maintenance costs increase linearly. The optimum of the total costs is the minimum of the sum of planned maintenance costs and downtime costs. Even this model appears very logical, it has only limited significance in practice. The reason for that is that in practice, one can only determine the actual position on the total costs curve. Determining the shape of the total cost curve by increasing or decreasing maintenance intensity is not possible, since a change in machine downtime might not occur immediately, but with a delay of several months. Since the shape of the curve is not known in practice, no statement whether the optimum requires a more or less intense

maintenance, can be made. Therefore in practice risk-based strategies are used to determine the required maintenance intensity for production facilities (Matyas, 2011, pp. 49).

2.1.3 Key Performance Indicators in Maintenance Management

As already stated, the optimum maintenance intensity cannot be calculated based on costs for complex manufacturing systems. But still in the past, measurement and evaluation of maintenance services was primarily done using cost aspects. The problem of this method is, that the performance of maintenance cannot always be determined economically. On the one hand, the results of maintenance activities can only be measured time delayed and on the other hand some results as an increased machine-lifetime or a reduced machine degradation can only be measured very limited (Schröder, 2010, pp. 159).

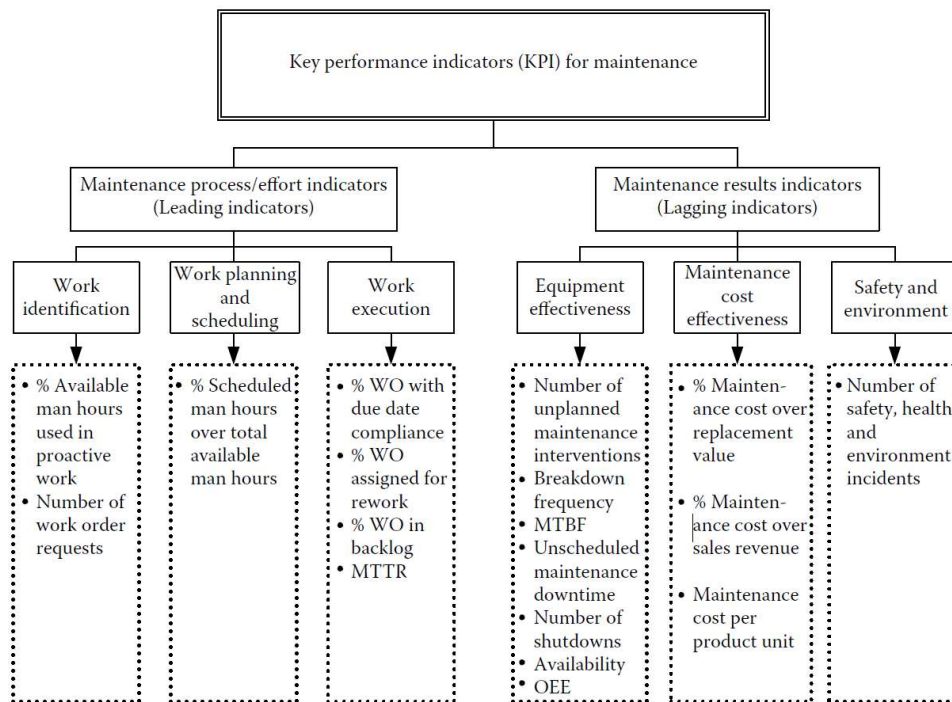


Figure 3: Excerpt of KPIs for maintenance management © (Pascual and Kumar, 2016, p. 9)

Figure 3 gives an overview about some KPI's for maintenance management. Especially for companies which follow the concept of *Total Productive Maintenance* (TPM), the most common KPI is the *Overall Equipment Effectiveness* (OEE). A

detailed explanation about the OEE and its importance to the concept of TPM will be given in chapter 2.1.5.

2.1.4 Maintenance Strategies

Maintenance strategies define the methods and rules that are used to fulfil the maintenance objectives. The strategy regulates which activities are taken at what point of time and on which production facility. Figure 4 shows four main strategies which can be divided in corrective and preventive maintenance activities. A well balanced maintenance concept, should include an optimal mix of corrective and preventive maintenance activities (Matyas, 2011, pp. 105).

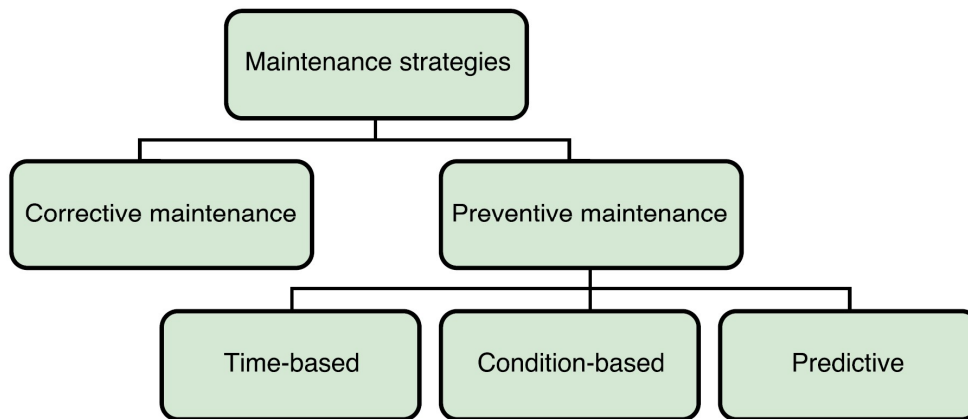


Figure 4: Maintenance strategies adapted from Matyas (2016, p. 120)

The application of a suitable maintenance strategy is decisive for the reliability of a production system as well as the total maintenance costs. Since the effects of a new strategy cannot be measured immediately, it is difficult to make statements about their effect on the total maintenance costs. Therefore, when it comes to controlling decisions in maintenance management, performance indicators as the downtime of machines or the utilization of wear stock, are good indicators for the evaluation of a maintenance strategy (Schenk, 2010, pp. 26).

2.1.4.1 Corrective Maintenance

Corrective, sometimes titled as reactive maintenance, is a strategy where maintenance is only applied in the case of a machine failure or a predefined wear-level. At first sight, this strategy seems to be very cost-efficient since every machine's wear stock is completely depleted before a repair is done. There are also no costly planning activities required. But on a closer examination, this strategy

brings a lot of drawbacks. Concerning the maintenance objective *safety*, there are a lot of machines where this strategy cannot be applied, since a breakdown of a part or a machine failure would be hazardous to the operator or the environment. Concerning the maintenance objectives *availability* and *costs*, a corrective maintenance strategy brings also drawbacks, since unplanned machine failures are more difficult to handle, than planned downtimes (Schenk, 2010, pp. 26).

Figure 5 shows the actions that are taken after the breakdown of a machine. The net maintenance time is only a fraction of the total machine downtime in the case of corrective maintenance. For planned or preventive maintenance activities, the machine downtime should be much shorter, since some activities can be run in parallel to production (Matyas, 2011, p. 107).

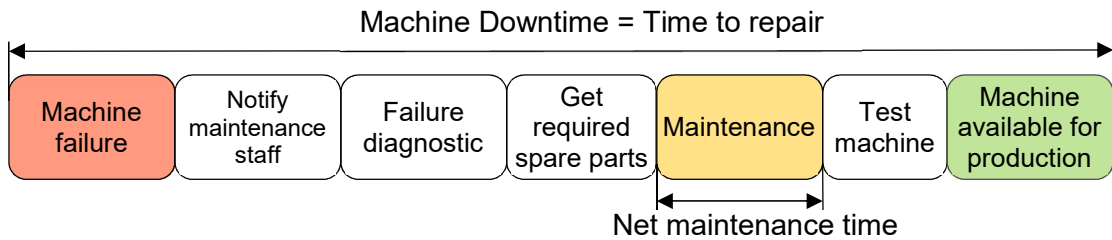


Figure 5: Activities during a machine failure adapted from Matyas (2011, p. 107)

Another risk of corrective maintenance is that the supply with all required resources (staff, spare parts, etc.) cannot always be guaranteed. To summarize this, corrective maintenance is the most critical one concerning the availability of a production system and it causes the highest costs compared to other maintenance strategies. A corrective maintenance strategy is therefore only suitable for machines that have a low criticality to the overall production system. This means that a breakdown is not hazardous to staff or environment and it will not cause any interruptions of other machines. Furthermore a corrective strategy can be applied if there are enough redundancies and if the availability of maintenance resources (material, staff) can be guaranteed. But even when a company is following a preventive maintenance strategy, machine breakdowns cannot be avoided completely. Therefore, corrective maintenance is still an important aspect in maintenance management, since the unplanned downtimes due to a breakdown have severe consequences on the OEE of a production facility (Schenk, 2010, pp. 26; Matyas, 2011, pp. 107).

2.1.4.2 Time-based Maintenance

Time-based strategies follow the directive, to maintain machines before they lose their required function. After a predefined interval of time, produced pieces, kilometres, etc. is reached, components will be changed preventively. The length of these intervals, is usually given through specifications of the manufacturer or they are set based on prior experience. Using a time-based maintenance strategy will decrease the risk of machine failures significantly compared to a corrective maintenance strategy. Another big advantage is the possibility of planning. The maintenance work can be planned in such a way, that enough resources are available and, if possible, production is not affected at all. For this strategy, the definition of the maintenance intervals is a very crucial thing when it comes to availability and costs. Too short intervals might affect production and decrease the available time for production. Furthermore it is not cost efficient to change a component before it has reached its wear limit.

In order to improve this strategy and make it more cost efficient, every component should be used as close as possible to its wear limit. This requires good documentation of prior failures for each component. Then statistical procedures could be used to extend each components maintenance interval and still prevent most of the failures.

This strategy is required for all machines, where a breakdown would be hazardous to the staff or the environment. Furthermore it is a good strategy for components, where a change is much cheaper than a breakdown (e.g.: Oil-filters) (Schenk, 2010, pp. 28).

2.1.4.3 Condition-based Maintenance

Condition-based maintenance is a further improved preventive strategy. Instead of using fixed intervals, inspections are used to determine the best time for a component replacement. By doing that, the usage of wear stock of a component can be optimized, since a replacement is only done, if a component requires one. The surveillance of components can be done through manual inspections, or by using a condition monitoring system.

Condition-based maintenance only works, when the degradation of components is measurable and when there is a clearly defined wear limit. Furthermore it is only recommended, if the costs for continual manual inspections or the costs for a condition monitoring system are economically justifiable. If those requirements are

given, condition-based maintenance can increase the availability of machines, prevent breakdowns and detect failures long enough before they occur, in order to give enough time to the maintenance management for planning a restoration (Schenk, 2010, pp. 30).

2.1.4.4 Predictive Maintenance

A further improvement of a condition-based approach, is the predictive maintenance strategy, which aims for detecting also concealed failures which are usually not measurable through condition monitoring. Starting point of this strategy is a more detailed look on the functions of a machine (Schenk, 2010, pp. 32). The functions of a machine can be divided into three categories (Moubray, 2000, pp. 35):

1. The **Primary function** of a machine is derived from the main reason, a company acquired the machine.
2. **Secondary functions** are additional requirements a machine is expected to fulfil. Those can be safety, environmental or economical functions of a machine.
3. **Superfluous functions** are all functions or components that do not serve the primary or secondary function of a machine. According to Moubray (2000), it is not unusual that 5 – 20 % of the components of a complex system are superfluous. Even these components are not value-adding, a failure could still influence the machine's performance.

After determination and classification of the functions of a machine, potential failure modes can be identified and eventually eliminated. Compared to condition-based maintenance, this strategy does not focus on primary functions where failures are more likely to detect. This strategy is looking at a machine as a whole, and tries to detect all potential (concealed) failure causes that could influence a machine's performance.

Especially the usage of Big Data Analysis has created new possibilities in the field of predictive maintenance. A set of data, consisting of various records of data about the primary, secondary and superfluous functions of a machine, is analysed concerning any patterns that would indicate an upcoming machine breakdown (Schenk, 2010, pp. 32; Moubray, 2000, pp. 35) .

2.1.5 Total Productive Maintenance

The concept of total productive maintenance (TPM) was developed by Seiichi Nakajima in the 1960's. The concept follows the principle of preventing all kind of losses in order to produce immaculate products without having any disturbances of production. In a nutshell this means *Zero-defects* and *Zero-breakdowns*. To achieve this, TPM combines different maintenance strategies and executes them through the participation of all employees (Reichel *et al.*, 2009, pp. 79; Nakajima, 1988, p. 1).

Over the years TPM evolved from a concept into a management system which considers maintenance as a necessary and vital part of a business. In order to set measurable goals, TPM uses the OEE as a core metric for measuring the performance of a production system. This metric has become widely accepted and can therefore also be used for benchmarking purposes (Ahuja and Khamba, 2008, pp. 722).

Figure 6 shows the calculation of the OEE, where the value adding operating time is reduced by six types of losses. These six losses are classified in three categories:

- Failures & set-up adjustments (Downtime losses)
- Idling & reduced speed (Speed losses)
- Defects & reduced yield (Defect losses)

The overall goal is to increase the value adding operating time and eliminate all kinds of losses. In order to identify losses properly, a visualization as it is done in Figure 6 should be done. This makes it easier to analyse the losses and set further steps for improvement projects (Matyas, 2011, pp. 191).

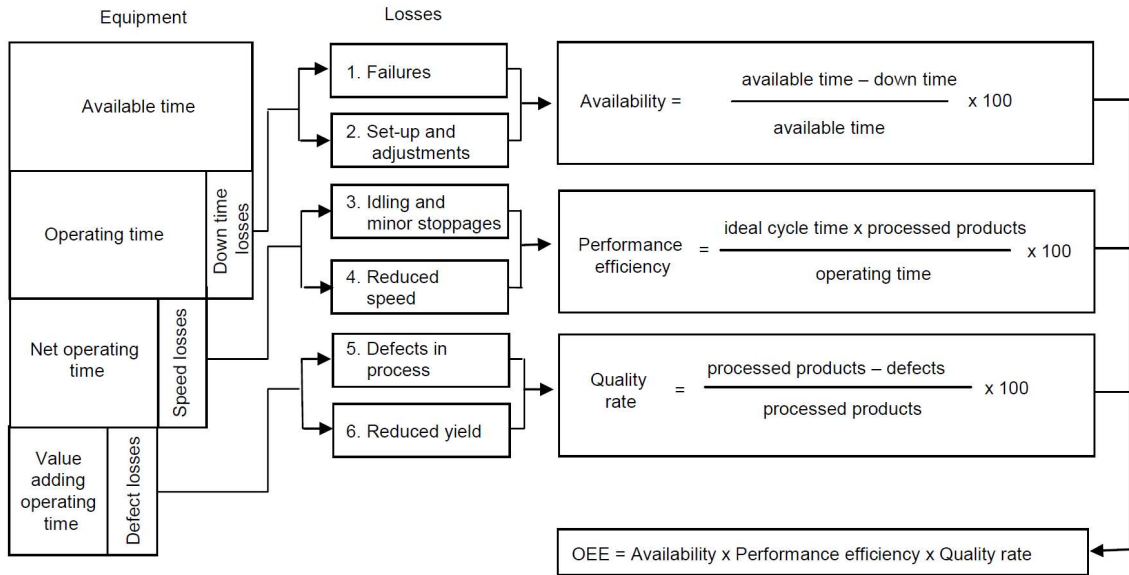


Figure 6: Calculation of OEE © (Bellgran and Säfsten, 2010, p. 263)

The framework for those improvement projects is visualized in Figure 7. The rooftop of the TPM framework represents the goals and objectives of TPM. Those goals have to be specific, measurable, achievable and reasonable and furthermore they need a proper time frame (SMART-Goals). In addition, the whole management has to adhere to the defined goals. To achieve the goals, TPM provides five basic methods which are called the pillars of TPM. The foundation everything is built upon, represents the kind working behaviour a company has to promote, in order to work more efficient.

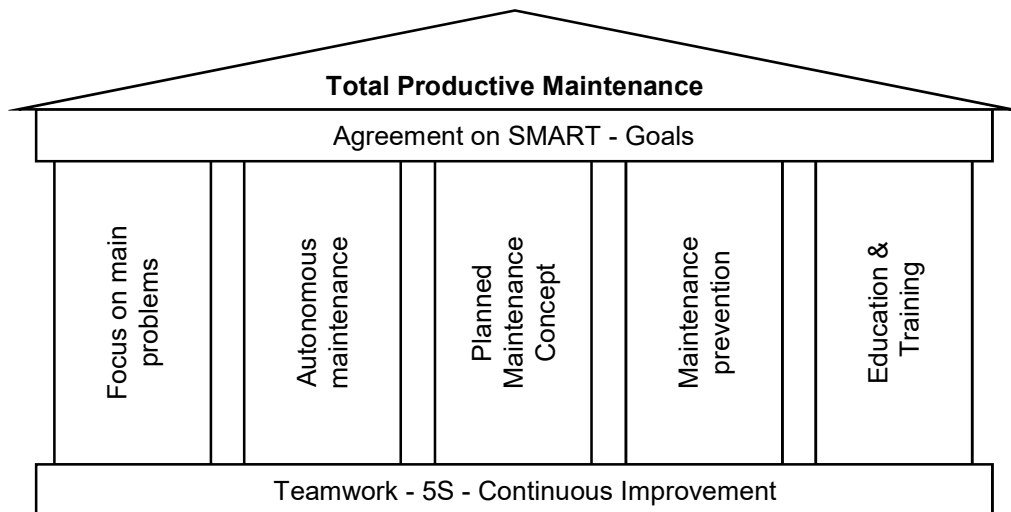


Figure 7: The five pillar model of TPM adapted from Matyas (2011, p. 200)

The key elements of a desired working behaviour are the following:

- **Teamwork:** TPM involves all departments and it requires the active participation of all employees from the shop floor to the top management.
- **5S:** A workplace organization method known from the Toyota Production System (Ahuja and Khamba, 2008, p. 732).
 - Sort – Sort out unnecessary items from the workplace
 - Set in order – Arrange items in a good order in order to pick them up easily
 - Shine – Clean the workplace regularly
 - Standardize – Set up standards for workplace organization
 - Sustain – Train and motivate people to follow the 5S-principle
- **Continuous improvement:** Approach also known from the Toyota Production System which empathizes all employees to improve all products, processes or services constantly through small incremental steps.

On this foundation, the five pillars of TPM, which represent the methods of reaching the goals, are built upon (Matyas, 2011, pp. 210):

- **Focus on main problems:** An analysis of the OEE identifies losses, quantifies them and assigns them to different categories. When planning improvement projects to eliminate those losses, it is important to be aware of which kind of losses have the biggest impact on the OEE.
- **Autonomous maintenance:** In contrast to traditional maintenance management concepts, TPM involves all employees to maximize the OEE. This does not mean that a machine operator has to be capable of doing all maintenance work that a machine requires. Much more this means that the responsibility for the condition of facilities is shared among all employees and therefore everyone has to take care to keep them in a good condition. In practice this is implemented by assigning routine work as cleaning, lubricating, small inspections or the repair of small failures, to the responsibility of the machine operator. By doing that, the maintenance staff has more time to focus on their main tasks.

- **Planned maintenance concept:** A planned maintenance concept guarantees to have the required resources in order to fulfil all maintenance tasks on time. Furthermore an adequate maintenance strategy for each equipment is developed for the whole equipment life cycle.
- **Maintenance prevention:** A lot of failure sources of a machine have their origin in the machine's design. The future maintainability is therefore mainly determined during the design and development phase of a machine. This emphasizes the need for cooperation and involvement of all departments over the whole life cycle of a good in order to achieve a high equipment effectiveness.
- **Education & training:** In harmony with the continuous improvement philosophy, also employees have to be trained throughout their time of employment. The more employees know about the concept of TPM and about techniques for problem solving, the better they can participate in improvement projects.

The experience of companies that implemented TPM successfully showed the following effects (Matyas, 2011, p. 216):

- Increased product quality and equipment efficiency
- Less working incidents
- Positive effect on team spirit and sense of responsibility

The TPM framework is more than a maintenance strategy. It allows a holistic view on the topic of maintenance and its importance for manufacturing companies.

2.1.6 IT-Systems in Maintenance Management

Since the practical part of this thesis is IT-related, the following shall provide an overview about what IT-systems are used within maintenance management.

As other processes in the value chain, also maintenance management makes use of modern IT-systems in order to accomplish its tasks successfully. The main purpose of an IT-system for maintenance management is to provide the user all required data, information and technical documents which are necessary for the execution of maintenance tasks (Schenk, 2010, p. 231).

2.1.6.1 Maintenance Planning and Control Systems

The most common systems used in the field of maintenance are the maintenance planning and control systems. The main purpose of such systems is the planning, control and surveillance of all maintenance activities. The core functions can be divided into six categories (Reichel *et al.*, 2009, pp. 157):

- **Job planning and execution:**
All preventive and corrective maintenance work orders can be managed concerning task scheduling, capacity and cost planning.
- **Asset management:**
The asset management provides relevant data and information about all assets that are in the responsibility of the maintenance department. Besides master data of the assets, also historical data about prior machine failures, inspections, etc. is stored in the system.
- **Material management:**
The material management includes the management of spare parts, wear parts and other components that are necessary to ensure the availability and safety of the assets.
- **Resources management:**
The planning of personnel and operating resources is carried out by functions of resource management.
- **Analysis and reporting:**
Furthermore, maintenance planning and control systems provide tools, to create statistics and reports about the collected data.

Besides maintenance planning and control systems, also inter-divisional IT-systems are used for the purpose of maintenance management. These can be Enterprise Resource Planning, Document Management, Product Data Management and Production Data Acquisition Systems. Especially the cross-linking of information of all those systems is becoming particularly important. An example for the need of cross-linking is the prioritization algorithm proposed within this thesis. The algorithm requires input data that comes from the production data acquisition system. The output is a priority ranking, which shall be implemented in the job planning function of a maintenance planning system.

The first part of the theory-chapter has given a brief overview about the topic of maintenance. The following part examines, which factors influence the priority of a maintenance task and how a data-driven prioritization method can be developed.

2.2 Prioritization policies

In maintenance management, the prioritization of work-orders can become a crucial task, especially in the case of limited resources. If there are more work-orders than there are workers available, the sequence of execution might have an impact on the performance of a production system. A random execution of work-orders might potentially extend the production downtime, cause losses and decrease the efficiency of production facilities. Since the importance of task prioritization is well recognized in industrial communities, most companies have internal policies to determine the optimal sequence of work-order execution. Factors that influence these decisions could be safety related issues, cost related issues as an assets value or machine related issues as the reliability of a machine and its importance to the overall production system. Since there are so many aspects that have to be considered when determining priorities, prioritization policies make usually use of heuristic rules or common sense derived from human expert knowledge (Yang *et al.*, 2007, p. 435).

This thesis aims for providing a quantitative method to assign priorities, which can then be used as a decision support for maintenance management. In order to evaluate the performance of a prioritization policy, a measureable optimization criteria is needed. Common optimization criteria in maintenance management are (Wang, 2002, p. 482):

- Minimize costs
- Maximize availability
- Minimize failure rate
- Minimize downtime
- Maximize reliability

For complex production lines, performance indicators as availability, reliability or downtime cannot be determined for the whole production line directly. Much more the performance of a production line is a function of the performance of its single machines and is further dependant on the structure of the line. Within this thesis,

the **total throughput** is used as the optimization criteria for the evaluation of different prioritization policies. The throughput of a production line is usually determined by its constraints which are known as the bottlenecks of a system. Roser and Nakano (2015, p. 274) define a bottleneck as follows:

“Bottlenecks are processes that influence the throughput of the entire system. The larger the influence, the more significant is the bottleneck.”

From a static point of view, a bottleneck can be identified easily as the machine with the longest cycle time. But the more complex a production system is, the less it behaves static. Therefore also the bottleneck situation varies over time as a result of machine failures or other events that influence a production line’s dynamics (Roser *et al.*, 2003, pp. 1192).

In the following subchapters, different service and prioritization policies will be discussed. Concerning the bottleneck-based prioritization policies, different methods to identify bottlenecks dynamically will be described.

2.2.1 First-In, First-Out

First-in, first-out (FIFO) is a service policy in queuing theory, which states that requests are processed in the order they arrive. For the case of corrective maintenance task scheduling this means that all machines are treated equally and only the time a repair request occurs, decides which request is processed first. Within this thesis, the FIFO policy is used as a benchmark to which the other policies can be compared to.

2.2.2 Heuristic Prioritization Policies

Even without a dynamic bottleneck detection, the workers on a production line are usually capable of identifying critical machines. Therefore the FIFO scenario does not come close to reality, since maintenance staff would always prioritize critical machines over non-critical ones. In the project partner’s plant, maintenance staff is using the following facts as a decision support for prioritization:

- Part-Out-Part-Out time
- Availability
- Redundancy of machines

All these methods are easy to apply since all required data is already available in the production data acquisition system. But these methods provide only limited information about the performance of a machine and they are explicitly no bottleneck detection methods. These methods are only used in order to have a simulation scenario that comes close to the heuristic prioritization policy that maintenance staff is nowadays working with. In the following, the three methods are discussed briefly:

2.2.2.1 Part-Out-Part-Out Time

The part-out-part-out time is the timespan from a part leaving the machine until the next part is leaving the machine. In the following this time is also called “true cycle time”. In comparison to the theoretical cycle time of a machine, the true cycle time also takes into account the time a machine is down, blocked or starved. The more a machine is blocked, starved or down, the more the true cycle time will deviate from the theoretical cycle time. One drawback of this method is, that it does not take into account whether a machine’s cycle time is extended due to external reasons (Blocked / Starved) or due to internal reasons (Machine Failure).

In order to have a statistical confidence of the calculated true cycle time, all cycles over a simulation run of 90 days are recorded and the median of the recorded cycle times is calculated. The reason for using the median instead of the arithmetic average is that the cycle time is limited downwards to the theoretical cycle time, whereas it is theoretically not limited upwards. In case of a long time to repair, the arithmetic mean would deliver a biased result.

Figure 8 shows an example of measured true cycle times. The machine has a theoretical cycle time of 30 seconds, so this is the most frequent cycle time measured. But due to blocking, starvation and machine errors, the true cycle time is sometimes extended. As a result of that, the arithmetic mean of this sample is 61,05 seconds. The median, which is the value where 50 % of the measured cycles are shorter and 50 % are longer, brings a result of 32,25 seconds, which is in that case a statistically more accurate result.

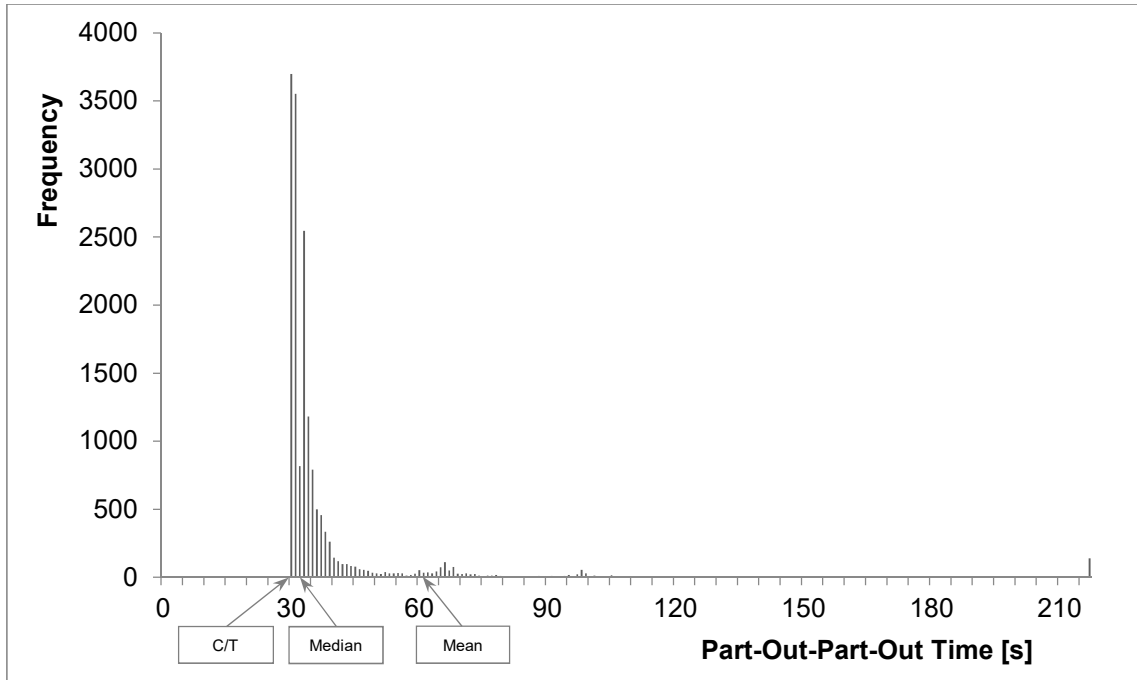


Figure 8: Histogram of measured part-out-part-out times

In order to generate priorities out of the true cycle time, the machine with the longest true cycle time is assigned the priority of 1. The priority of all other machines is calculated through the ratio of their true cycle time to the longest true cycle time.

2.2.2.2 Availability

Another method that is used in practice for estimating the criticality of machines is to compare their availabilities. As already described in chapter 2.1.1, a machine is available whenever it is capable of producing during the planned production time assuming all external resources are given. This means that during the planned production time, the only machine state that is reducing a machine's availability, is the state "Machine Failure". This is why the availability can also be calculated only by knowing the MTBF and MTTR values of a machine.

The biggest drawback of this method is that it does not take into account any effects caused by a production line's dynamics. Whether a machine is blocked frequently or not, will not influence the availability. But these external factors have a huge impact on the performance of a machine. Therefore the availability is a good indicator for evaluating the performance of maintenance management, but for evaluating the performance of a machine itself, it has only limited use.

2.2.2.3 Redundancy

Using the redundancy of machines as a prioritization factor is a very straight forward approach. Even the redundancy itself is not meaningful for the bottleneck situation of a production line, it is very useful to estimate the consequences of a machine breakdown. The reason for that is that a breakdown of a redundant machine will only slow down material flow, whereas a breakdown of a single machine will stop material flow completely.

This failure sensitivity on machine breakdowns was shown by Neubacher *et al.* (2016) using simulation.

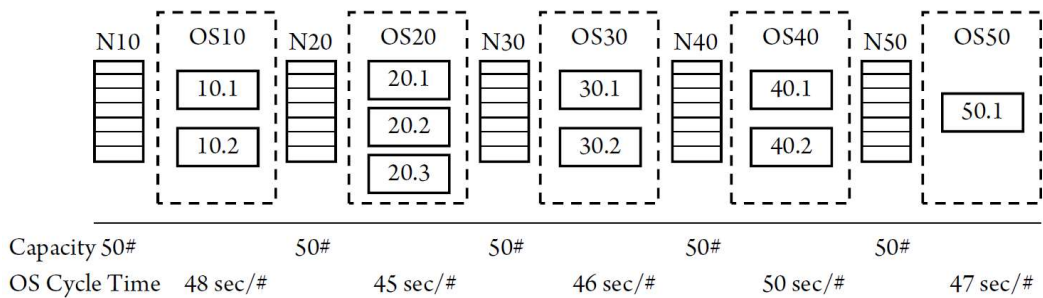


Figure 9: Failure sensitivity simulation setup © Neubacher *et al.* (2016)

Figure 9 shows a production line with five operating sequences with buffers in between. To have a baseline for the failure sensitivity, a simulation run over 72 hours without any machine failures is performed. Then for each operating sequence a simulation run is performed, where one machine of the sequence breaks down. The downtime is initially zero and is then increased by 15 minutes up to 10 hours for each machine. The impact of the breakdown on the overall system performance is measured in order to draw conclusions about the systems failure sensitivity.

Figure 10 shows the results obtained in the simulation. The time a machine was taken down during the experiment is plotted on the horizontal axis, the impact on the system's performance on the vertical axis. For example taking down one machine of the OS 50 for 18000 seconds, will reduce the system's throughput of the next 72 hours by six percent.

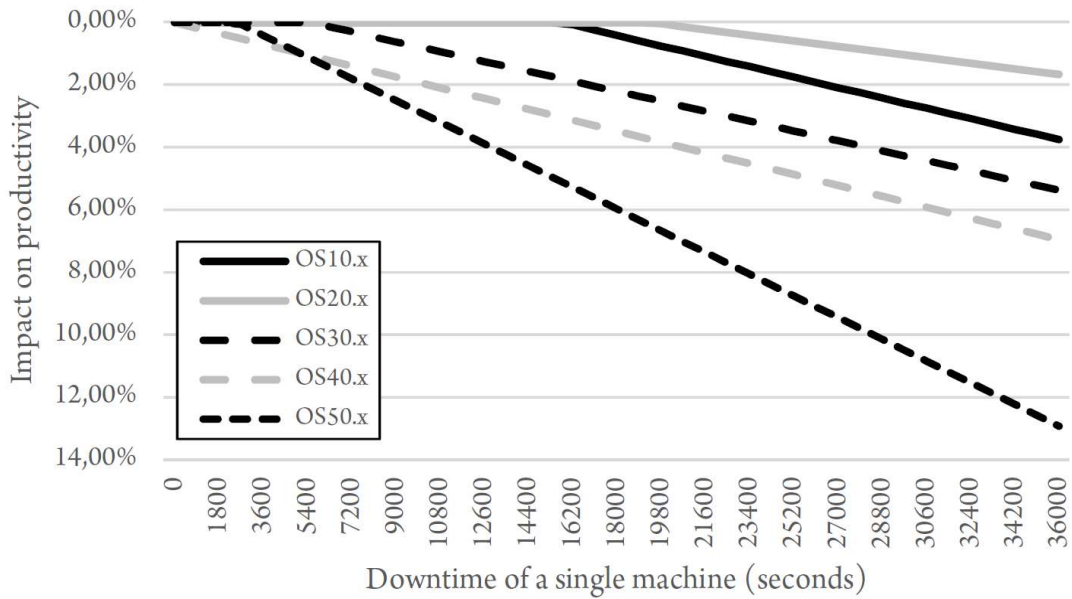


Figure 10: Failure sensitivity of redundant machines © Neubacher *et al.* (2016)

Two interesting effects are shown in this experiment. First, the duration until a breakdown will have an impact on the system's performance, is depending on a machine's cycle time and buffer level. Second, the curves are linear and their slope differs among the operating sequences depending on the amount of redundant machines they have. The less redundant machines an operating sequence has, the bigger the impact of an additional hour of downtime on the system's performance.

Since the method of Neubacher *et al.* (2016) requires the usage of simulation, it cannot be used as a heuristic method that maintenance worker on the plant floor can use for prioritizing. But the experiments confirms, that a very basic prioritization can be done using the number of redundancies in an operating sequence. On the plant floor, maintenance workers would then assign the highest priority to the operating sequence with the lowest number of redundant machines and vice versa.

2.2.2.4 Comparison of the Heuristic Methods

The benefit of the proposed heuristic methods is, that they are very easy to apply and no further data processing is required. But the Availabilities and the Part-Out-Part-Out times of machines cannot be used for assessing the criticality of machines. A machine with a short cycle time and a low availability does not have to be more critical than a machine with a long cycle time and a high availability.

The Part-Out-Part-Out Time takes the times a machine is under repair, blocked or starved into account. But since it does not distinguish between those states, in which a machine is not producing, no statement about the criticality for the system can be made.

The only heuristic method which is suitable for prioritization from a theoretical point of view, is using redundancies. A breakdown of a redundant machine will always have a less severe impact on the system performance than the breakdown of a single machine.

2.2.3 Bottleneck Prioritization Policies

The last chapter described three basic methods how prioritizing machines can be done on the plant floor. In the following chapter, bottleneck identification methods which are using real-time data are introduced. The aim is to improve the total system throughput by assigning high priority to bottleneck machines and by doing that, reduce their time to repair.

2.2.3.1 Blocking & Starvation Probability

This method utilizes the blockage and starvation probabilities of machines to identify bottlenecks. A common definition of a bottleneck is that it is the machine which limits a system's performance. Therefore an improvement in non-bottleneck machines will have a lower impact on the overall system improvement than improvements on the bottleneck machine. Equation (5) shows a mathematical formulation for the bottleneck definition. The initial throughput of a production system is compared to the new throughput after one machine i is improved by whether reducing TTR, increasing TBF or reducing the cycle time. $\Delta TP_{sys,i}$ is the system's throughput increment which is caused by an improvement of machine i and ΔTP_i is the machine's throughput increment. A bottleneck can be identified as the machine with the highest system sensitivity value θ (Chang *et al.*, 2007, p. 655).

$$\theta_{max} = MAX \left(\frac{\Delta TP_{sys,1}}{\Delta TP_1}, \frac{\Delta TP_{sys,2}}{\Delta TP_2}, \dots, \frac{\Delta TP_{sys,n}}{\Delta TP_n} \right) \quad (5)$$

This definition of a bottleneck is very precise, but it has the big disadvantage, that it is not possible to determine a bottleneck based on real-time production data. An indirect method has to be developed. The foundation of the proposed data-driven method are the characteristics a bottleneck usually follows. For example, bottlenecks cause upstream machines to be blocked and downstream machines to get starved (Li *et al.*, 2009, p. 7053).

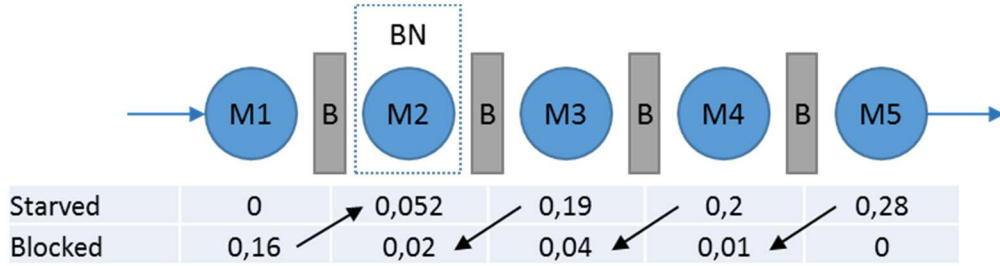


Figure 11: Bottleneck identification using blockage and starvation probabilities

Figure 11 shows a 5-machine system with four buffers in-between. Below each machine, the probability for the machine to be blocked or starved is written down. The arrows which are pointing up- or downstream are indicating in which direction the bottleneck has to be. Kuo *et al.* (1996) stated the following rules for assigning those arrows:

- If the blockage probability of a machine is greater than the starvation probability of the next downstream machine, the bottleneck is located downstream the line. In the case shown in Figure 11, this is true for M1.

$$TB_j > TS_{j+1} , \quad j = 1, \dots, n - 1 \quad (6)$$

- If the blockage probability of a machine is smaller than the starvation probability of the next downstream machine, the bottleneck is located upstream the line. In the case shown in Figure 11, this is true for M2 to M5.

$$TB_j < TS_{j+1} , \quad j = 1, \dots, n - 1 \quad (7)$$

When applying those rules, a bottleneck can be identified, if two arrows are pointing towards a machine. For the case of the first and the last machine, a bottleneck is identified whenever there is one arrow pointing towards the machine. If $TB_j = TS_{j+1}$ or $TB_j = TS_{j+1}$ and one arrow is pointing towards a machine, this machine can also be considered to be a bottleneck. Figure 12 shows configurations in which not one, but multiple bottleneck exists.

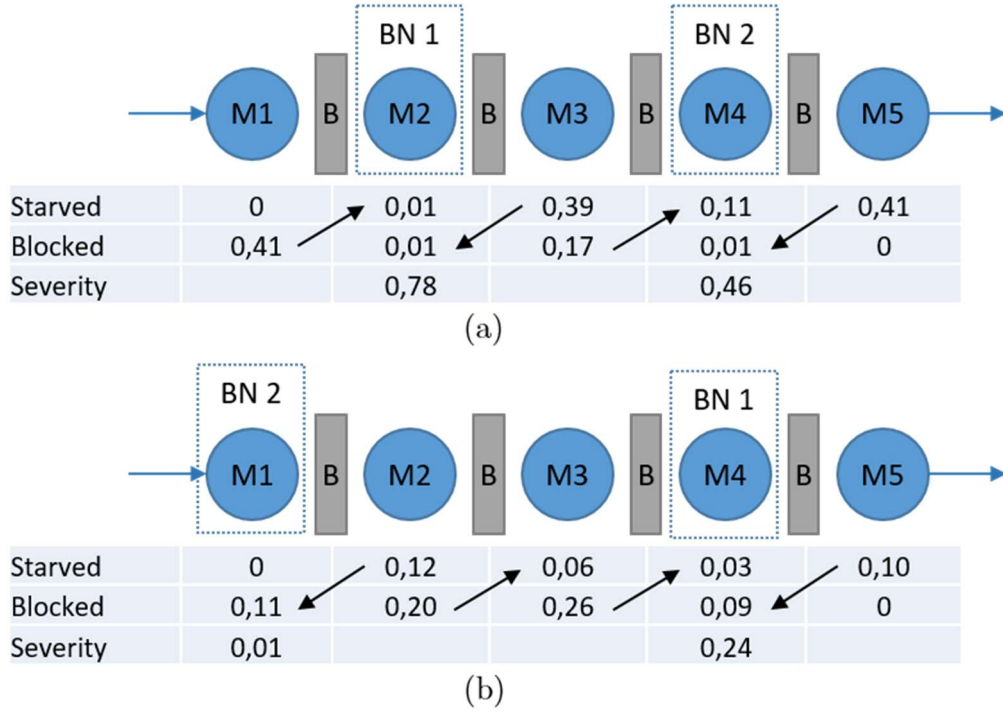


Figure 12: Examples of multiple bottlenecks

Assigning arrows according to (6) and (7) will identify two bottlenecks, but statements about which bottleneck is more severe cannot be made. Therefore Kuo *et al.* (1996) introduced the *Bottleneck Severity* S_j which is calculated as follows:

$$\begin{aligned}
 S_1 &= TS_2 - TB_1 \\
 S_j &= (TB_{j-1} + TS_{j+1}) - (TB_j + TS_j), j = 2, \dots, n - 1 \\
 S_n &= TB_{n-1} - TB_n
 \end{aligned} \tag{8}$$

If there are multiple bottlenecks in a system, the machine with the largest bottleneck severity is the primary bottleneck. Using the equations from (8) both examples from Figure 12 can be solved finding a primary bottleneck.

Kuo *et al.*, (1996, p. 248) stated that they justified the equations (6) to (8) through discrete event simulation. Also Chiang *et al.* (2001, pp. 566) simulated dozens of

systems using equations (6) to (8) to identify bottlenecks and they came to the conclusion that discrepancies between the results from this method with a validation using the system sensitivity value are quite infrequent and minimal in case they occur. Therefore the proposed method can be used for bottleneck identification in serial production lines.

For the case of redundancies, average values of the blocking and starvation probabilities of the machines in an operating sequence are calculated. Using this averages, the redundant operating sequence can be treated like a single operating sequence for the bottleneck detection.

Adaption for complex manufacturing systems

Another heuristic bottleneck detection method based on blocking and starvation probabilities was introduced by Li *et al.* (2007). Similar as the method of Kuo *et al.* (1996) it makes use of the characteristics a bottleneck machine usually follows. A Machine j is the bottleneck of a system with n -machines if the following conditions are fulfilled (Li *et al.*, 2007, p. 77):

- Bottleneck machines tend to make upstream machines blocked

$$TB_i - TS_i > 0 : i \in [1, \dots, j - 1] ; j \neq 1, j \neq n \quad (9)$$

- Bottleneck machines tend to make downstream machines starved

$$TB_i - TS_i < 0 : i \in [j + 1, \dots, n] ; j \neq 1, j \neq n \quad (10)$$

- Bottleneck machines have a lower overall sum of blocking and starvation than the neighbouring machines

$$TB_j + TS_j < TB_{j-1} + TS_{j-1} ; j \neq 1, j \neq n \quad (11)$$

$$TB_j + TS_j < TB_{j+1} + TS_{j+1} ; j \neq 1, j \neq n \quad (12)$$

- For $j = 1$:

$$TB_1 - TS_1 > 0 \ \& \ TB_2 - TS_2 < 0 \ \& \ TB_1 + TS_1 < TB_2 + TS_2 \quad (13)$$

- For $j = n$:

$$TB_{n-1} - TS_{n-1} > 0 \ \& \ TB_n - TS_n < 0 \ \& \ TB_{n-1} + TS_{n-1} < TB_n + TS_n \quad (14)$$

When plotting the blockage and starvation probabilities of a production line, the underlying idea of this method can be visualized. A bottleneck marks a “turning-point” in the trend of blockage and starvation. This turning-point can be seen in Figure 13 at M3.

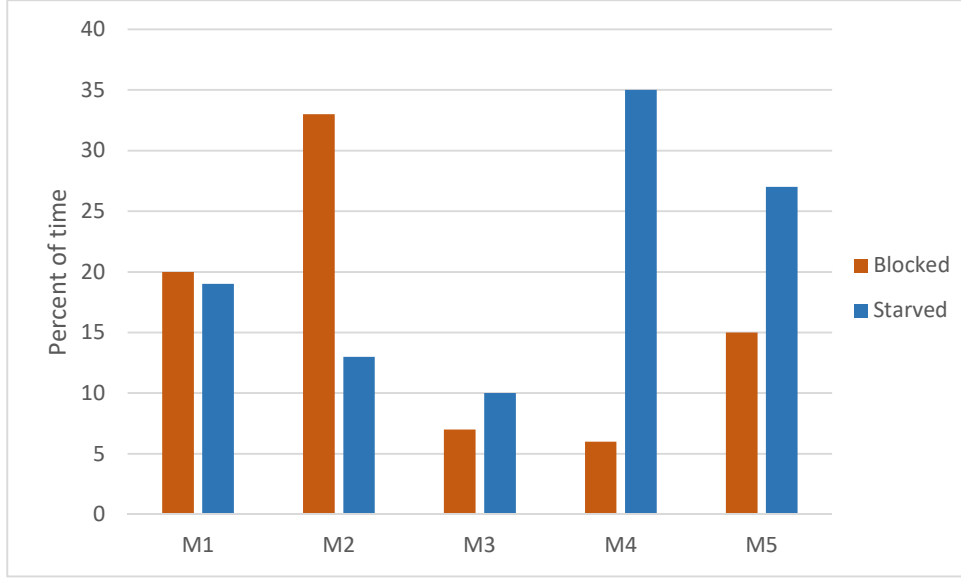


Figure 13: Trend of blocking and starvation in a serial production line

Whereas machine M1 and M2 are more blocked than they are starved, this trend changes at M3. Furthermore M3 has the lowest sum of blocking and starvation. Since equations (9) to (14) are fulfilled, M3 can be identified as the bottleneck of the system. Li *et al.* (2007) verified the proposed method analytically for a three-machine no-buffer production line. Furthermore they introduced a heuristic notion of a bottleneck index I , which can identify the primary bottleneck in a case of multiple bottlenecks (Li, 2009, p. 6932):

$$I_1 = \frac{TS_2}{TS_1 + TB_1} \quad (15)$$

$$I_i = \frac{(TB_{i-1} + TS_{i+1})}{(TB_i + TS_i)}, i = 2, \dots, n - 1 \quad (16)$$

$$I_n = \frac{TB_{n-1}}{TS_n + TB_n} \quad (17)$$

In a case of more than one identified bottlenecks, the machine with the highest bottleneck index I_n is the more significant one.

Further modifications are required in order to use this method for complex manufacturing systems. Figure 14 shows an example of concurrent processes as they exist in complex manufacturing systems. After M1, the part is processed at the station S1, where 3 processes are performed simultaneous until the part moves on to M5.

The three processes M2, M3 and M4 have to be transformed into one virtual machine V2 before equations (9) to (14) can be used to detect the bottleneck of the line.

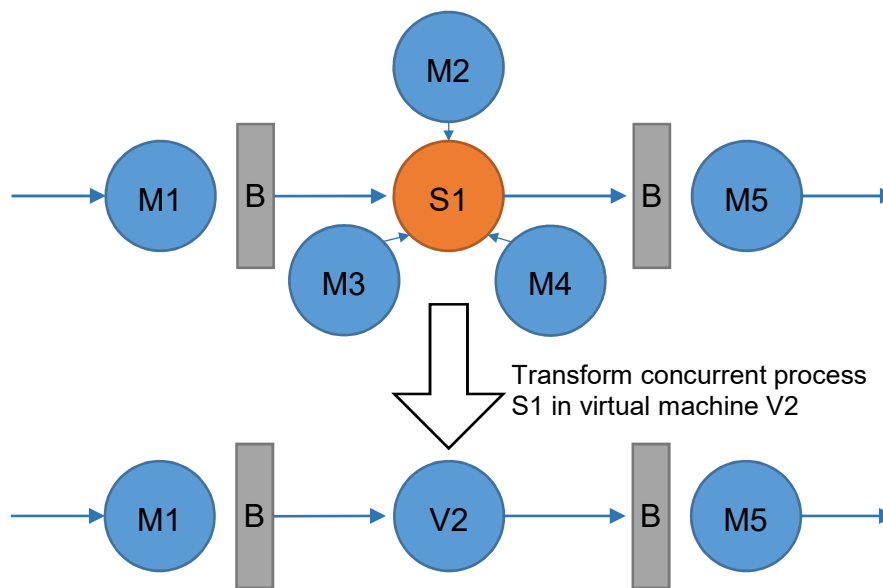


Figure 14: Transformation of a concurrent station into a virtual machine

The transformation is done using further considerations on the behaviour of the bottleneck within station S1. The part cannot move on until all processes M2, M3 and M4 are finished. This means that the bottleneck of the three processes within S1 will cause the other two processes to become starved frequently.

Therefore the bottleneck of a concurrent station is defined as the machine with the lowest TS_i . Once the bottleneck within a station is found, the station can be transformed into a virtual machine that has the blocking and starvation probabilities of the bottleneck of the station. Then the line can be treated as a serial production line using equations (9) to (14) to identify the overall bottleneck (Li, 2009, pp. 6932).

Another element of complex manufacturing systems is shown in Figure 15. Whereas the main path is a serial line, some parts leave the main part at C4, go through the feedback loop and enter the main path again at C2.

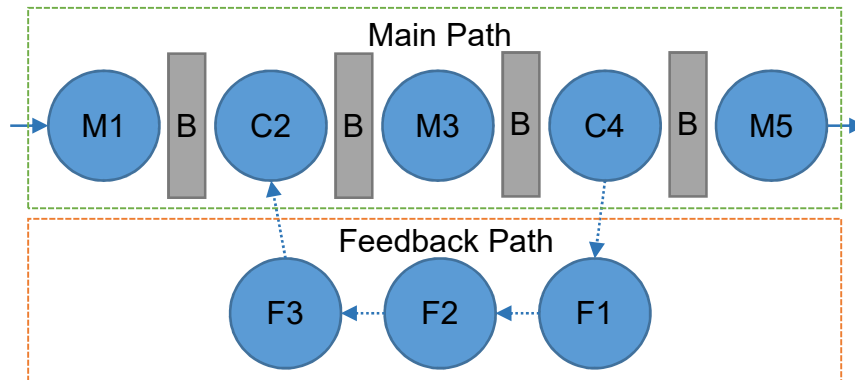


Figure 15: Feedback loop in a serial line production

For identifying bottlenecks in a system with a feedback loop, first the main path is analysed using the same equations as for the serial line. If the machines at the branches (C2 and C4) are not identified as bottlenecks, the machines in the feedback loop cannot constrain the main system and the bottleneck has to be whether M1, M3 or M5.

If C2 or C4 are identified as bottlenecks in the first analysis, the feedback loop has a potentially great influence on the overall system. A second analysis is required in order to find out, whether the feedback loop is slowing down the branch machines in the main path, or the branch machines are really the overall bottleneck of the system. Therefore potential bottlenecks in the feedback path (F1, F2 and F3) have to be detected. If the results of the bottleneck analysis of the feedback path detects the first or the last machine of the path as a bottleneck, the results of the first analysis are correct and the overall bottleneck is at the branches of the system (Li, 2009, p. 6933).

But if the second analysis identifies another machine than the last or the first as a bottleneck, in the example of Figure 15 machine F2, then this machine is also the bottleneck for the overall system. This is because the border-elements of the feedback path are slowed down by a bottleneck inside the path and as a result of that, they are slowing down the branch elements (C2 or C4) of the main path. Therefore the performance of the overall system is limited by the performance of the bottleneck inside the feedback loop (Li, 2009, p. 6934).

The same procedure can be applied to branches as shown in the example of Figure 16:

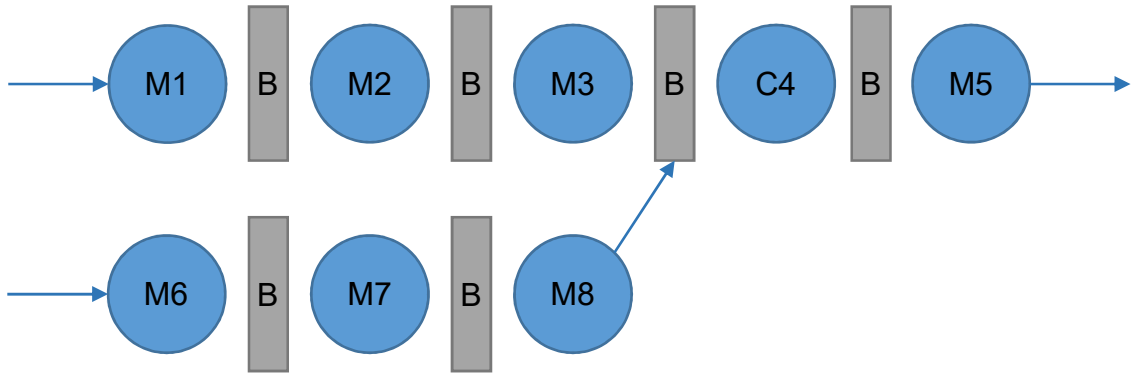


Figure 16: Branch in a serial production line

In a first analysis, the main path from M1 to M5 is analysed. If C4 turns out to be the bottleneck, a second analysis of the path M6 to M8 has to be conducted, to find out if this path is slowing down the branch machine C4 or if C4 is the true bottleneck of the system.

Since an analytical verification of complex production systems as shown in Figure 14, Figure 15 and Figure 16 is intractable, the proposed method was verified using simulation. The results were compared with the bottlenecks identified using a sensitivity analysis according to the bottleneck definition from equation (5). Li (2009) stated, that in most cases (more than 90% of over 2000 cases), the real bottleneck was successfully identified using the proposed method.

Since the planned simulation scenarios do not comprise production lines with branches or feedback loops, the first proposed blockage and starvation method, which is also called *Arrow Method* will be used for the simulation study in this thesis.

2.2.3.2 Active period method

The active period method is another method to detect the entity of a system, which has the largest effect on the overall system performance. The underlying idea is that the longer an entity is working without interruptions, the more likely it is the bottleneck of a system. The momentary bottleneck at a given point of time, is the entity which has the longest uninterrupted active period at that point of time. An overlapping of active periods of two entities, signals that the bottleneck is shifting

from one to another entity. The aim of the active period method is to detect and monitor the momentary bottlenecks as well as the shifting bottlenecks of a production system at any time t (Roser *et al.*, pp. 59).

The first step of the described method is to group all possible states of a system into an active or an inactive state. According to Table 1, active states of a machine are processing, repair, changing tools, service. Inactive states of a machine are starved and blocked. One of the benefits of this method is that it can be applied to a variety of system entities as shown in Table 1. The active period method does not require any information about the structure of a system. The only data required are the active and inactive states of the system's entities.

Table 1: States of different system entities adapted from (Roser *et al.*, 2001, p. 950)

Entity	Active States	Inactive States
Machine	Processing, Repair, Service, Tool-change	Starved, Blocked
AGV	Moving to pick-up/drop-off location, Recharge, Repair	Waiting
Human Worker	Working, Recovering	Waiting
Supply	Obtaining new part	Blocked
Output	Removing part	Waiting

Whereas a conventional bottleneck detection method would now calculate the percentage a machine is processing in order to determine the workload of the machine, the active period method measures the duration a machine is in an active state. As shown in Figure 17, the active state of a machine is not interrupted by repair or tool changing work. Only waiting which is usually because of blocking or starvation will interrupt an active state (Roser *et al.*, 2001, p. 950).

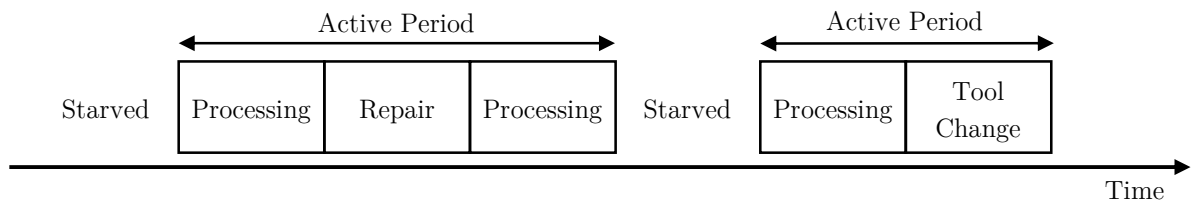


Figure 17 Active Periods of a machine adapted from (Roser *et al.*, 2001, p. 950)

Out of this perspective it can be seen, that a machine that is active over a long period without any interruptions is very likely to be the bottleneck, since it is not influenced by other machines. On the other hand a machine that is switching its state from active to inactive very often, cannot work continuously because of other machines that make it starved or blocked. Therefore machines that are often in an inactive state cannot be the bottleneck of a system.

To determine the momentary bottleneck of a system, the method compares the active durations of all machines in the system. If at a time t no machines are in an active state, there is no momentary bottleneck in the system. If there are more machines active, the bottleneck for the current period has to be the machine that has the longest uninterrupted active period.

The above described method is capable of identifying the bottleneck of a system at any point of time. But in this thesis the goal of the bottleneck identification is to have an objective variable for the prioritization of maintenance activities and therefore it would not be beneficial to switch the priorities every time a machine changes its state. This means not the bottleneck at an instant of time, but rather at a given period of time is of interest. In order to do so, two different methods are described in the following:

Average Active Period

Based on the machine data, each machine i can be classified into active and inactive states. Over a certain period of observations, each machine i has n active periods of which each period has a duration of $a_{i,j}$. The result is a set of durations A_i for each machine:

$$A_i = \{a_{i,1}, a_{i,2}, \dots, a_{i,n}\} \quad (18)$$

The average active period (AAP) of a machine i over a certain period of observation is calculated as shown in equation (19):

$$\bar{a}_i = \frac{\sum_{j=1}^n a_{i,j}}{n} \quad (19)$$

The machine with the longest average active period \bar{a}_i is considered to be the bottleneck of a system. Roser *et al.* (2001, pp. 949) have shown using simulation, that this method can detect bottlenecks reliably in steady state production systems.

To detect also shifting bottlenecks in non-steady state systems, the average active period method was developed further.

Shifting Bottleneck Detection

A shifting of the bottleneck occurs during the overlapping phase of two active states as shown in Figure 18. During these overlapping phases there is no unique bottleneck in the system. Both machines are denoted as shifting bottlenecks during the overlapping phase.

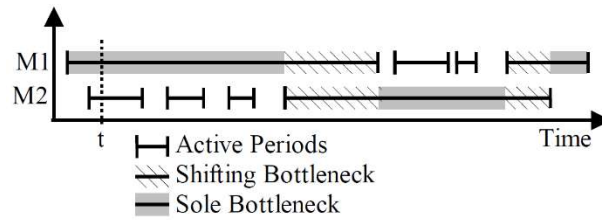


Figure 18: Shifting bottlenecks (Roser *et al.*, 2002, p. 1081)

The example in Figure 18 shows a simple system consisting of two machines. At the selected time t , Machine 1 is the sole bottleneck since it has the longest active period at that time. Before Machine 1 switches to inactive, both machines are the shifting bottleneck of the system, since the bottleneck is shifting from Machine 1 to Machine 2. During this phase the primary bottleneck cannot be determined exactly. So both machines are considered to be equally. The same happens at the end of the bottleneck period of Machine 2, when the bottleneck switches back to Machine 1.

To determine the bottleneck over a period of time, the percentage of time a machine is the sole and the percentage a machine is the shifting bottleneck is calculated for the selected period of time.

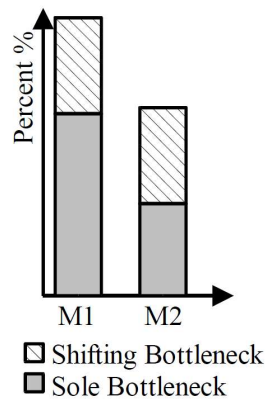


Figure 19: Average bottleneck over a period of time (Roser *et al.*, 2002, p. 1082)

Figure 19 shows the results of the average bottleneck detection with the example given in Figure 18. Machine 1 is more often the sole bottleneck of the system and therefore improvements on the performance of this machine will yield in a larger improvement of the system performance than an improvement of Machine 2 (Roser *et al.*, 2002, p. 1081).

When deriving rules for prioritizing out of the shifting bottleneck detection results, the first decision criteria is the sum of a machine's sole and shifting duration. The machine with the highest sum can be considered as the bottleneck of the system. If there are machines with an equal sum, the machine with the longer sole bottleneck duration is considered to be the more significant bottleneck.

The drawback of the shifting bottleneck method compared to the average active period (AAP), is that it only detects the most significant bottleneck of a system and provides no information about the non-bottleneck machines.

2.2.3.3 System Sensitivity Analysis

The last bottleneck identification method is the system sensitivity analysis. The basics of this method and equation (20) were already introduced in chapter 2.2.3.1. The following shall give a more detailed explanation of how this method can be applied for bottleneck detection in practice.

$$\theta_{max} = MAX \left(\frac{\Delta TP_{sys,1}}{\Delta TP_1}, \frac{\Delta TP_{sys,2}}{\Delta TP_2}, \dots, \frac{\Delta TP_{sys,n}}{\Delta TP_n} \right) \quad (20)$$

The throughput of a system ΔTP_{sys} is a complex function consisting of the system's single machines throughputs. An analytical solution for equation (20) cannot be given for a system with more than three machines (Chang *et al.*, 2007, p. 656). Therefore the system sensitivity values θ_i have to be determined using experiments or simulation.

To determine the system sensitivity values for a n -machine system, n experiments are necessary. In each experiment the performance of one machine i is changed by reducing the time to repair. Then the impact on the total system throughput $\Delta TP_{sys,i}$ is observed. Since non-bottleneck machines do not limit a system's performance, a reduced time to repair of such a machine will not improve the system's performance. The system throughput increment to its own throughput increment ratio will be close to zero, whereas the ratio of a bottleneck machine can

be up to 1. The bottleneck of a system is the machine with the maximum system sensitivity value θ_{max} (Chang *et al.*, 2007, pp. 655).

If all machines are having a sensitivity value of $\theta = 0$, the production system is optimum balanced. In that case, no balance-losses occur and the performance of the overall system cannot be improved by improving one single machine (Chang *et al.*, 2007, p. 655)

Applying this method for a long-term bottleneck detection, equation (20) will deliver the same result for all machines. The reason for that is, that a throughput increment of one machine has to result in a similar throughput increment of the total systems. Because in case of limited buffer capacities and no redundancies, a single machine cannot produce more parts than the total systems produces on a long run. Therefore for each machine the sensitivity value would be close to 1. In order to determine long-term bottlenecks using a system sensitivity analysis, equation (21) is used:

$$\theta_{LT} = MAX(\Delta TP_{sys,1}, \Delta TP_{sys,2}, \dots, \Delta TP_{sys,n}) \quad (21)$$

Equation (21) states, that the bottleneck of a system, is the machine which's single throughput increment will generate the highest system throughput increment. This equation is used for the validation of the bottleneck detection algorithms in chapter 4.1.

2.2.4 Comparison of Bottleneck Detection Methods

Before comparing the simulation results, a theoretical investigation of the bottleneck detection methods is done. The methods shall be compared regarding the following criteria:

- Accuracy in a static system
- Accuracy in a dynamic system
- Effort for implementation

2.2.4.1 Heuristic Methods

The heuristic methods determine a bottleneck using the following facts:

- Cycle time
- Part-Out-Part-Out time
- Availability

The biggest advantage of the heuristic methods is that they are very easy to implement and easy to comprehend. The required data is usually available and no actions have to be taken in order to prioritize machines using this methods.

Concerning their accuracy, a prioritization using the cycle time only works in static systems. Under the assumptions of no machine breakdowns and constant cycle times, a system can be considered as static. In such systems, the machine with the longest cycle time will always be the bottleneck. The part-out-part-out time is not suitable for prioritization in static systems with finite buffers, since all machines will have a similar part-out-part-out time in a steady state. Also the availability is not suitable for prioritization in static systems, since all machines have a similar availability in such systems.

In dynamic systems, where random machine failures occur, the heuristic methods are more problematic. All three methods cannot determine a bottleneck accurately. The reason for that is that machine breakdowns make the system behaviour more dynamic. Depending on the frequency, the time of a breakdown, the time to repair, and the buffer levels, the bottleneck may shift in a production line. Roser and Nakano (2015, pp. 276) have shown that the cycle time can therefore not be used to prioritize machines in dynamic systems. The part-out-part-out time does consider delays that occur due to the line dynamics. But it does not consider the reason why a machine has an extended part-out-part-out time. A machine that breaks down frequently will cause its neighbouring machines to be blocked and starved frequently. But in that case all machines will have the same extended part-out-part-out time and it cannot be distinguished which machine is more critical to the system. Comparing the availability values of machines would make a distinction possible, but using the availability for prioritization only works under the assumption that all machines have an equal cycle time. Whether a fast machine with low availability is more critical than a slow machine with high availability

cannot be determined accurately. Furthermore the availability is calculated using averages and can therefore not track the line dynamics accurately.

2.2.4.2 Blockage & Starvation Probability

Theoretically this method is suitable for detecting bottlenecks in static and in dynamic systems. For the arrow-based method described in chapter 2.2.3, Roser and Nakano (2015, pp. 277) have shown that situations might occur where the method cannot provide any statement about the bottleneck situation. This is the case when two neighbouring machines have equal blockage and starvation probabilities. But this is a special case and shall not be seen as a limiting factor for this method. More problematic is the distinction between primary and secondary bottlenecks, which cannot be done using the arrow-based blockage and starvation method. The modification of the method described in chapter 2.2.3 is capable of identifying and ranking multiple bottlenecks. But also for the modification, Roser and Nakano (2015, pp. 278) have shown that the method cannot identify bottlenecks reliably in practice.

The implementation of a method using blockage and starvation probabilities requires a lot of effort. Even the data is usually available in production data acquisition systems, the processing of the data cannot be done easily. The structure of the line has to be known for this method and the algorithm has to be modified for each line where it shall be implemented. Furthermore, this method is not suitable for flexible production systems or for job shop layouts.

2.2.4.3 Active Period Method

Roser and Nakano (2015, pp. 278) have shown that the active period method is capable of identifying bottlenecks in static and in dynamic systems accurately. Furthermore, this method can provide a ranking of multiple bottlenecks which is a very important aspect especially in large production systems.

The required data for this method is usually available in all production data acquisition systems. A big advantage of this method is, that it does not require any information about the production line structure or the position of machines within a line. Therefore the algorithm, which will be explained in chapter 3.1.1, can be applied to all production systems without any restrictions. There are also no further modifications necessary if the structure of a production system is changed.

Furthermore this method can also be used for flexible production systems or job shop layouts.

2.2.4.4 System Sensitivity

In terms of accuracy, this method is the best of all described bottleneck detection methods. That is why the System Sensitivity approach is in literature usually used to validate other bottleneck detection methods.

The big disadvantage of this method is, that it requires the use of simulation and can therefore not be implemented easily. Every time the bottleneck shall be detected, a simulation has to be conducted using the actual system parameters as cycle times and buffer levels. Furthermore the structure of the line has to be known, since a simulation model has to be created for each production line separately. Therefore this method has only a limited applicability for dynamic task prioritization in maintenance management.

Table 2: Comparison of bottleneck detection methods

	Heuristic Methods	Blockage & Starvation	Active Period Method	System Sensitivity
Accuracy for static systems	+	~	+	+
Accuracy for dynamic systems	-	~	+	+
Effort for implementation	+	-	~	-

Table 2 provides an overview of the discussed methods. From a theoretical point of view, the Active Period Method and the System Sensitivity Method are the best bottleneck detection methods. Still all four methods will be evaluated using simulation.

2.3 Simulation

The described prioritization policies will be analysed and verified using simulation. Therefore the following sub-chapter will give a brief introduction to the topic of simulation and especially to discrete event simulation.

2.3.1 Introduction to Simulation

Simulation is a technique to imitate the operations of real-world processes. The processes of interest are usually called a system. Figure 20 shows different methods of how systems can be studied. In a lot of fields of research, studying the actual system or a physical model of a system is expensive or simply impossible. Therefore mathematical models are established with the goal to imitate the real system's behaviour as accurate as necessary. The model building can be described as a simplification of the reality. The model itself consists of mathematical and logical relationships that can be used for the understanding of a system's behaviour. Depending on how complex the mathematical relationships of a model are, analytical solutions may or may not be available. Complex models where analytical solutions cannot be found have to be studied by means of simulation (Law and Kelton, 2000, pp. 1).

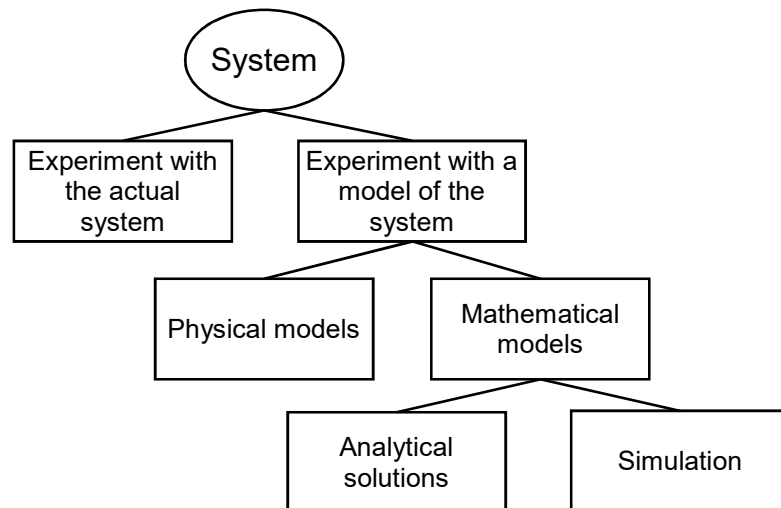


Figure 20: Methods to study a system adapted from (Law and Kelton, 2000, p. 2)

If analytical solutions are not feasible and simulation is chosen to analyse a system's behaviour, Banks (2005, pp. 11) suggests the following steps to carry out a simulation study:

1. *Problem definition*

At the beginning of a simulation study the real-world problem has to be defined clearly.

2. *Setting of objectives and overall project plan*

The objectives are the questions to be answered by the simulation. A decision if simulation is the appropriate methodology to answer those questions has to be made. Furthermore a project plan including a timetable, involved people and costs of the study has to be done.

3. *Model conceptualization*

In this step the assumptions about how the system works are defined. A crucial task in modelling a system is to abstract the essential features of it. The quality of the model is strongly dependant on the level of abstraction and complexity. It is recommended to start with a simple model and build it then towards the required complexity.

4. *Data collection*

When abstracting a problem and building a model, it is important to know which data is required to simulate a system's behaviour. Looking at a production line simulation, an objective of the simulation could be to maximize the throughput. In this case throughput is classified as output data. Then there are several parameters as cycle times, buffers and MTBFs which influence the throughput. These parameters are input data. The model itself describes how input parameters influence output parameters. The collection of the required data is an important step to get a better understanding of the system and the level of abstraction. Furthermore data of the real-world system is necessary to validate the simulation model.

5. *Model translation*

The result of the abstraction of a real-world system is a model consisting of mathematical and logical relationships between the system's entities. The next step is to translate this conceptual model into computer language. This is usually done using simulation software.

6. *Verification*

The verification is the step of checking whether the computer model is representing the logical structure of the conceptual model or not.

7. *Validation*

After verifying the computer program it is necessary to check if the conceptual model itself is an accurate representation of the real-world system. This can be done by comparing the model behaviour to the real-

world system behaviour. If discrepancies cannot be eliminated through calibration, the conceptual model has to be changed.

8. *Experimental design*

This step defines the scenarios which are planned for the simulation study. The input parameters which shall be adapted to achieve a desired output are determined. Furthermore the number and length of simulation runs is defined.

9. *Simulation runs and analysis*

Different scenarios are simulated and analysed to estimate measures of performance of the system.

10. *Additional simulation runs*

Depending on the results of the first simulation runs, it is decided whether further scenarios are necessary or not.

11. *Documentation and reporting*

The results of a simulation study have to be reported concisely and clearly in a final report. This enables others to review the model formulation, scenarios, results of the simulation and the recommended solution for the initial problem.

In addition to the final report, a program documentation shall help others users to understand and use the computer model which was built.

12. *Implementation*

The final step is to implement the results of the simulation to solve the real-world problem.

These twelve steps serve as a guideline for conducting a simulation study within this thesis. The following sub-chapter shall provide a closer look on simulation modelling and the major simulation approaches.

2.3.2 Simulation Modelling

Simulation modelling includes the processes of transforming a real-world problem into a model, analyse and optimize the model and transform the solution back into the real-world system. Depending on the problem, different approaches are used for simulation modelling. Figure 21 provides an overview over the major approaches which are used.

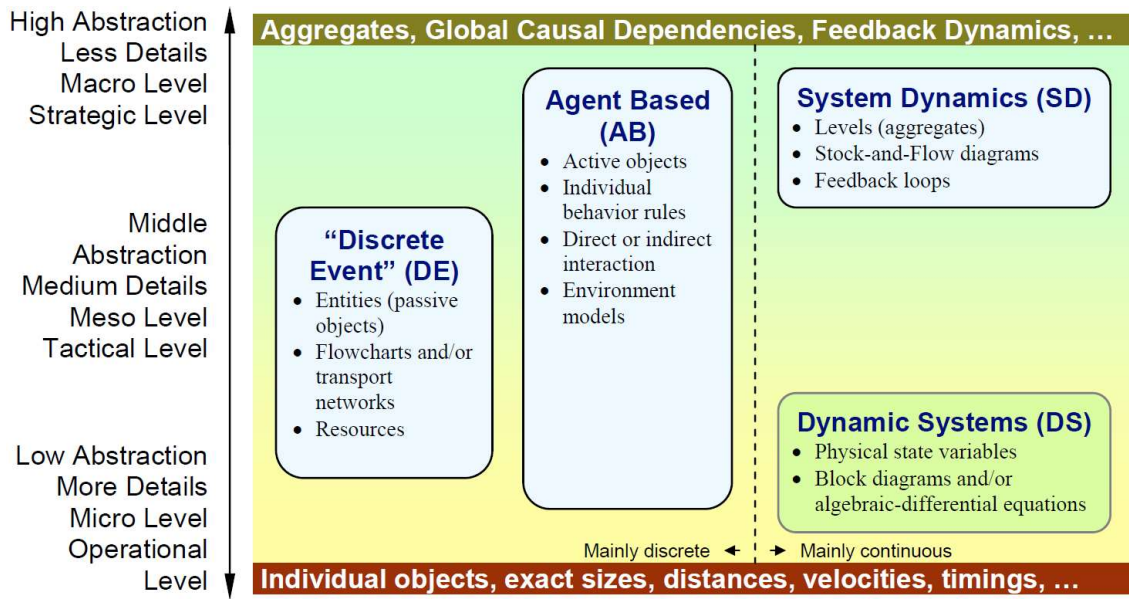


Figure 21: Major approaches in simulation modelling © (Borshchev and Filippov, 2004, p. 3)

The approaches are used for different kind of processes and different levels of abstraction. To give an example, System Dynamics deals with problems where a high level of abstraction can be applied and where the processes that have to be simulated are mainly continuous. On the other hand, Discrete Event Simulation is used to picture processes which follow discrete time steps.

2.3.2.1 System Dynamics Simulation

System Dynamics can be applied for studying urban, organizational, social or ecological types of systems. The processes within these systems are represented in terms of stocks, flows between those stocks and information about the values of the flows. The use of flows makes clear that the processes in System Dynamics Models behave continuously and they do not follow discrete time steps. The level of abstraction in System Dynamics is very high. This means, not single elements of the system are modelled, much more the behaviour of aggregates is studied. The behaviour of the aggregates is usually modelled using feedback loops which describe how an increase of one aggregate will influence other aggregates (Borshchev and Filippov, 2004, pp. 4).

2.3.2.2 Agent Based Simulation

In contrast to the other simulation approaches shown in Figure 21, there is no central logic in an agent based model which controls the model's behaviour. Much

more the behaviour is defined at an individual level and the global behaviour emerges as a result of all individual's behaviours. All entities, called agents, follow their own behaviour rules and they interact in a common environment. This is a big advantage of Agent Based Simulation, since it enables the modeller to capture more complex structures and dynamics without having knowledge about the global interdependencies (Borshchev and Filippov, 2004, pp. 6).

2.3.2.3 Discrete Event Simulation

Discrete Event Simulation (DES) concerns the modelling of a system as a sequence of events over time. The term discrete is used because the simulation moves forward in time at discrete intervals. A discrete event model consists of entities, states, attributes, events and activities. An event is defined as anything that causes a change in the state of the system or the system's entities. The state of an entity is defined by its attributes. The transformation of the state of an entity over time is carried out by activities. Activities are initiated by events and they end with the occurrence of another event (Balci, 1988, p. 291).

Using these components, DES enables us to model a variety of real-world problems and run those simulations comparably fast. The reason for that is that a DES does not track the system's behaviour over time and requires therefore less computing power compared to a continuous simulation. In DES the system's behaviour and the entities states are only tracked on discrete time steps. In between that time steps, no change in the system is assumed to occur.

How the time steps are chosen, depends on which time flow mechanism (TFM) is used. A *fixed-time increment* TFM will always advance the simulation clock by a fixed length of t and all state changes that occurred during this time step will be processed. The second approach is the *variable-time increment* TFM, which advances the simulation clock from one event to another. Depending on the timespan between those events, the time steps can differ. An advantage of the *variable-time increment* approach is that no execution time is wasted for time steps where no events happened and for the process of searching for state changes that happened during the time increment. On the other hand, systems where state changes occur in constant time steps, the *fixed-time increment* approach might be more beneficial (Balci, 1988, pp. 291–294; Law and Kelton, 2000, pp. 7).

Whether using the *variable* or the *fixed-time increment* approach, a framework is necessary to model, control and trigger the state changes of the system. This framework is called the *Conceptual Framework*, also called *World View* which is a structure concept under which the simulation is developed. The classical CFs are Event Scheduling, Activity Scheduling and Process Interaction (Balci, 1988, p. 1). A more detailed look on the conceptual framework used within this thesis will be given in the next chapter.

3 Methods & Implementation

This chapter shall provide an accurate description of how solutions for the problem stated in chapter 1 were found and implemented. The development of a bottleneck detection algorithm and the procedure of how a simulation study was conducted will be explained to ensure that the reader can comprehend and verify the results of the study.

3.1 Bottleneck Detection Algorithms

For the purpose of machine prioritization, two algorithms are developed, which create a bottleneck ranking based on real-time production data. The first algorithm uses the Average Active Period and the second the Blockage and Starvation Method, which were both described in chapter 2.2.3.

3.1.1 Active Period Method

The input data for the first bottleneck detection algorithm are the machine states that are given by the Production Data Acquisition System. Table 3 shows the data which is used as an example for the explanation of the active period method algorithm. In addition a period of observation is required as an input parameter. The period of observation determines, which time stamps will be considered for the bottleneck identification. For short periods of observation, the algorithm will detect short-term bottlenecks, whereas for long periods, the algorithm will detect the long-term bottlenecks.

Table 3: Example of production data for a period of 9 time steps

t	Machine 1	Machine 2	Machine 3
0	Produce	Starved	Starved
1	Repair	Starved	Starved
2	Produce	Produce	Starved
3	Blocked	Tool Change	Produce
4	Blocked	Produce	Produce
5	Produce	Starved	Starved
6	Produce	Starved	Produce
7	Produce	Produce	Starved
8	Blocked	Produce	Produce

3.1.1.1 State-Matrix **A**

In a first step the algorithm reads all time stamps of the machines and in order to classify them into active and inactive states. The states are written into a State Matrix A which has i columns, where i equals the number of machines in the system and j rows, where j is representing the number of time steps. The State-Matrix for the example given in Table 3 looks the following:

$$A = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \quad (22)$$

3.1.1.2 Accumulated State Matrix **B**

The next step is to accumulate the values from matrix A . The entries b_{ij} in the Matrix B represent the accumulated duration of the active period of each machine i .

$$b_{ij} = a_{ij} * (b_{i,j-1} + 1) \quad (23)$$

Using equation (23) will generate the accumulated state matrix B :

$$B = \begin{pmatrix} 1 & 2 & 3 & 0 & 0 & 1 & 2 & 3 & 0 \\ 0 & 0 & 1 & 2 & 3 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 & 2 & 0 & 1 & 0 & 1 \end{pmatrix} \quad (24)$$

3.1.1.3 Bottleneck Matrix **C**

In order to determine the bottleneck of the system, the algorithm has to find the machine with the longest active duration b_{ij} at each time step. If a machine has to longest accumulated active period in a time step, the algorithm will assign the machine a value $c_{ij} = 1$ in the bottleneck matrix C . Furthermore, the algorithm

will assign this value for all entries of the corresponding machine within the actual active period.

The result of this step can be seen in (25). Each entry where $c_{ij} = 1$ in the matrix is representing a bottleneck state.

$$C = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (25)$$

In order to distinguish between a sole and a shifting bottleneck, further steps are necessary.

3.1.1.4 Shifting bottleneck matrix D

The state of a shifting bottleneck is characterized by more than one $c_{ij} = 1$ entries in one row. Therefore, the algorithm will assign each machine the value $d_{ij} = 1$, in case there are more than one entries with $c_{ij} = 1$ in one row.

As shown in matrix D , the bottleneck shifts in time step 3 and 8.

$$D = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (26)$$

3.1.1.5 Sole bottleneck matrix E

The sole bottleneck matrix can be calculated by subtracting the shifting bottleneck matrix from the total bottleneck matrix.

$$E = C - D \quad (27)$$

$$E = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (28)$$

As described in chapter 2.2.3, there are two approaches how the active period method can be used. The first is using the average active period (AAP) to detect bottlenecks. For that case only equations (22) to (24) are required. In order to generate a bottleneck ranking, the algorithm calculates the average active period using equation (29):

$$\bar{a}_i = \frac{\sum_{j=1}^n a_{i,j}}{n} \quad (29)$$

The active durations $a_{i,j}$ and the number of active durations n can be found in the accumulated state matrix B . Then the machine with the longest active duration \bar{a}_i is assigned a value of 1 and the other machines according to their active relation related to the longest active duration.

The second approach looks at the average durations a machine is the sole or the shifting bottleneck. For the bottleneck ranking the sum of a machine's sole and shifting duration are accumulated and the machine with the highest accumulated sum can be considered as the bottleneck of the system and a value of 1 will be assigned to that machine.

As already mentioned, the second approach has the drawback of only providing information about the most significant machines and no information about non-bottleneck machines. First results, have shown that the AAP method is therefore better suited for larger production systems. Since the industrial use cases which are analysed in the simulation study, are both large production lines, the simulation study will be conducted using the AAP method.

3.1.2 Blockage & Starvation Probability

As for the AAP method algorithm, this algorithm requires the machine states and a period of observation as an input. Then the machine states within the period of observation are read and the duration each machine is blocked and starved is calculated. The probability for being blocked is the duration a machine is blocked divided by the duration of the period of observation. The same applies for the probability of being starved. Furthermore each machine has a Boolean variable which defines whether it is a bottleneck machine or not. This variable is set to *true*, if the following requirements are fulfilled:

- If the starvation probability of a machine j is smaller than the blockage probability of the next upstream machine and the blockage probability of a machine j is smaller than the starvation probability of the next downstream machine, j is a bottleneck of the system.

$$TB_{j-1} > TS_j , \quad j = 1, \dots, n - 1 \quad (30)$$

$$TB_j < TS_{j+1} , \quad j = 1, \dots, n - 1 \quad (31)$$

- The first machine of a line is considered to be a bottleneck, if:

$$TB_0 < TS_1 \quad (32)$$

- The last machine of a line is considered to be a bottleneck, if:

$$TB_{n-1} < TS_n \quad (33)$$

For the bottleneck ranking each machine where the Boolean value, which indicates it is a bottleneck of the system, is set to true, a value of 1 is assigned. This procedure explains, why this method is not capable of providing information about the importance of all machines of the system. In contrast to the AAP method, only bottlenecks machine will have a ranking which can be further used for the purpose of prioritization.

3.2 Simulation

The output of both described bottleneck detection algorithms is a bottleneck ranking for a certain period of observation. In order to evaluate the performance of the prioritization using bottleneck detection, a simulation study is required. The following describes how the simulation study was conducted.

3.2.1 Development of the Simulation Study

The simulation study is carried out, using the steps that have been described in chapter 2.3.1:

3.2.1.1 Problem definition

In large production systems, situations might occur where more maintenance workers are required than there are available. This means, especially for the case of corrective maintenance, that a prioritization might be useful to improve a production line's performance. The decision criteria for the prioritization is a machine's importance for the production line. A machine's importance can be determined by finding the bottlenecks of a production line.

Therefore a production line will be modelled and its performance using a FIFO maintenance strategy will be compared to a prioritization strategy where bottleneck machines will be prioritized over non-bottleneck machine in the case of a

simultaneous breakdown. A more detailed description of the real-world problem is given in chapter 1.1.

3.2.1.2 Setting of objectives

This study focusses on investigating the economic potential of a prioritization of corrective maintenance tasks on a single production line. Therefore different bottleneck detection methods shall be evaluated in terms of their accuracy, usability and their potential for system throughput improvement.

3.2.1.3 Model conceptualization

The simulation model is built using a hierarchical control conceptual modelling (HCCM) framework. The HCCM framework was developed, to build DES models with complex control policies. Instead of modelling queues, an advanced control mechanism steers the behaviour of a model's entities. The entities can change their role in the model and their behaviour depending on their current activity. The control structure of the HCCM framework enables the modeller to model real-world problems, where a high degree of entity interaction and complex control policies are required. Even manufacturing systems are comparably simple systems to model, the HCCM framework was used because of the possibility to implement custom dispatching rules for the repair workers (Furian *et al.*, 2014; Furian *et al.*, 2015).

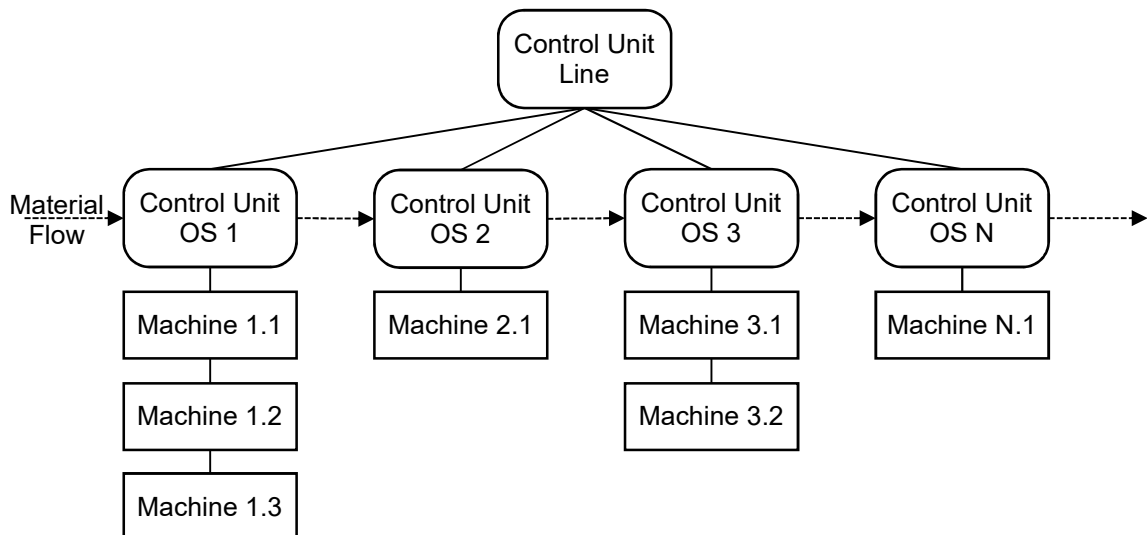


Figure 22: Main elements and hierarchical structure of the HCCM framework adapted for the usage in production research

Figure 22 shows the main elements and the hierarchical structure of the model built using the HCCM framework. The model consists of different organizational areas,

which can be seen as sub-models. Each sub-model is controlled by a Control Unit which handles the activity requests of its entities. In the proposed simulation model, the single operating sequences (OS) represent the sub-models. Each OS consists of one or, in case of redundant OS, multiple machines. One level above, there is the Control Unit Line which handles the interactions and requests of the Control Units OS.

The following shall give a more detailed explanation of the elements used within the simulation framework:

- **Machine:**
A machine is the most basic entity and it is always part of an operation sequence. The attributes of the machine are an identification number, a procedure ID of the production process and a cycle time, which are all inherited by the operating sequence. Furthermore, the state of a machine is defined by its activities which can be Produce, Wait for Material, Blocked or Repair. The activity changes are initiated by events.
- **Operating sequence:**
The operating sequence consists of one or multiple machines. An OS is defined by an ID, its position in the production line, the amount of machines it consist of and a buffer capacity.
- **Material:**
The material travels from one OS to another. Each machine has a list of procedures to pass starting with the first machine in the production line and ending with the last. If a material was processed in an OS, the procedure ID of the corresponding machine will be deleted from the list, and the Control Unit travels the material to the OS which performs the next procedure on the list.
- **Control Unit OS:**
This Control Unit handles the requests of its entities and communicates with the Control Unit Line. The following requests are handled within one Control Unit OS:
 - **Produce Request:**
The moment a material is travelled from one machine to the next in the production sequence, it sends a produce request to the Control Unit OS. Then the Control Unit OS checks the states of its machine

and in case a machine is in the activity “Wait for material”, the Control Unit removes the material from the buffer and processes it in the machine.

- Travel Request:

When a machine has processed a material it starts the activity “Blocked” and sends a travel request to its Parent Control Unit in order to travel the material to the buffer of the next OS in the production sequence. Since the Control Unit OS cannot control the behaviour of another Control Unit OS, it has to hand off this request to the superordinate Control Unit Line.

- Repair Request:

In case a machine breaks down, it switches to the activity “Repair” and sends a repair request to its Control Unit OS. Since the repair workers are organized on a line-level, the request will be hand off to the Control Unit Line.

- Control Unit Line:

This is the superordinate Control Unit which handles all request that cannot be handled within one sub-system. These request are the repair and the travel requests.

- Travel Requests:

When a travel requests reaches the Control Unit Line, the Control Unit Line acts as the communication interface between the Control Unit OS the request comes from and the Control Unit OS where the material is processed next. The Control Unit Line reads the next procedure to pass from the material’s list and checks if the OS of that procedure has still capacity in its buffer. If it has enough capacity, the material is travelled to the next OS and is given to the control of the corresponding Control Unit OS. Furthermore the activity of the machine where the material comes from is changed to “Wait for material”.

- Repair Requests:

When a repair request reaches the Control Unit Line, the Control Unit Line decides whether a repair worker can be assigned to the machine or not. These decision rules, are where the prioritization algorithm is implemented. In case there are more workers available

than requests, the Control Unit will assign a repair worker to the broken machines immediately. If there are more requests, the prioritization algorithm will be executed in order to prioritize and the available worker will be sent to the machine which has the highest priority.

- **Simulation Engine:**

The simulation engine handles the scheduled events of the simulation using an event calendar. The events are added at the beginning or end of an activity. If all events for a certain time step are executed, the simulation engine advances to the time of the next event in the calendar.

3.2.1.4 Data collection

The simulation model requires the following input data:

- Structure of the production line
- For each operation sequence:
 - Cycle time
 - Time between failure
 - Time to repair
 - Buffer capacity
 - Amount of machines (Redundancy)

- Amount of repair workers

- Simulation Period

The Simulation Period defines which period of time will be simulated. All experiments will be done simulating 90 days of production.

- Warm-Up Period

Before the 90 days start, a Warm-Up Time is necessary to ensure the simulation model has reached a steady state and all buffers are on a realistic level. The Warm-Up Period for all experiments is chosen with 7 days.

The used data comes from an engine manufacturing plant. The time to repair and time between failure values are sampled out of a list of historical production data. In context of this simulation study, the time to repair (TTR) will be used in a different way than in literature. Figure 23 shows the different tasks which are executed in a case of a machine breakdown. The conventional definition of TTR is the duration from a machine failure until the machine is available again for production. The TTR values which are sampled out of historical data, represent

the duration from when the repair worker starts his maintenance task until the machine is available again for production.

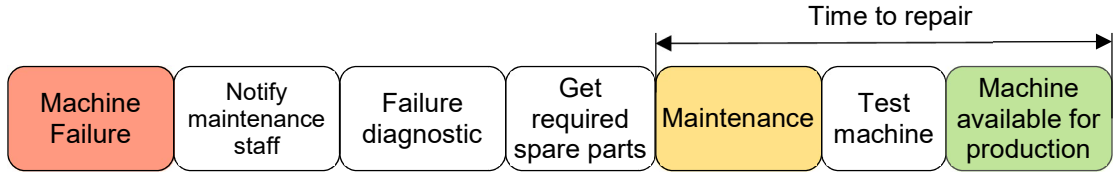


Figure 23: Time to repair in context of the simulation study

The reason for sampling out of historical production data instead of using mean values is that the machine failures do not follow any distribution. Therefore sampling out of historical data is a more precise replication of reality. The random number generator for the sampling process is using a seed. This guarantees that the FIFO simulation run will use the same sequence of TBF and TTR values as the prioritization simulation run. Otherwise the results would not be comparable.

For the bottleneck detection algorithm, more input data is required:

- Period of Observation T_{obs}
 A long period of observation will detect long-term bottlenecks and vice-versa. The first simulations have shown, that T_{obs} is a very critical factor for the success of a task prioritization. Therefore different scenarios are done to find an optimum for T_{obs} .
- Priority Increase Factor p_{inc}
 The first simulation runs have shown that, especially for long periods of observation, the algorithms do not give enough priority to non-bottleneck machines that are already waiting for repair for several hours. This is because a waiting time of several hours cannot influence the result of an observation over 7 days significantly, even this waiting time already has a significant influence on the performance of the production line. Therefore a Priority Increase Factor is introduced. Figure 24 shows an example of how the p_{inc} works. Machine 1 and Machine 2 break down at different points in time. For both machines, there was no repair worker available when the failure occurred. When a worker becomes available, the prioritization algorithm is executed. The calculated priorities for both machines are represented by the grey vertical bars. The priority of Machine 2 is higher than the one of Machine 1. So the worker would be assigned to Machine 2

first. But since the breakdown of Machine 1 happened much earlier, this machine might be more important to the system even the algorithm has not detected that.

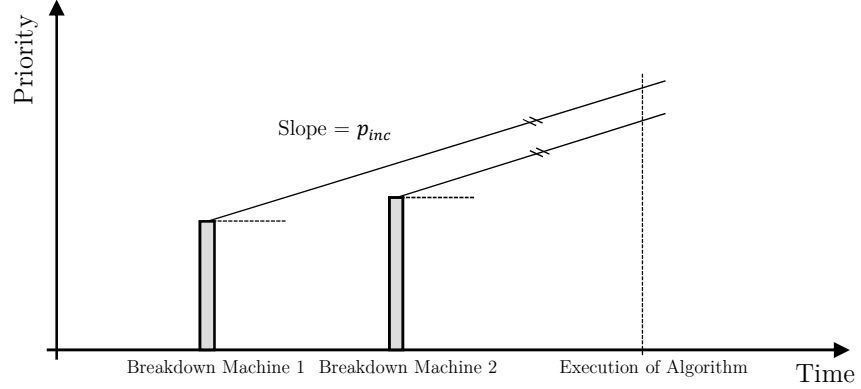


Figure 24: Introduction of a Priority Increase Factor

By increasing the priority of each machine over time using a constant gradient p_{inc} , this behaviour can be prevented.

$$Prio_{modified} = Prio_{calculated} + p_{inc} * (t_{execution} - t_{breakdown}) \quad (34)$$

Equation (34) shows how the calculated priorities are increased with a Priority Increase Factor. The higher the gradient p_{inc} is chosen, the more the prioritization strategy will behave as the FIFO strategy. Therefore different simulations will be run to find an optimum value for p_{inc} .

- Prioritization Frequency $f_{Prioritization}$
The last required input for the algorithm is the Prioritization Frequency. It determines how often the algorithm will be executed. In reality the algorithm will be executed every time, a machine breaks down. But since the execution of the algorithm takes up to a minute of time, the algorithm will be executed in a certain frequency $f_{Prioritization}$ in order to speed up the simulation.

Concerning the output data, the simulation has to provide the following values in order to validate and evaluate the results:

- Total throughput
- Throughput of each operation sequence
- Waiting, Blocking, Repair and Production time for each machine

3.2.1.5 Model translation

The conceptual model now has to be translated into a computer model. This is done using the HCDESLib, which is an open source simulation software for discrete event simulation (Furian, 2017). The HCDESLib is based on the principles of the HCCM framework which has already been described.

3.2.1.6 Validation

The simulation will be conducted over a period of 90 days. From the production data acquisition (PDA) system of the engine manufacturing plant, real historical data is used to validate the results of the simulation. For a period of 90 days, all times between failure and times to repair are available as real historical data. This TBF and TTR values are the basis for the sampling of TBF and TTR values within the model. For the validation, the total downtime and the total throughput of each machine was compared to the real production system.

3.2.1.7 Experimental design

This step defines the scenarios which are planned for the simulation study. Different scenarios are necessary to validate the result of the bottleneck detection algorithm and to optimize the performance of the algorithm using different input parameters.

3.2.2 Simulation Scenarios

Within the simulation study, the following scenarios are planned:

- Validation of the bottleneck detection algorithm:
Since the real-world production lines are well-balanced complex systems, a validation has to be done on a more simple line. Otherwise the output of the bottleneck detection algorithm would not be comprehensible. Therefore, the algorithms for the active period method and the blockage and starvation method will be validated on a simple production line using a system sensitivity analysis. The aim of this scenario is to validate whether the bottleneck detection algorithms are capable of identifying bottlenecks correctly or not. A detailed description of the line structure and the input data will be given in the next chapter.
- Industrial Use Case 1
Line 1 is a production line of an engine manufacturing plant. This scenario is a real-world use-case which aims for determining the potential of dynamic

task prioritization for corrective maintenance. Two different bottleneck detection approaches are used to prioritize machines within Line 1. For both approaches simulation runs, using different input parameters for the algorithm are conducting in order to find optimal settings. To evaluate the performance of the prioritization methods, the total throughput is compared to the total throughput using a FIFO repair policy. All simulation scenarios will be run 30 times over 90 days in order to have a statistical confidence about the results.

- Industrial Use Case 2

The prioritization method which performed best in the prior scenario, will be tested in this scenario on a different production line of the engine manufacturing plant. The aim is to test whether there is are optimal settings for the input parameters of the algorithm and whether those settings have to be modified for other production lines or not. Therefore several simulation runs using different input parameters are conducted to find out if the optimal settings are the same as for the prior scenario. As for Use Case 1, all simulation scenarios will be run 30 times over 90 days in order to have a statistical confidence about the results.

4 Use Cases

This chapter contains all results from the different use cases, starting with the validation of the used bottleneck detection algorithms on a simple demo production line. After validating the algorithms, an industrial use case for a more complex production line is conducted. The method which performs best in the first industrial use case, will then be tested in a second industrial use case to find out, whether the algorithm can be applied without any modifications on different production lines or not.

Table 4: Overview of the conducted use cases within the simulation study

Use Case	Objective	Method	Experiment
Demo Use Case	Validate Bottleneck Detection Method	System Sensitivity Analysis	Active Period Method
			Blocking & Starvation Probability
Industrial Use Case 1	Throughput Improvement by Bottleneck Prioritization	Heuristic Methods	Part-Out-Part-Out Time
			Availability
			Availability + Redundancy
		Active Period Method	Initial Experiment
			Variation of T_{obs}
			Static Bottleneck Ranking
		Blocking & Starvation Probability	Variation of p_{inc}
			Initial Experiment
			Variation of T_{obs}
			Static Bottleneck Ranking
Industrial Use Case 2	Verify Optimum Settings of Industrial Use Case 1	Active Period Method	Variation of T_{obs} and p_{inc} + Static Bottleneck Ranking

4.1 Demo Use Case

This first use case, is aiming for validating the output of the bottleneck detection algorithms. To do so, three different simulation scenarios are necessary:

- System Sensitivity Analysis
- Active Period Method
- Blocking & Starvation Method

First, a System Sensitivity Analysis is conducted, since this method is considered to be the most reliable bottleneck detection method. Then two simulations scenarios using the Active Period Method and the Blocking & Starvation Method are conducted. The output of those two scenarios, which is a bottleneck ranking, can then be compared to the bottleneck ranking of the System Sensitivity Analysis in order to validate the result.

Since the bottleneck detection for complex lines is hard to comprehend, the validation will be done for a simple Demo production line pictured in Figure 25.

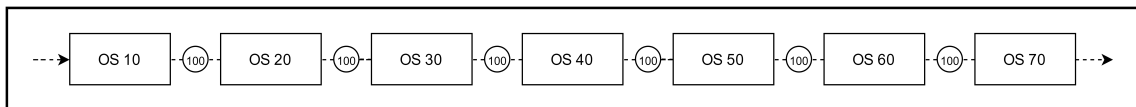


Figure 25: Structure of the Demo production line

The line consists of seven operating sequences where each consists of one machine. Between each operating sequence there is a buffer with a capacity of 100. Furthermore there is one repair worker which will perform all maintenance tasks that take longer than 10 minutes. This behaviour is set with the Boolean variable *Remove Short Repair = true*. The reason for doing that, is that small repair tasks are conducted by the machine operators and only more complex tasks require a more skilled repair worker. The machine operators are not modelled within the simulation model, since they are no subject of interest concerning the goals of this simulation study. Instead repair tasks shorter than 10 minutes are performed instantly. This simplification was resolved together with the project partner and will be used throughout all use cases of this simulation study.

Table 5: Simulation input parameters for the Demo Use Case

Global Settings					
Repair Workers	1				
Remove Short Repair	true				
Simulation Period	90 Days				
Warm Up Period	7 Days				
Simulation Runs	1				
Machine Settings	C/T [s]	Puffer Capacity	MTBF [h]	MTTR [h]	Redundant Machines
OS10	45	∞	1	1	0
OS20	48	100	2	0,5	0
OS30	65	100	1	1,3	0
OS40	50	100	2	0,5	0
OS50	54	100	3	1	0
OS60	45	100	1	1	0
OS70	48	100	2	0,5	0

The input data, shown in Table 5 will be used for all three scenarios within this use case.

4.1.1 System Sensitivity Analysis

The objective of this scenario is to create a reliable bottleneck ranking using a System Sensitivity Analysis. This ranking can be considered as the most realistic ranking and can therefore be used as a benchmark for the other two bottleneck detection methods. One simulation run, which consists of $n + 1$ experiments, will be performed, where n is the amount of machines of the production line.

Table 6: Settings for the System Sensitivity Analysis

Bottleneck Detection Method		System Sensitivity Analysis
Objective		Bottleneck Detection
Comparative Basis		FIFO Policy
Simulation Settings	Global Settings	Table 5
	Line Structure	Demo Line
Method Settings	Period of Observation	—
	Prioritization Frequency	—

The first experiment is conducted using the initial machine input data of the Demo Line (Table 5). The result of this experiment will be the baseline for the following experiments. In a next step, the performance of OS10 is improved by reducing the MTTR by 50 % and another experiment is conducted. By comparing the total throughput of this experiment to the total throughput of the baseline experiment, the impact of an improvement of OS10 on the overall system performance can be observed. This determines the value $\Delta TP_{sys,OS10}$. The higher the overall system improvement, the more OS10 is a significant bottleneck of the Demo Line.

In a next step, the MTTR of OS10 is set back to the initial value and the MTTR of OS20 is reduced by 50 %. Again the result is compared to the baseline, which determines the value $\Delta TP_{sys,OS20}$. This process is done for each machine in the production line. The machine with the highest $\Delta TP_{sys,i}$ is considered to be the primary bottleneck of the system and it is assigned a value of 1. The other machines according to their $\Delta TP_{sys,i}$ in relation to the highest $\Delta TP_{sys,i}$.

4.1.1.1 Results

Figure 26 shows the results of the sensitivity analysis of the Demo production line. The primary bottleneck of the system is OS 30. Secondary and tertiary bottlenecks are OS 10 and OS 60.

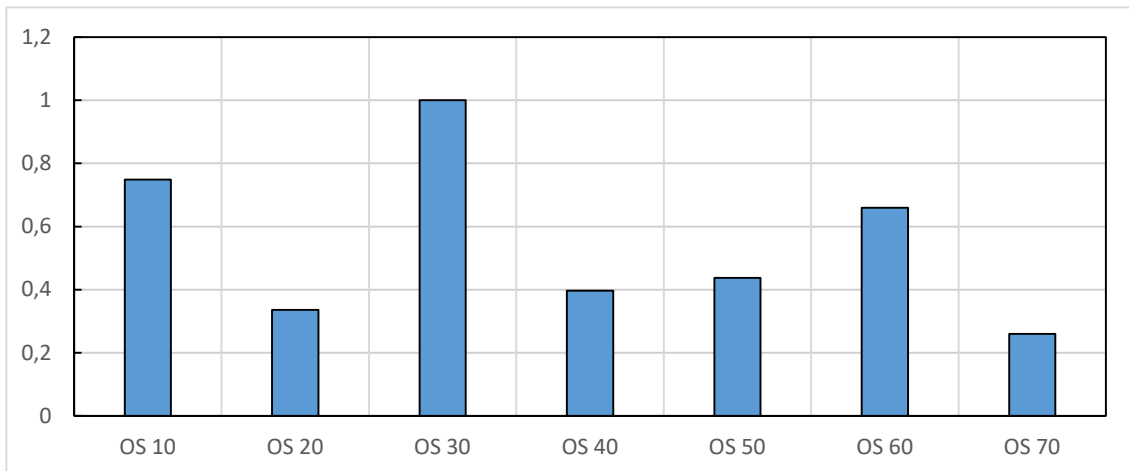


Figure 26: Bottleneck ranking of the Demo line using a system sensitivity analysis

4.1.1.2 Discussion

The basics of this method were already explained in chapter 2.2.3.3. Even this method is the most precise bottleneck detection method, it cannot be used for dynamic task prioritization on the plant floor. Even for this comparable small production line, eight experiments are necessary in order to determine the actual bottleneck situation. It is not suitable for tracking the actual bottleneck situation using real-time production data. Therefore, this method is only used to validate the two data-driven bottleneck detection approaches.

4.1.2 Validation of the Active Period Method

Within this scenario, a simulation run using the Active Period Method is conducted. The bottleneck ranking according to the Active Period Method can then be compared to the bottleneck ranking of the System Sensitivity Analysis.

Table 7: Settings for the validation of the APM

Bottleneck Detection Method		Active Period Method
Objective		Bottleneck Ranking Validation
Comparative Basis		System Sensitivity Analysis
Simulation Settings	Global Settings	Table 5
	Line Structure	Demo Line
Method Settings	Period of Observation	$T_{obs} = 1 \text{ Day}$
	Prioritization Frequency	$f_{prioritization} = 0,33 \text{ per Hour}$

This scenario is using the same simulation input data as the System Sensitivity Analysis (Table 5). Since the objective is to determine and validate the bottleneck situation, all repair requests will be executed using a FIFO strategy. This is because a prioritization would change the bottleneck situation and the bottleneck ranking could not be compared to the ranking of the System Sensitivity Analysis.

In the simulation run, the algorithm will be executed 0.33 times an hour in order to track the actual bottleneck situation of the system, considering the machine states of the last 24 hours.

4.1.2.1 Results

The output of this scenario is a bottleneck ranking over time which is pictured in Figure 27.

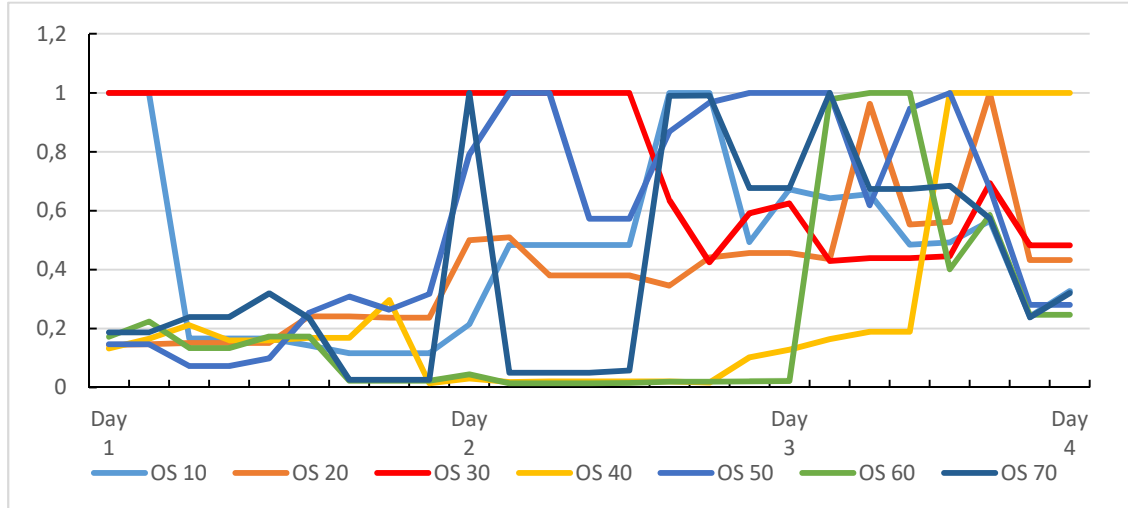


Figure 27: Shifting bottlenecks of the Demo production line using the APM

Since the System Sensitivity Analysis was conducted over a period of 90 days, an average priority over the whole simulation period has to be calculated for each machine. Figure 28 shows the result of the average bottlenecks over 90 days determined with the active period method, compared to the result of the system sensitivity analysis. The primary, secondary and tertiary bottleneck of the system are identified correctly by the average active period method.

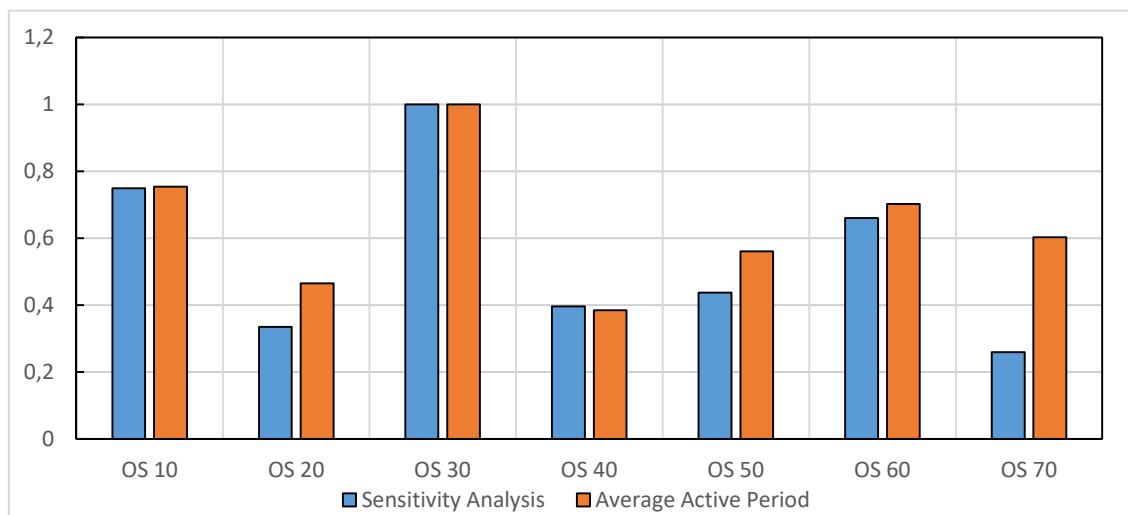


Figure 28: Comparison of the priority ranking of the active period method and the system sensitivity analysis

4.1.2.2 Discussion

The result of the average active period method comes very close to the result of the system sensitivity analysis which is considered as the real world situation. The deviations at OS 20, OS 50 and OS 70 are negligible, since those machines are the least critical machines for the overall systems. Concerning the priority over time as shown in Figure 27, it can be seen that the system behaves very dynamic and that the APM is capable of determining the dynamic bottlenecks. Even OS 30, which is marked in red, is most of the time a bottleneck machine, there are periods, where other machines are more important to the system's performance. This indicates, that the system's performance can be improved through a dynamic prioritization. Therefore the proposed method is suitable for further investigations, how a system's performance can be improved through dynamic bottleneck detection.

4.1.3 Validation of the Blocking & Starvation Probability Method

In the last scenario of this use case, a simulation run using the Blocking & Starvation Probability Method for bottleneck detection is conducted. The bottleneck ranking can then be compared to the bottleneck ranking of the System Sensitivity Analysis in order to validate this method.

Table 8: Settings for the validation of the BSP method

Bottleneck Detection Method		Blocking & Starvation Prob.
Objective		Bottleneck Ranking Validation
Comparative Basis		System Sensitivity Analysis
Simulation	Global Settings	Table 5
Settings	Line Structure	Demo Line
Method Settings	Period of Observation	$T_{obs} = 1 \text{ Day}$
	Prioritization Frequency	$f_{Prioritization} = 0,33 \text{ per Hour}$

Concerning the methodology, this scenario is conducted equally to the prior scenario. All repair requests are executed using a FIFO strategy, the bottleneck situation is detected 0.33 times per hour and the machine states of the last 24 hours are considered. If the requirements for a bottleneck, which were explained in chapter 2.2.3, are fulfilled, the algorithm assigns a priority of 1 to the machine. Otherwise a zero.

4.1.3.1 Results

The output of this scenario is a bottleneck ranking over time, which is shown in Figure 29. According to this method, the bottleneck shifts between OS 30 (red) and OS 40 (yellow). In contrast to the Active Period Method, this method does not provide information about the criticality of the other machines.

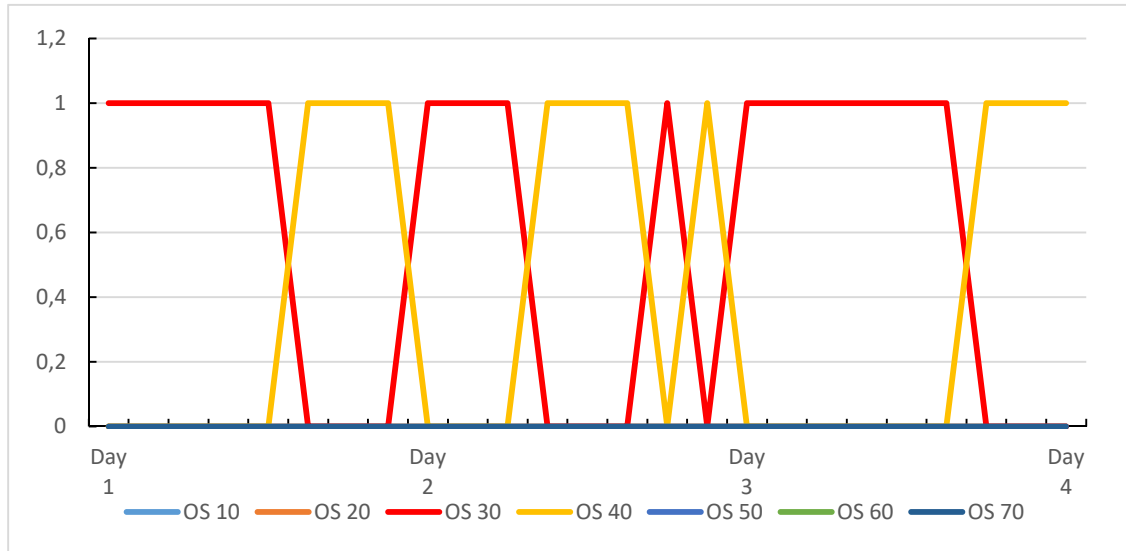


Figure 29: Shifting bottlenecks using the BSP method

In the same manner as for the Active Period Method, an average bottleneck ranking over the whole simulation period of 90 days is calculated. The result of this step, is shown in Figure 30.

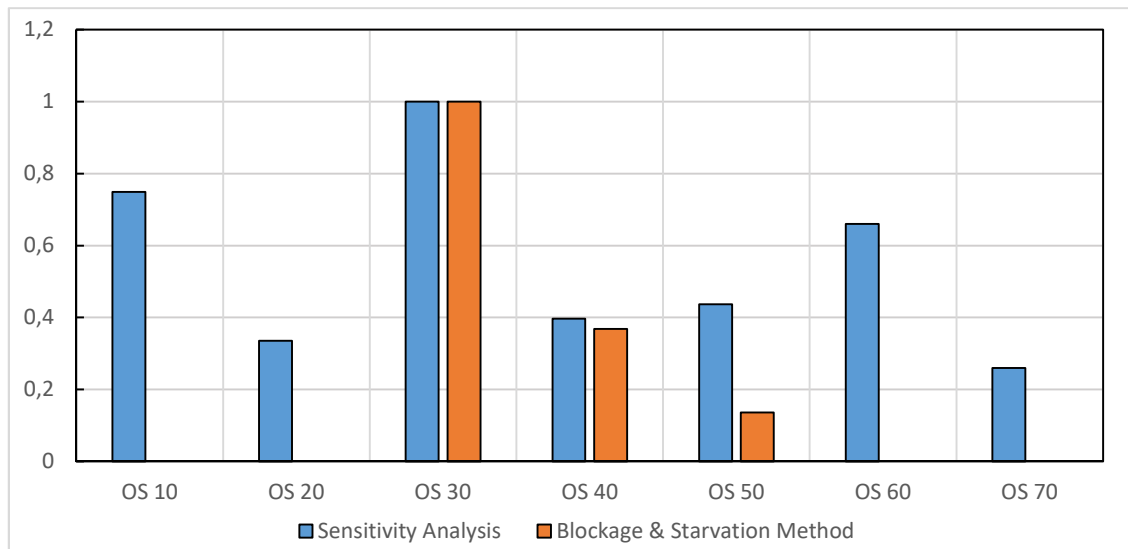


Figure 30: Comparison of the BSP method to the system sensitivity method

4.1.3.2 Discussion

The blockage and starvation probability method could only detect the primary bottleneck of the system, which is OS30. It failed for detecting OS 10 and OS 60 as the secondary and tertiary bottlenecks. Since it cannot be said, how significant the primary bottleneck is, this method still might be able to improve the system's performance and it will also be used for further investigations on a real-world production line.

4.1.4 Comparison of the Bottleneck Detection Methods

Figure 31 gives a direct comparison of the results of the three bottleneck detection methods used within this use case. As already mentioned, the System Sensitivity Analysis is considered to be the most reliable bottleneck detection method and is therefore used to validate the two data-driven methods.

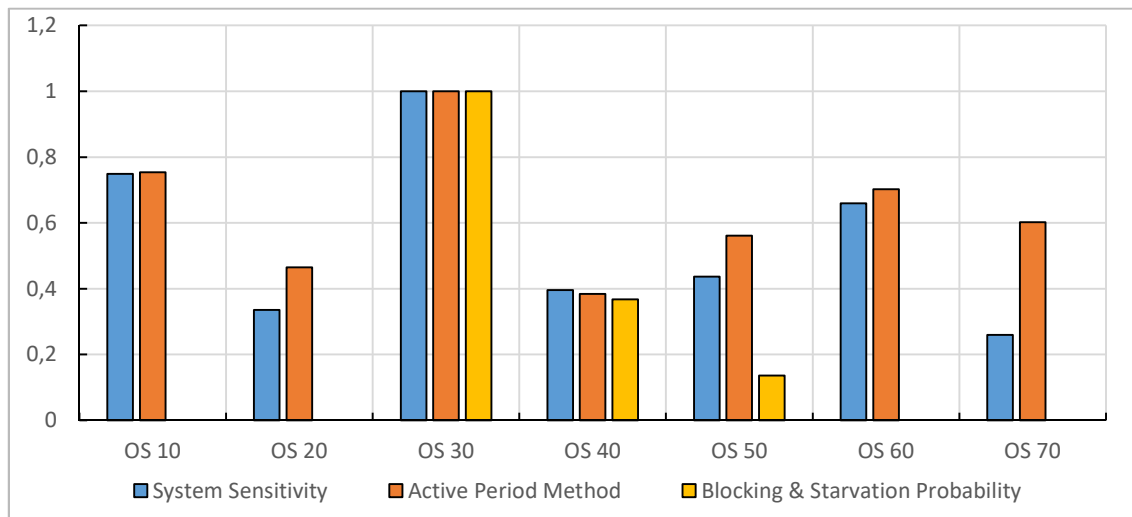


Figure 31: Bottleneck ranking of the Demo line using three different detection methods

The Active Period Method could detect the three most significant bottlenecks of the Demo production line, whereas the Blocking & Starvation Probability Method could only detect the primary bottleneck. Another drawback of the Blocking & Starvation Probability Method is, that it does not provide any information about the criticality of non-bottleneck machines. Especially in larger production systems, this is required in order to assign a certain priority to each machine of the system.

Table 9: Simulation input parameters for Industrial Use Case 1

Global Settings					
Repair Workers	2				
Remove Short Repair	true				
Simulation Period	90 Days				
Warm Up Period	7 Days				
Simulation Runs	15				
Machine Settings	C/T [s]	Puffer Capacity	TBF [h]	TTR [h]	Amount of Machines
AF200-Transferstrasse	30	∞	Sample	Sample	1
AF210-AF220-ZEN	1	10	Sample	Sample	1
AF210-BZs	33	1	Sample	Sample	1
AF220-BZs	33	3	Sample	Sample	7
AF230-ZEN	1	40	Sample	Sample	1
AF230-BZs	33	1	Sample	Sample	6
AF240-Transferstrasse	30	25	Sample	Sample	1
AF250-Transferstrasse	30	10	Sample	Sample	1
AF255-ZEN	1	20	Sample	Sample	1
AF255-BZs	33	1	Sample	Sample	2
AF260-ZEN	1	10	Sample	Sample	1
AF260-BZs	33	1	Sample	Sample	6
AF270	30	20	Sample	Sample	1
AF280	30	20	Sample	Sample	1
AF300-Transferstrasse	30	6	Sample	Sample	1
AF310-AF320-ZEN	1	20	Sample	Sample	1
AF310-AF320-BZs	33	20	Sample	Sample	4
AF325	27	48	Sample	Sample	1
AF325P	33	1	Sample	Sample	1
AF330	30	20	Sample	Sample	1
AF335	33	20	Sample	Sample	2
AF335-HB	33	25	Sample	Sample	1
AF340	30	10	Sample	Sample	1
AF345-ZEN	1	10	Sample	Sample	1
AF345-W	33	1	Sample	Sample	2
AF350	30	5	Sample	Sample	1

4.2.1 Heuristic Prioritization

The heuristic prioritization methods, aim for imitating the actual prioritization behaviour of the maintenance staff on the plant floor. Three different heuristic approaches will be used within this use case:

- Part-Out-Part-Out Times
- Availability
- Availability + Redundancy

As for the other methods, the baseline for the heuristic methods will be a FIFO strategy. Since all heuristic methods work with long-term averages, they are considered to be static. This means that the time a machine is already waiting for repair, cannot influence the priority. Therefore also for this scenario a Priority Increase Factor is used, which is set to $p_{inc} = 0,2 \frac{1}{Hour}$ for all heuristic methods.

4.2.1.1 Prioritizing using Part-Out-Part-Out Times

As already discussed in chapter 2.2.2, in the engine manufacturing plant, all Part-Out-Part-Out times are recorded in order to evaluate the performance of machines. The longer this “true cycle time” is, the more critical is a machine. Even this method cannot detect bottlenecks at all, it is used for comparison to other prioritization strategies.

Table 10: Settings for prioritizing using Part-Out-Part-Out Times

Prioritization Method		Part-Out-Part-Out Time
Objective		Throughput Improvement
Comparative Basis		FIFO Policy
Simulation Settings	Global Settings	Table 9
	Line Structure	Line 1
Method Settings	Period of Observation	–
	Priority Increase Factor	$p_{inc} = 0,2 \text{ per Hour}$
	Prioritization Frequency	–
	Reduce Redundant	<i>false</i>

The simulation requires an input file, where the priorities of all machines according to their Part-Out-Part-Out times are listed. Figure 33 shows the priority ranking

where the machine with the longest Part-Out-Part-Out time is assigned the priority of 1. The Part-Out-Part-Out times of the redundant operation sequences have been reduced by the number of redundant machines. Since the Part-Out-Part-Out times of the machines in a production line cannot differ significantly in a long term, also the priorities do not differ a lot.

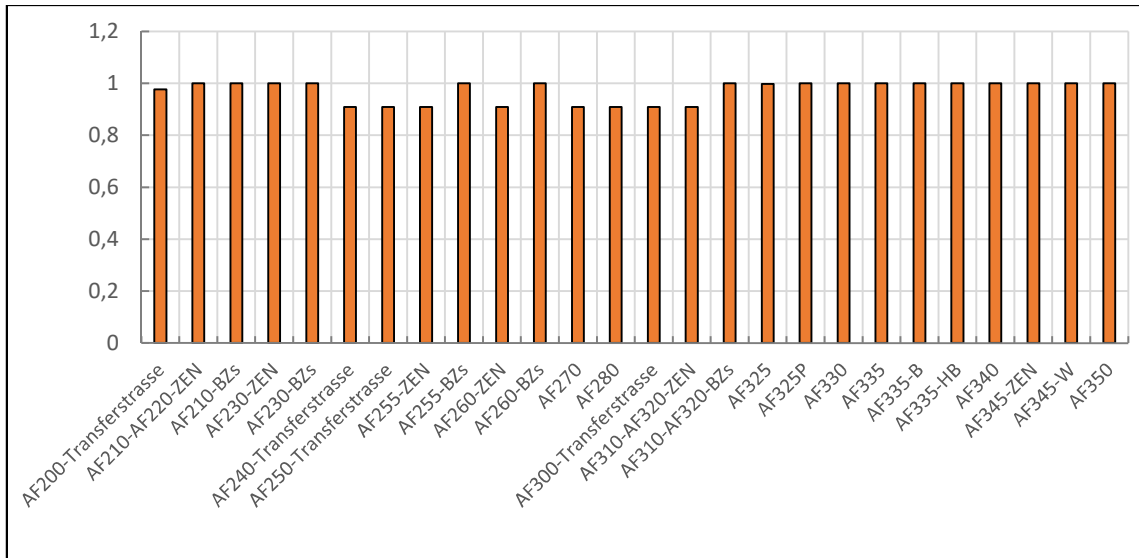


Figure 33: Priority ranking using Part-Out-Part-Out times

Results

The output of the simulation is the average throughput per day of 30 simulation runs of 90 days. This average throughput can be compared to the output of the same simulation using a FIFO service policy.

The performance of this prioritization policy compared to FIFO, each with a 95% confidence interval is shown in Figure 34.

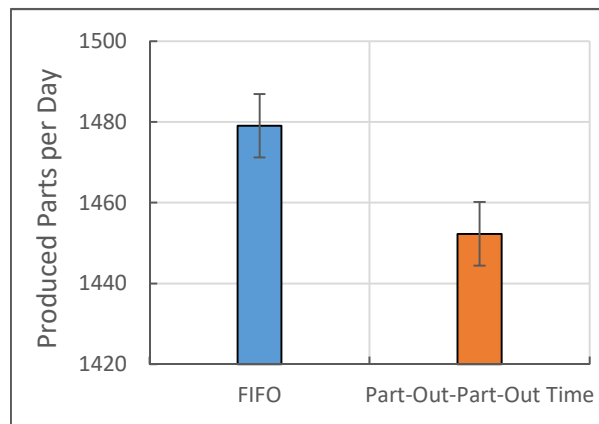


Figure 34: Average throughput per day - FIFO vs. Part-Out-Part-Out Time

Discussion

The reason why this policy performs worse than a FIFO policy is that all redundant operation sequences have a high priority, whereas some of the single machines have a low priority. This means that the redundant machines may be, due to their long cycle times, bottlenecks for the system. But this method does not consider, that a breakdown of a redundant machine is not as critical as a breakdown of a single machine. Furthermore, the proposed method is questionable concerning its use for prioritization, since it cannot detect bottlenecks and the priorities of all machines are almost equal.

4.2.1.2 Prioritizing using Availabilities

The second heuristic approach uses availability values of the machines for maintenance task prioritization.

Table 11: Settings for prioritizing using availabilities

Prioritization Method		Availabilities
Objective		Throughput Improvement
Comparative Basis		FIFO Policy
Simulation Settings	Global Settings	Table 9
	Line Structure	Line 1
Method Settings	Period of Observation	–
	Priority Increase Factor	$p_{inc} = 0,2 \text{ per Hour}$
	Prioritization Frequency	–
	Reduce Redundant	<i>false</i>

The machine with the lowest availability is assigned a priority of 1. Figure 35 shows the priority ranking using average availability values over 90 days.

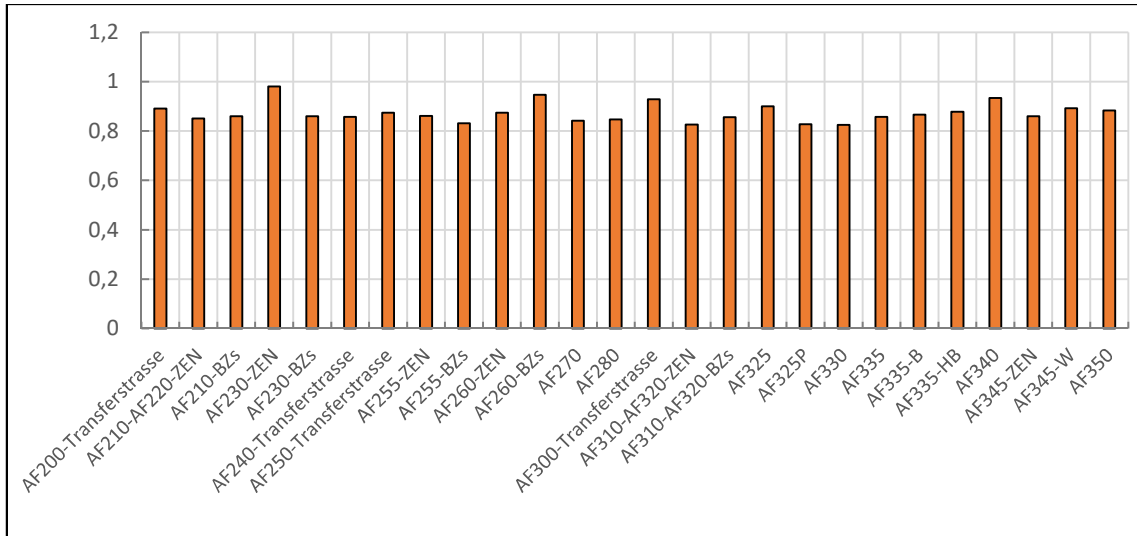


Figure 35: Priority ranking using availability values

Results

Using the priorities from Figure 35 will not bring an improvement compared to a FIFO service policy. The average throughput per day is shown in Figure 36.

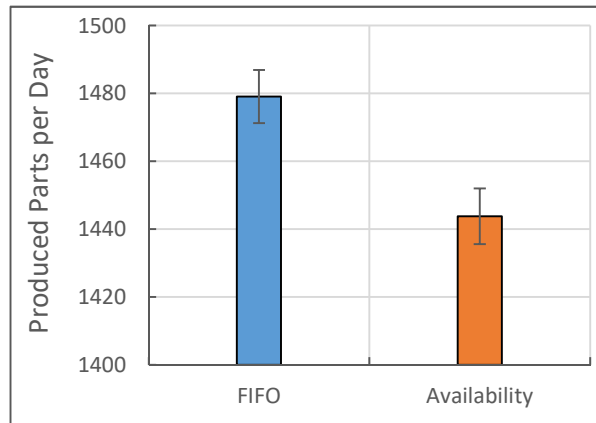


Figure 36: Average throughput per day - FIFO vs. Availability

Discussion

The availabilities of machines can whether give any statement about the importance of machines, nor can they be used for bottleneck identification. Because of that, and because the lower breakdown severity of redundant machines is not considered, no throughput increment could be achieved. This will be taken into account for the next scenario.

4.2.1.3 Prioritizing using Redundancy & Availability

The last method utilizes the average availability values from the last method and decreases the priorities, depending on if there are redundant machines in an operating sequence. The priority of each operating sequence is reduced by its amount of machines.

Table 12: Settings for prioritizing using availabilities and redundancies

Prioritization Method		Availability + Redundancy
Objective		Throughput Improvement
Comparative Basis		FIFO Policy
Simulation Settings	Global Settings	Table 9
	Line Structure	Line 1
Method Settings	Period of Observation	–
	Priority Increase Factor	$p_{inc} = 0,2 \text{ per Hour}$
	Prioritization Frequency	–
	Reduce Redundant	<i>true</i>

The simulation requires an input file, where each machine's priority based on availabilities is reduced by the number of redundant machines. Figure 37 shows the adapted priority ranking, which will be used for this scenario.

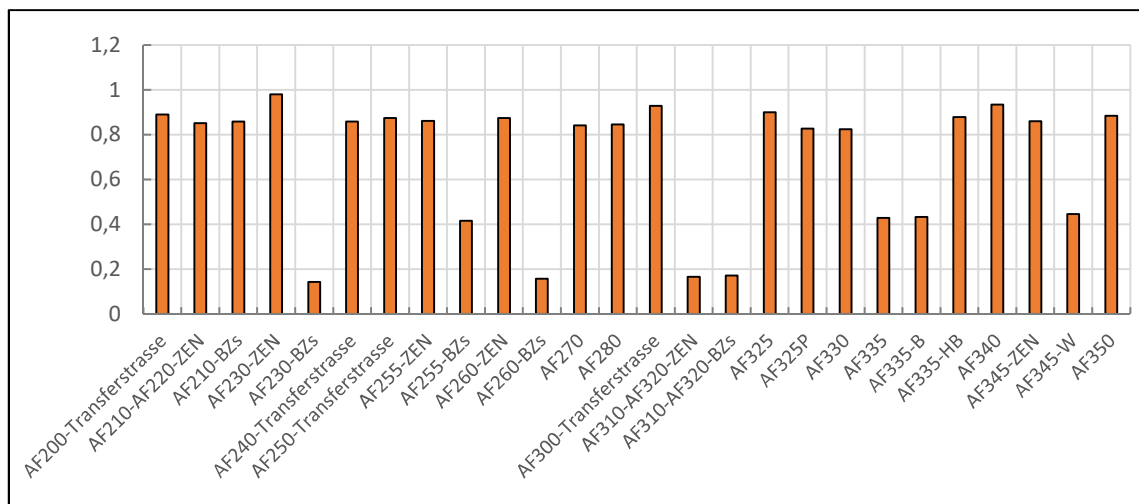


Figure 37: Priority ranking using availability values with decreased priorities for redundant machines

Results

Dividing the priority of redundant machines by the amount of redundant machines improves the availability approach and generates a throughput increment of 5 % compared to a FIFO strategy.

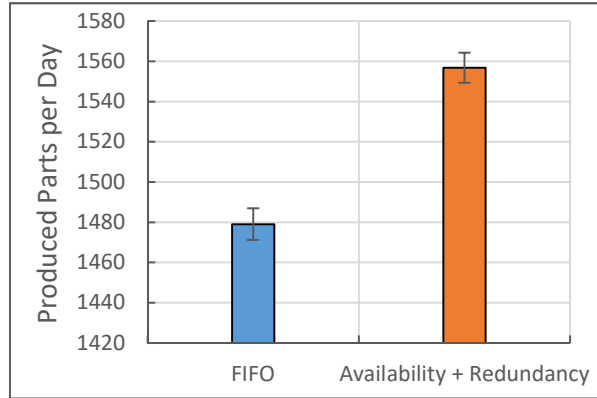


Figure 38: Average throughput per day - FIFO vs. Availability & Redundancy

Discussion

Since the prior scenario, which only used availability values, could not bring any throughput improvements, the good result of this scenario can be traced on the priority reduction of redundant machines.

4.2.1.4 Comparison of Heuristic Methods

Figure 39 shows the average throughput increment of the 3 heuristic methods compared to a FIFO policy with a 95 % confidence interval.

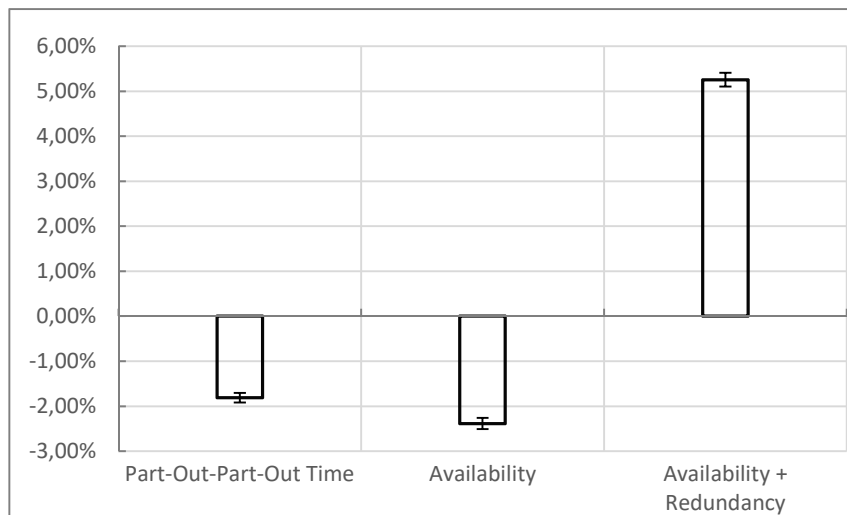


Figure 39: Average throughput increment of heuristic methods compared to a FIFO policy

The only heuristic approach which generates a throughput increment, was the approach based on availabilities and redundancies. But since prioritizing using only availability values could not bring any improvement, the good result is only due to the reduction of the priority of redundant machines. The next sub-chapter investigates the potential of bottleneck prioritization methods.

4.2.2 Bottleneck Prioritization

In contrast to the heuristic methods, the following methods are based on the strategy of reducing the downtime of bottleneck machines in order to improve the system's performance. The bottleneck detection algorithms use real-time production to determine the actual bottleneck situation and enable a dynamic prioritization of machines. The following bottleneck detection methods are used within this use case:

- Active Period Method
- Blocking & Starvation Probability Method

For both methods, several simulations using different input parameters are conducted in order to find optimal settings for the algorithm concerning the Priority Increase Factor and the Period of Observation.

4.2.2.1 Active Period Method

The objective of the simulations using the Active Period Method is to improve the total throughput by prioritizing bottleneck machines and to find optimal settings for the algorithm.

The following experiments are conducted for the Active Period Method:

- Initial Experiment: Track the dynamic behaviour of the system
- Variation of the Period of Observation: Find optimal settings for T_{obs}
- Static Priorities: Use Average Active Period over 90 days
- Variation of the Priority Increase Factor: Find optimal settings for p_{inc}

Initial Experiment

To have a first result of how dynamic the system behaves, a simulation using the following scenario is conducted:

Table 13: Settings for the initial experiment – APM

Prioritization Method		Active Period Method
Objective		Throughput Improvement
Comparative Basis		FIFO Policy
Simulation Settings	Global Settings	Table 9
	Line Structure	Line 1
Method Settings	Period of Observation	$T_{obs} = 48 \text{ Hours}$
	Priority Increase Factor	$p_{inc} = 0,1 \text{ per Hour}$
	Prioritization Frequency	$f_{prioritization} = 0,2 \text{ per Hour}$
	Reduce Redundant	<i>true</i>

RESULTS

Figure 40 shows for the most significant bottleneck machines of Line 1, how the bottlenecks are shifting over a period of 3 days. In the example of Figure 40, the primary bottleneck is shifting between machine AF220-Bz, AF260-Bz and AF345-W. This behaviour continues over the whole simulation run of 90 days. This indicates that a dynamic prioritization could improve the system’s performance.

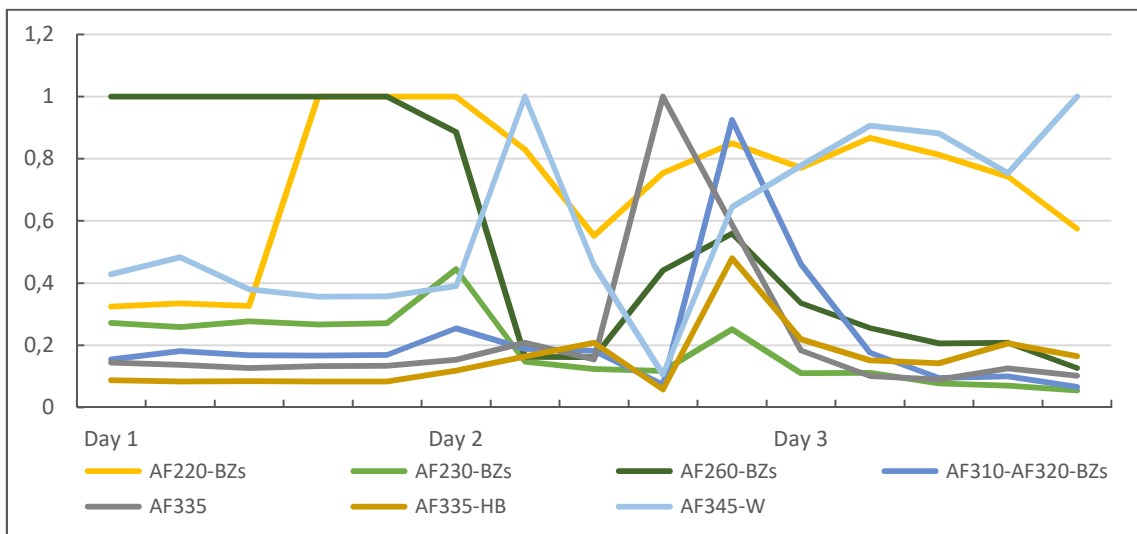


Figure 40: Excerpt of the priorities over a period of 3 days using the AAP method for Line 1

Using the priorities over time, of which an excerpt is shown in Figure 40, will bring a throughput improvement. Figure 41 shows the average throughput per day and the 95 % confidence intervals of 30 simulation runs over 90 days in comparison to the same simulation using a FIFO service policy.

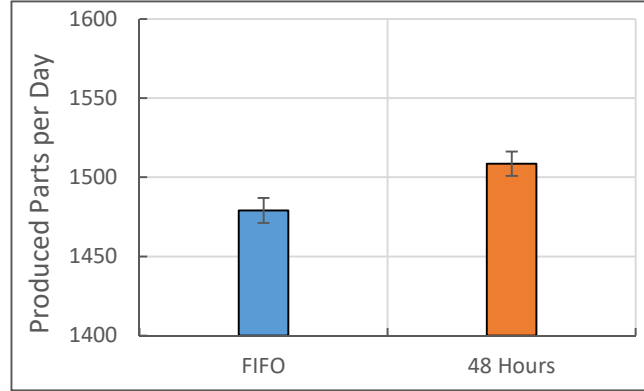


Figure 41: Average throughput per day - FIFO vs. APM (48 Hours)

DISCUSSION

The first simulation brought a throughput increment of 2 %. Since there are two input parameters for the algorithm which influence the result, further simulations are necessary to optimize the performance. The period of observation has the biggest influence on the output of the algorithm. Therefore further scenarios using different periods of observation are conducted.

Variation of the Period of Observation

This experiments aims for optimizing the algorithm by varying the input parameter T_{obs} . The other parameters will be the same as for the prior experiment.

Table 14: Settings for the variation of T_{obs} - APM

Prioritization Method		Active Period Method
Objective		Throughput Improvement
Comparative Basis		FIFO Policy
Simulation Settings	Global Settings	Table 9
	Line Structure	Line 1
Method Settings	Period of Observation	$T_{obs} = variable$
	Priority Increase Factor	$p_{inc} = 0,1 \text{ per Hour}$
	Prioritization Frequency	$f_{Prioritization} = 0,2 \text{ per Hour}$
	Reduce Redundant	$true$

RESULTS

Figure 42 shows the average throughput increment and the 95% confidence intervals using different periods of observation. Short periods of observation enable the algorithm to detect short-term bottlenecks and vice versa. The maximum is found using a period of observation of 7 days.

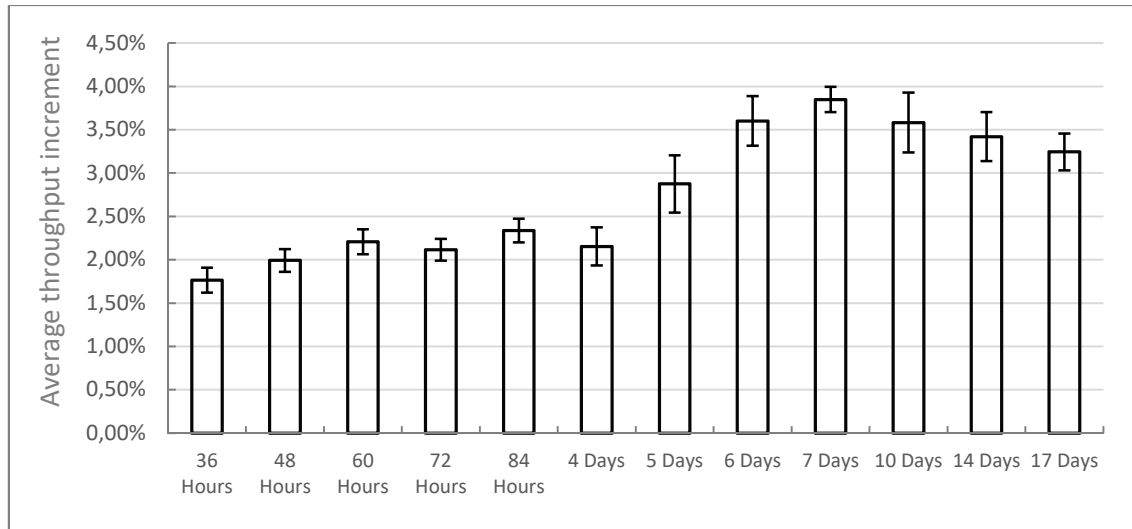


Figure 42: Average throughput increment using different periods of observation

DISCUSSION

The results indicate that these short-term bottlenecks do not significantly influence the system's performance, since a prioritization of those machines does not improve the system's throughput. Since the validation of the algorithm brought a good match with the system sensitivity analysis for the average priorities over 90 days, the same scenario will be investigated here.

Static Priorities

This scenario is a static bottleneck detection scenario, since the algorithm does not determine the actual bottleneck situation during a simulation run. Instead of that, the average bottlenecks over a simulation run of 90 days are calculated and used as an input for the static scenario. This is done, using the priorities over time of the $T_{obs} = 7 \text{ Days}$ scenario, and calculate average priorities for each machine. The other parameters are the same as for the prior experiment.

Table 15: Settings for the static bottleneck detection - APM

Prioritization Method		Active Period Method
Objective		Throughput Improvement
Comparative Basis		FIFO Policy
Simulation Settings	Global Settings	Table 9
	Line Structure	Line 1
Method Settings	Period of Observation	-
	Priority Increase Factor	$p_{inc} = 0,1 \text{ per Hour}$
	Prioritization Frequency	$f_{Prioritization} = 0 \text{ per Hour}$
	Reduce Redundant	<i>true</i>

The average priorities over a period of 90 days are shown in Figure 43. According to this result, the system has three significant bottlenecks.

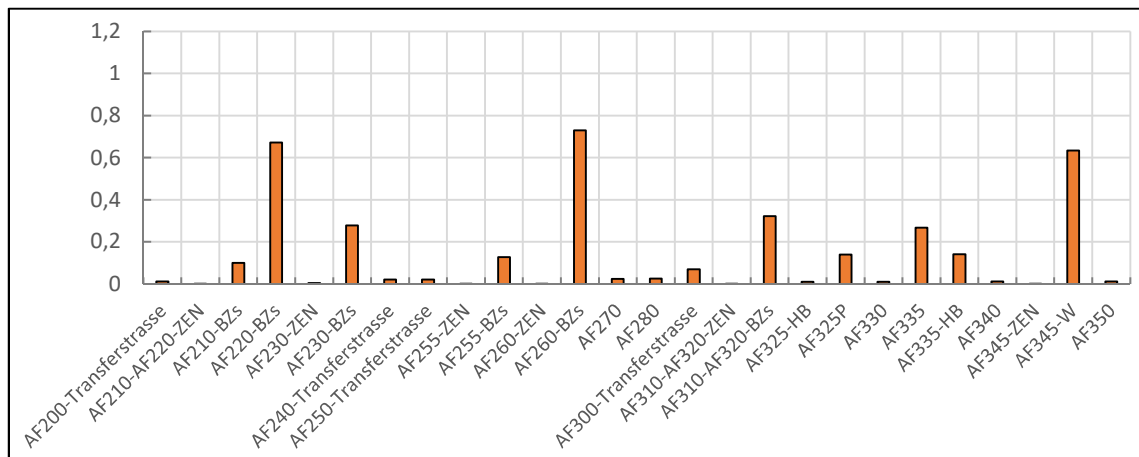


Figure 43: Static AAP priorities before redundant machines priorities are reduced

In order to consider the lower breakdown severity of redundant machines, the priorities have to be divided by the number of redundancies per operating sequence. The result of that modification is shown in Figure 44. The operating sequence with the highest priority is AF345-W, which is still a redundant operating sequence.

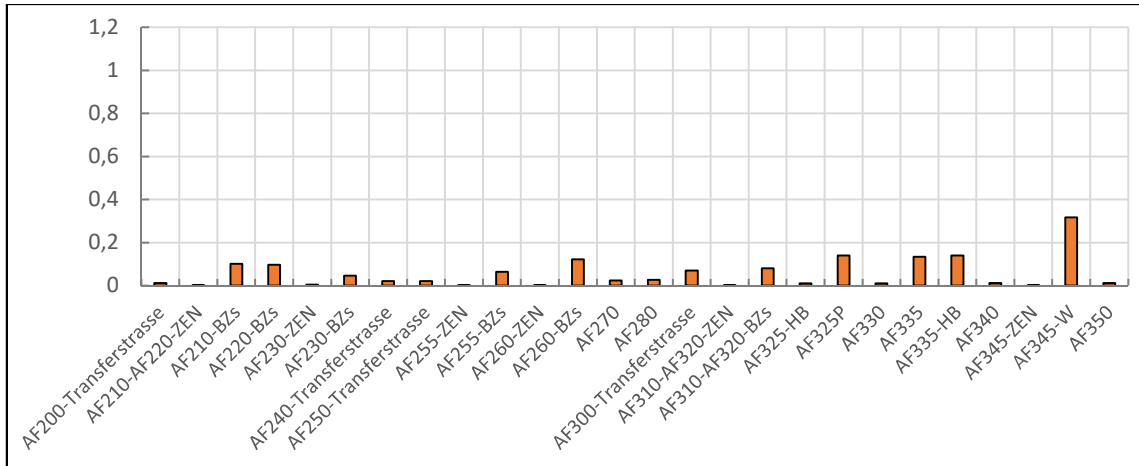


Figure 44: Static AAP priorities after redundant machines priorities are reduced

RESULTS

Using a static Active Period Method priority ranking will improve the throughput by 5 % compared to a FIFO policy:

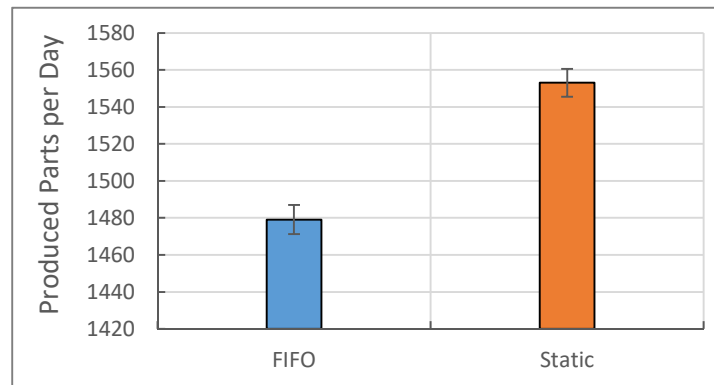


Figure 45: Average throughput per day FIFO vs. Static APM

DISCUSSION

The simulation scenarios with different periods of observation have shown that the system does behave dynamically, but a dynamic prioritization does not improve the throughput as much as a static prioritization does.

The Priority Increase Factor p_{inc} is another input parameter which has an impact on the result. The higher p_{inc} is chosen, the more the prioritization policy will behave like a FIFO policy. This is because p_{inc} increases the priority depending on the time of the breakdown. The next experiments aim for finding an optimum value for p_{inc} .

Variation of the Priority Increase Factor

This scenario uses the same static APM priorities as the prior experiment, since this priority ranking performed best. The Priority Increase Factor will be varied in order to find an optimal setting for this parameter.

Table 16: Settings for the variation of p_{inc} - APM

Prioritization Method		Active Period Method
Objective		Throughput Improvement
Comparative Basis		FIFO Policy
Simulation Settings	Global Settings	Table 9
	Line Structure	Line 1
Method Settings	Period of Observation	–
	Priority Increase Factor	$p_{inc} = variable$
	Prioritization Frequency	$f_{Prioritization} = 0 \text{ per Hour}$
	Reduce Redundant	$true$

RESULTS

The average throughput increment compared to a FIFO policy and the 95 % confidence interval are shown in the following figure:

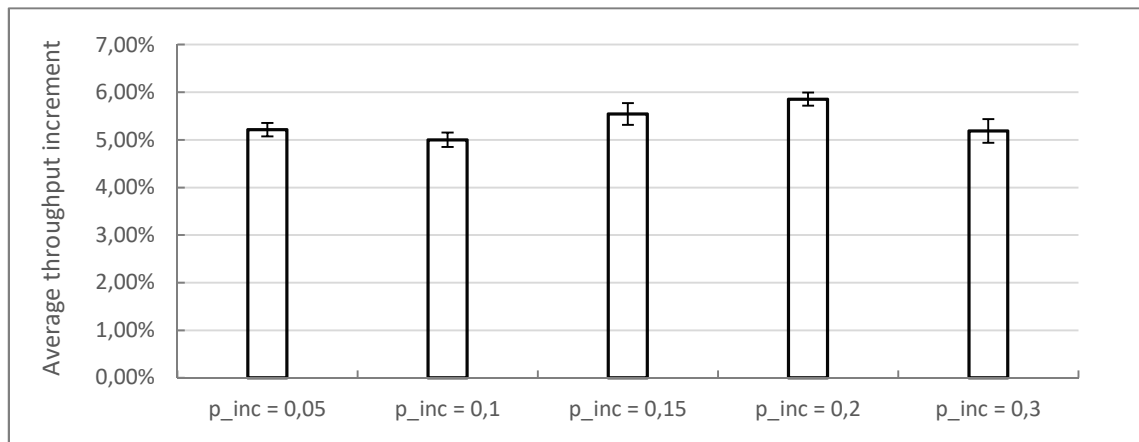


Figure 46: Average throughput increment of different p_{inc} using a static AAP priority ranking

DISCUSSION

Increasing the Priority Increase Factor to $p_{inc} = 0,2$ could improve the system's performance by 5,9 % compared to a FIFO service policy.

Conclusion – Active Period Method

Since the bottleneck detection algorithm shall be applied on different production lines, it is necessary to find out, if there is a global optimum which achieves good results on different production lines. Therefore the results of the different experiments in dependence of the two input parameters are visualized in Figure 47 using a bubble chart, where the diameter of a bubble is representing the average throughput increment of the simulation scenario compared to a FIFO policy.

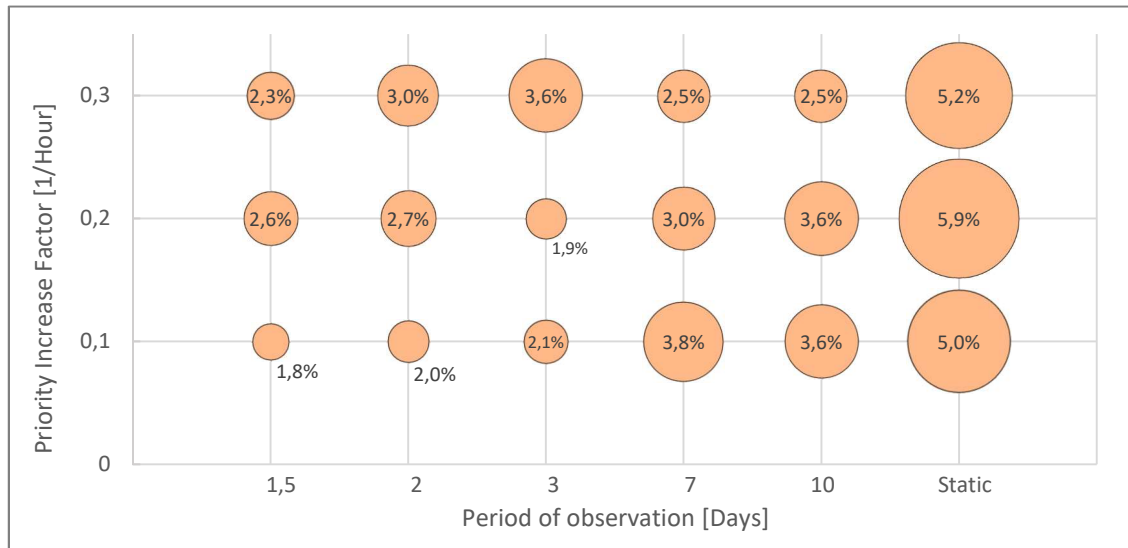


Figure 47: Average throughput increment for different input parameters on Line 1 compared to a FIFO service policy

The best result for Line 1 is achieved using a static Active Period Method priority ranking and a Priority Increase Factor of $p_{inc} = 0,2$. In 4.3, the same simulations will be conducted for a different production line in order to verify if this setting also performs best in a different production system.

4.2.2.2 Blockage & Starvation Probability Method

This method is the second which shall be investigated within this simulation study in order to evaluate its performance for throughput improvement by corrective maintenance task prioritization. The following experiments are conducted:

- Initial Experiment: Track the dynamic behaviour of the system
- Variation of the Period of Observation: Find optimal settings for T_{obs}

Initial Experiment

In the first experiment using the BSP method, the dynamic behaviour of the system and the ability of the algorithm to detect shifting bottlenecks shall be analysed. The experiment is done using the following input parameters:

Table 17: Settings for the initial experiment - BSP

Prioritization Method		Blocking & Starvation Prob.
Objective		Throughput Improvement
Comparative Basis		FIFO Policy
Simulation Settings	Global Settings	Table 9
	Line Structure	Line 1
Method Settings	Period of Observation	$T_{obs} = 7 \text{ Days}$
	Priority Increase Factor	$p_{inc} = 0,1 \text{ per Hour}$
	Prioritization Frequency	$f_{prioritization} = 0,2 \text{ per Hour}$
	Reduce Redundant	<i>true</i>

RESULTS

Figure 48 shows for the most significant bottlenecks of the system, how the bottlenecks are shifting according to the blockage and starvation probability method.

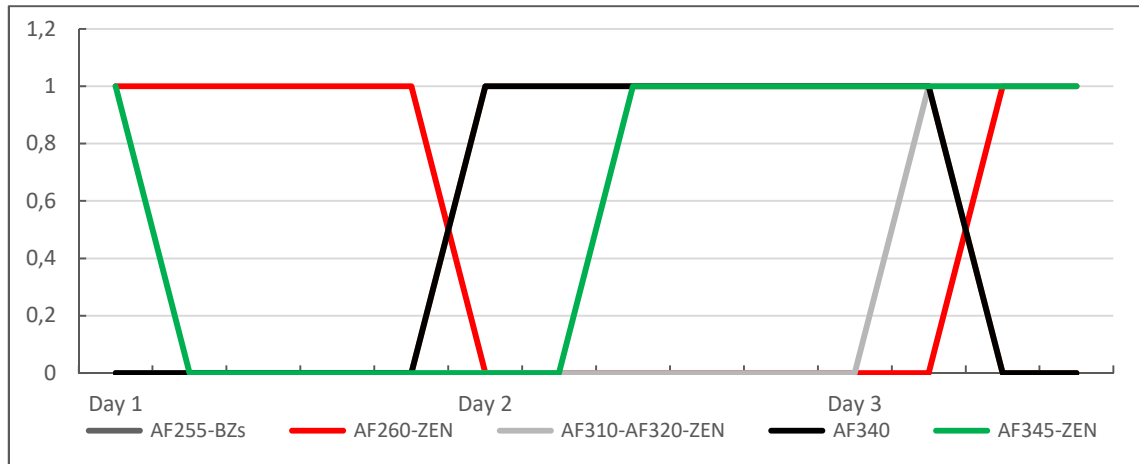


Figure 48: Excerpt of the priority over time using the BSP method

Using this dynamic bottleneck ranking will generate a throughput increment of 3,5 % compared to a FIFO policy:

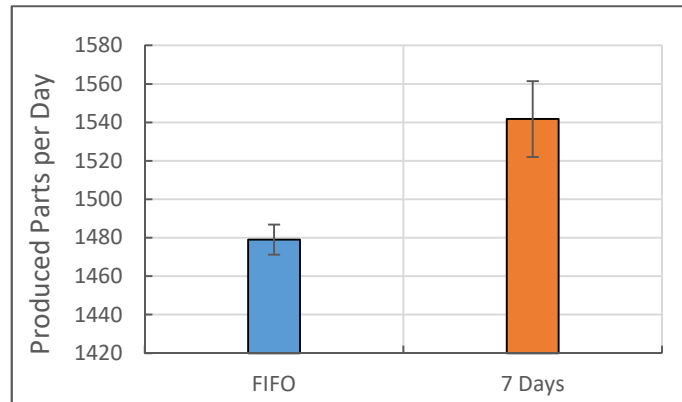


Figure 49: Average throughput per day - FIFO vs. BSP (7 Days)

DISCUSSION

Even also this method detects shifting bottlenecks, the chart looks very different to the one of the Active Period Method. This is because this method can only provide information about if a machine is a bottleneck or not. If a machine is a bottleneck, it is assigned a priority of 1. Else, it is assigned a priority of zero and no further statement about the criticality of that machine is possible. Furthermore, this method achieved 1,5 % less than the APM, using the same input parameters. Still further experiments are conducted in order to optimize this method.

Variation of the Period of Observation

Since the Active Period Method performed best using a static priority ranking, the same scenario will be conducted for the BSP method. Furthermore, several experiments where the Period of Observation is varied between 8 Hours and 9 Days are conducted.

Table 18: Settings for the variation of T_{obs} - BSP

Prioritization Method		Blocking & Starvation Prob.
Objective		Throughput Improvement
Comparative Basis		FIFO Policy
Simulation Settings	Global Settings	Table 9
	Line Structure	Line 1
Method Settings	Period of Observation	$T_{obs} = variable$
	Priority Increase Factor	$p_{inc} = 0,1 \text{ per Hour}$
	Prioritization Frequency	$f_{Prioritization} = 0,2 \text{ per Hour}$
	Reduce Redundant	$true$

As for the Active Period Method, the static priorities are calculated using the average priorities over 90 days from the scenario $T_{obs} = 7 \text{ Days}$; $p_{inc} = 0,1 \frac{1}{\text{Hour}}$.

Figure 50 shows the average priority ranking of the BSP method before the redundant machine's priority is divided by the number of machines:

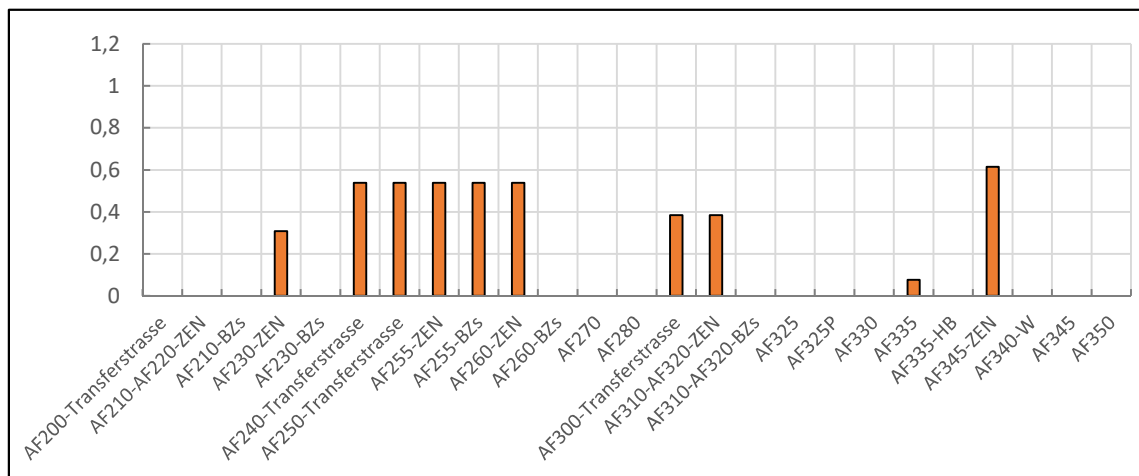


Figure 50: Static priorities of the BSP method before redundancies are reduced

This result is very different to the priority ranking from the AAP method. The AAP method detected three significant bottlenecks which were all redundant operating sequences. In contrast to that, the BSP method detects no significant bottleneck, but six machines which are equally often the bottleneck of the system. Reducing the priority of the operating sequences by the number of its machines, will generate the following priority ranking:

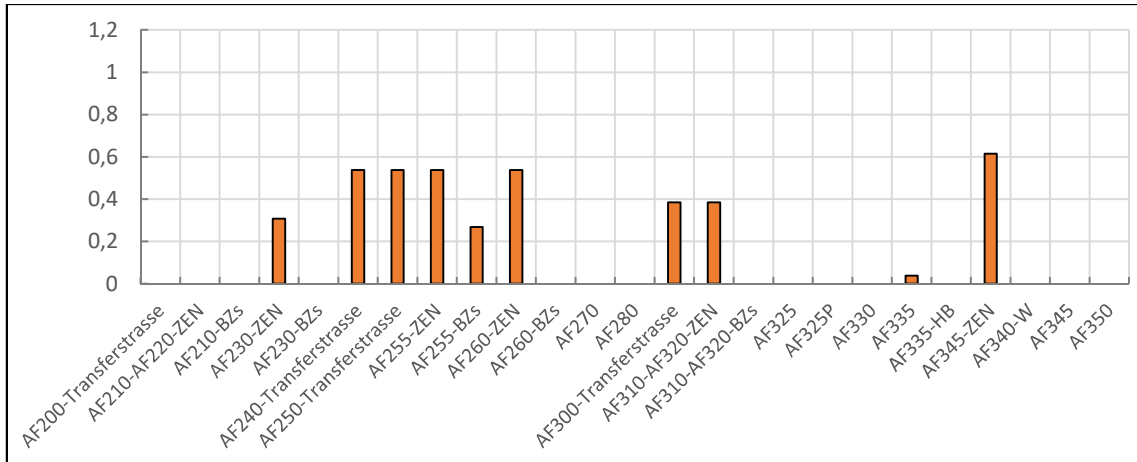


Figure 51: Static priorities of the BSP method after redundant machine's priorities are reduced

RESULTS

Using the priority ranking shown in Figure 51, will improve the system's throughput by almost 5%. The results for different Periods of Observation and for the static priority ranking are shown in Figure 52.

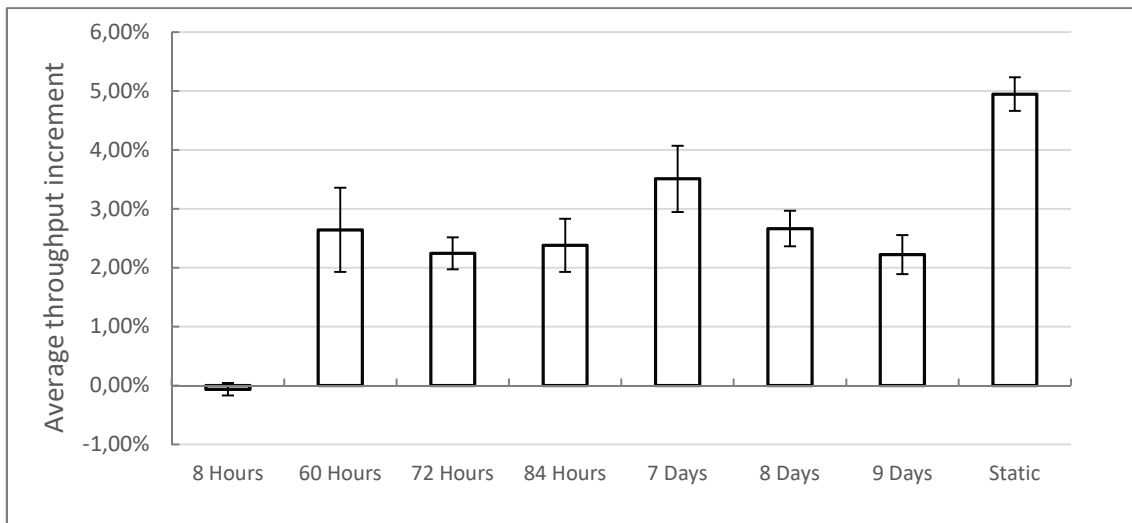


Figure 52: Average throughput increment and 95% confidence intervals for different periods of observation for the BSP method compared to a FIFO service policy

DISCUSSION

The best performance improvement was achieved using a static priority ranking. As before for the Active Period Method, a dynamic bottleneck detection performed better than FIFO, but worse than a static priority ranking.

4.2.2.3 Conclusion – Blockage & Starvation Probability Method

The best result using the BSP method achieved a throughput increment over a FIFO service policy of almost 5 %. A drawback of this method is that it cannot provide information about the criticality of non-bottleneck machines. This was shown in Figure 48, where the priorities over time are pictured. This behaviour is problematic for larger production systems, since the proposed method can only prioritize very few machines of the system. Most of the machines will have a priority of zero, wherefore those machines can only be serviced using a FIFO policy. Furthermore, the BSP method requires information about the line structure and is therefore not suitable for flexible production systems or job shop layouts. Because of its drawbacks concerning usability, validity and performance, the second industrial use case will be conducted using only the Active Period Method.

4.3 Industrial Use Case 2

The second use case is for another production line of an engine manufacturing plant. The aim for this use case is to verify, if the algorithm settings which performed best in the first use case, also perform best on a different production line. If this could be verified, the algorithm could be applied on all production lines in the plant, without the need for any modifications.

The structure of Line 2 is shown in Figure 53.

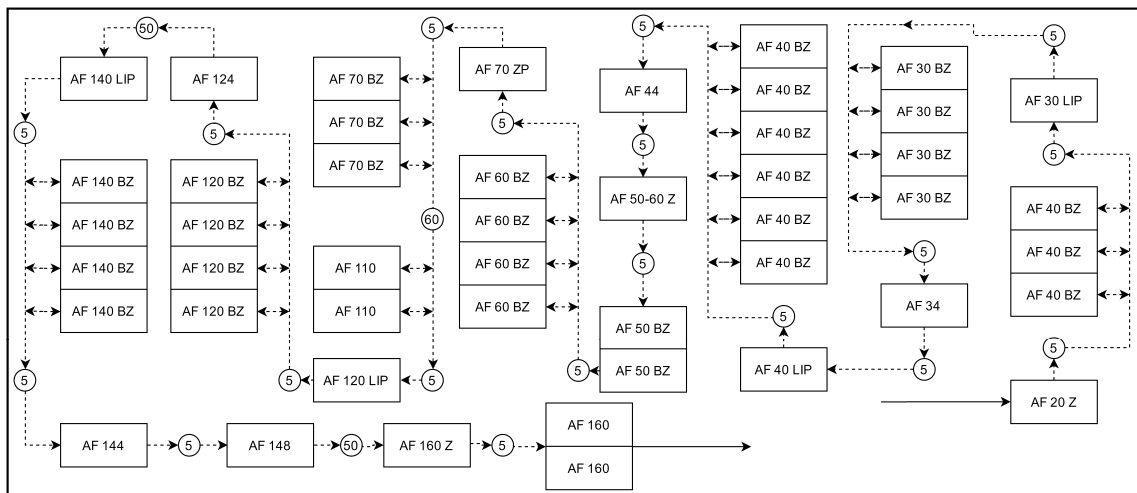


Figure 53: Structure of Line 2

As the line of the first use case, Line 2 is a highly automated production line. In contrast to Line 1, this line is even more balanced since the cycle times of all

machines are equal. Furthermore the buffer capacities are smaller compared to Line 1. All input parameters for the simulation can be seen in Table 19.

Table 19: Simulation input parameters for Industrial Use Case 2

Global Settings					
Repair Workers	2				
Remove Short Repair	true				
Simulation Period	90 Days				
Warm Up Period	7 Days				
Simulation Runs	15				
Machine Settings	C/T [s]	Puffer Capacity	TBF [h]	TTR [h]	Amount of Machines
AF20-Zen	27	∞	Sample	Sample	1
AF20-BZ	27	5	Sample	Sample	3
AF30-LIP	27	5	Sample	Sample	1
AF30-BZ	27	5	Sample	Sample	4
AF34	27	5	Sample	Sample	1
AF40-LIP	27	5	Sample	Sample	1
AF40-BZ	27	5	Sample	Sample	6
AF44	27	5	Sample	Sample	1
AF50-60	27	5	Sample	Sample	1
AF50-BZ	27	5	Sample	Sample	2
AF60-BZ	27	5	Sample	Sample	4
AF70-ZP	27	5	Sample	Sample	1
AF70	27	5	Sample	Sample	3
AF110	27	60	Sample	Sample	2
AF120-LIP	27	5	Sample	Sample	1
AF120-BZ	27	5	Sample	Sample	4
AF124	27	5	Sample	Sample	1
AF140-LIP	27	50	Sample	Sample	1
AF140-BZ	27	5	Sample	Sample	4
AF144	27	5	Sample	Sample	1
AF148	27	5	Sample	Sample	1
AF160-ZP	27	50	Sample	Sample	1
AF160	27	5	Sample	Sample	2

4.3.1 Variation of Input Parameters

The best results for Line 1 were achieved using the Active Period Method. Furthermore the Active Period Method provided reliable results, since it could identify the bottleneck situation correctly in the validation on the Demo production line. Therefore this method shall now be used for a different production line in order to identify if there is a global optimum for the input parameters which could be applied to any production line. The input parameters which influence the performance of the algorithm are:

Table 20: Settings for the variation of T_{obs} and p_{inc} – APM

Prioritization Method		Active Period Method
Objective		Verification of Global Optimum
Comparative Basis		FIFO Policy
Simulation Settings	Global Settings	Table 19
	Line Structure	Line 2
Method Settings	Period of Observation	$T_{obs} = variable$
	Priority Increase Factor	$p_{inc} = variable$
	Prioritization Frequency	$f_{Prioritization} = 0,2 \text{ per Hour}$
	Reduce Redundant	$true$

Results

To verify the optimum which was found at Line 1, the same scenarios are conducted for Line 2. The results of different scenarios are shown in a bubble chart in Figure 54.

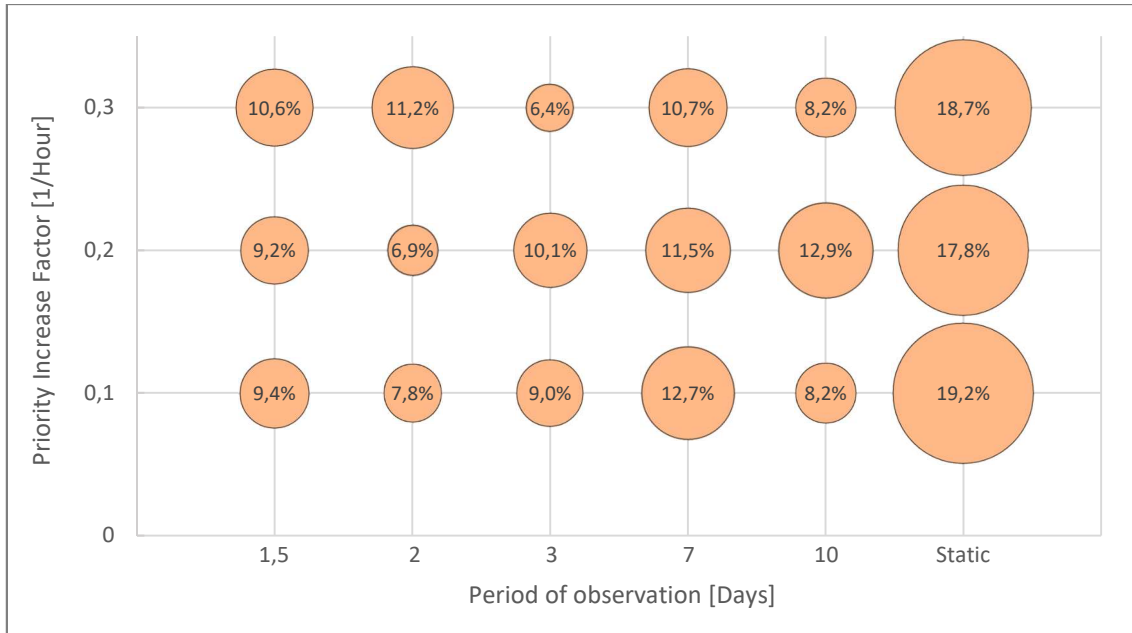


Figure 54: Average throughput increment for different input parameters on Line 2 compared to a FIFO service policy

As for Line 1, the best results were achieved using a static AAP priority ranking. This ranking is shown in Figure 55.

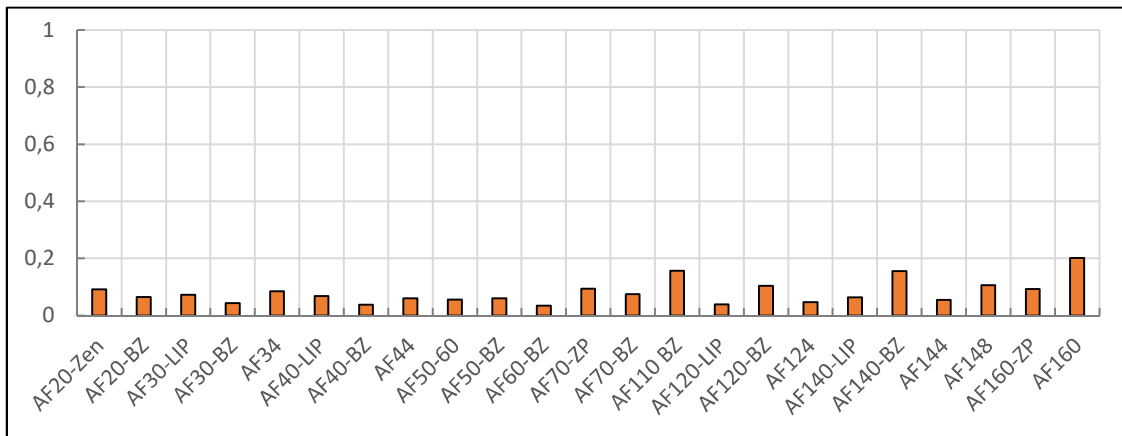


Figure 55: Static AAP priority ranking for Line 2 after redundant machine's priority was reduced

Discussion

Compared to the first industrial use case, Line 2 is even more well-balanced. No significant bottleneck was found and the differences between the priorities of the machines are marginal. This result coincides with the fact, that for Line 2 all machines have equal cycle times and therefore bottlenecks can only occur due to machine breakdowns.

The fact, that the average throughput increment using a prioritization strategy is much higher than for the first use case, can be explained by taking the number of simultaneous breakdowns and the amount of repair workers into account.

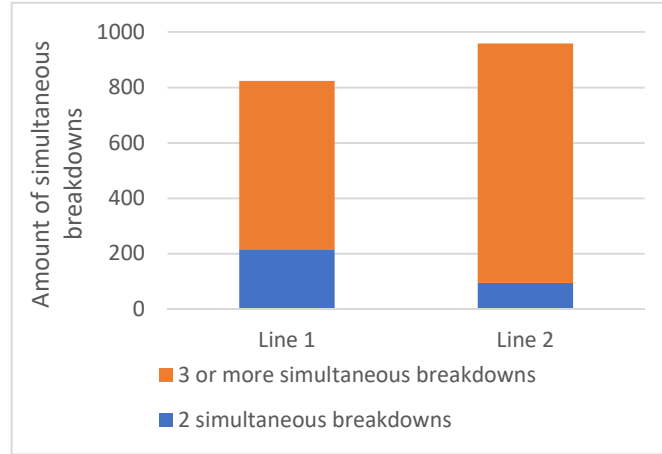


Figure 56: Simultaneous breakdowns Line 1 - Line 2

Figure 56 gives a comparison of Line 1 and Line 2 concerning how often simultaneous breakdowns occurred during the 90 days of simulation. Line 1 had more situations, where a repair request could be processed immediately because only one machine had a failure. Figure 56 shows, that Line 2 had more situations where three or more machines were down simultaneously. In consequence of that, Line 2 had an average downtime per machine of 350 hours whereas machines of Line 1 were only down for 150 hours using the same simulation scenario. This means, that Line 2 requires more maintenance workforce than Line 1. But both use cases were conducted using the same input parameter of 2 available maintenance workers. This resulted in an understaffed scenario for Line 2. The consequence of an understaffed maintenance team are more simultaneous breakdowns and a higher average downtime. On the other hand less resources reinforce the performance of a prioritization policy and this is why the average throughput increments in the scenarios of Line 2 are higher than for Line 1. The impact of available resources on the performance of a prioritization policy will be explained more detailed in the next chapter.

5 Discussion

This thesis aimed for developing a data-driven prioritization method for corrective maintenance tasks. In the following chapter, the methods which were developed and analysed, will be compared concerning their usability and performance. Furthermore the limitations for prioritization and different fields of application will be discussed.

5.1 Comparison of Prioritization Policies

The following prioritization policies were analysed within this thesis:

- Heuristic Approaches
 - Part-Out-Part-Out Time
 - Availability
 - Availability + Redundancy
- Bottleneck-based Approaches
 - Blockage & Starvation Probability
 - Active Period Method

The use of heuristic methods for the prioritization of maintenance tasks is the most common approach in industries. Especially the prioritization of single machines over redundant machines is a very straightforward and comprehensibly policy. The simulation study has shown, that this policy performs also well compared to a FIFO policy. But in the case of no redundant machines in a production line, no prioritization would be possible. Furthermore, under the assumption of existing bottlenecks in a production line, there is a potential for improvements by prioritizing bottlenecks.

The bottleneck-based approaches try to use this potential. Both analysed methods use real-time production data for bottleneck detection. Concerning the usability, the Blockage & Starvation Method is harder to implement in industries, since the line structure has to be implemented in the bottleneck detection algorithm. The Active Period Method does not require any information about the line structure, but only the machine states which are usually available in a PDA system. This makes the Active Period Method suitable for Line Production, Job Shop Production and even Flexible Production Systems (Klenner *et al.*, 2016, pp. 543).

Also concerning the validity the Active Period Method was the better detection method. Using a System Sensitivity Analysis it was shown, that the Active Period Method could detect the primary, secondary and tertiary bottleneck of a line correctly, whereas the Blockage & Starvation Method could only detect the primary bottleneck. Furthermore, the Blockage & Starvation method provides only values for very few machines and does not provide any information about the importance of non-bottleneck machines. This can be seen when comparing the priority ranking of the Active Period Method with the one from the Blockage & Starvation Method. Especially for large production systems, it is more beneficial to have a consistent priority ranking for all machines as it is provided by the Active Period Method.

For the evaluation of the performance of the prioritization policies, the average throughput increment compared to a FIFO policy was chosen as a performance indicator. Figure 57 shows the best result for each prioritization policy for the industrial use case Line 1:

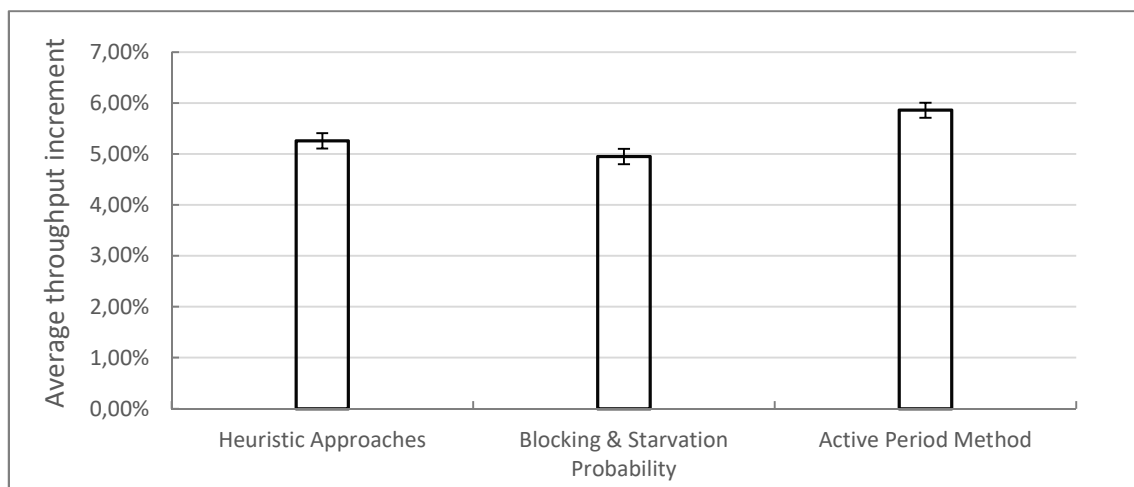


Figure 57: Performance of each prioritization policy for Line 1 compared to a FIFO policy including a 95 % confidence interval

Even the Active Period Method reached the highest average throughput increment of 5,9%, the good result of the heuristic method with 5,2% was not expected at the beginning of the simulation study. But only the heuristic scenario, where the priority of redundant machines was reduced, has performed that good. This explains, that the failure sensitivity value explained in chapter 2.2.2, is a major aspect for task prioritization in corrective maintenance. A prioritization of single machines over redundant machines will always bring a performance improvement independent of the bottleneck situation.

This finding explains, why the Active Period Method performed only 0,7% better than the best heuristic method. The most significant bottlenecks the Active Period Method detected, were all redundant machines. From a production and maintenance management point of view, this is a very positive result. This means, that the most important machines to the system are redundant machines and in case of a breakdown of such a bottleneck machine, the impact on the overall system performance will be less critical than in case of the bottleneck was a single machines. This behaviour was taken into account, by reducing the calculated priority by the number of redundant machines. But for a bottleneck-based prioritization approach this means, that in case of a redundant bottleneck machine, the potential for improvement is very limited.

In general it has to be emphasized, that bottleneck detection alone, cannot be a prioritization policy for corrective maintenance tasks. Further modifications as the reduction of redundant machines and a Priority Increase Factor have to be made in order to improve a system's performance.

Concerning the period of observation for the bottleneck detection, the algorithm performed best using a static average active period priority ranking over 90 days. Even the bottlenecks are shifting over time it could not be shown, that a prioritization of short-term bottlenecks would improve the system's performance. The better performance of a static prioritization was also confirmed by Gopalakrishnan *et al.* (2014, pp. 2173). Even the same behaviour occurred in the second industrial use case, there might be systems where a more dynamic bottleneck detection performs better. This is because both use cases are conducted on highly automated and well-balanced production lines, which have only minor changes in the bottleneck situations due to machine breakdowns. For assembly lines or in

general production systems that are producing multiple products, the cycle times are not always constant over time and therefore such lines are not as well-balanced as the lines of the use cases. In that case, a more dynamic bottleneck detection could be more suitable.

5.2 Limitations for Prioritization in Corrective Maintenance

The biggest limiting factor for the potential of improvement by prioritization in corrective maintenance is the amount of repair workers. If there are infinite workers available, each request can be processed immediately and there is no need for prioritization. Still, companies strive for a cost-efficient use of their resources and therefore especially personnel resources as maintenance workers will always be limited. Therefore in the first industrial use case, several simulation runs with a different amount of workers were conducted. The results of this simulation runs are shown in Figure 58:

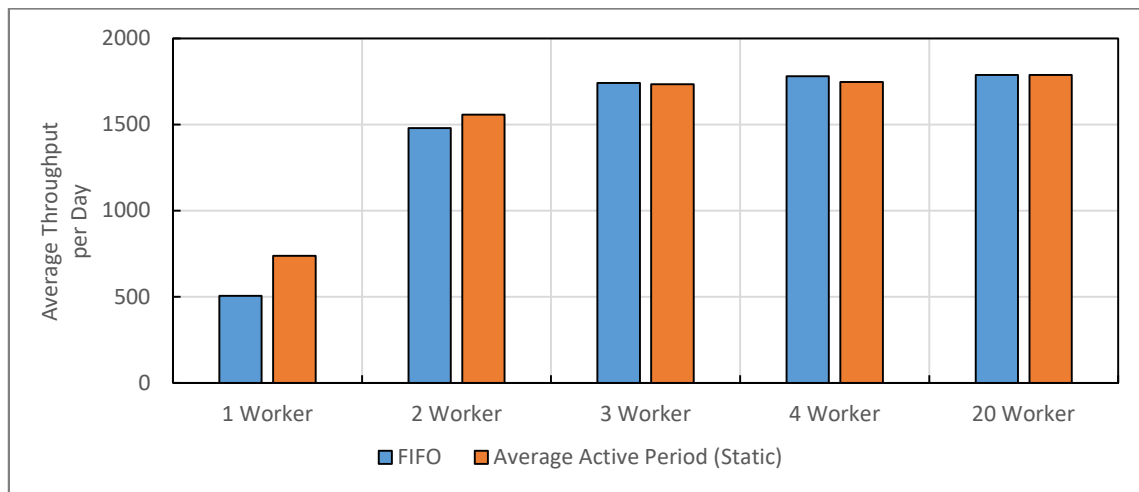


Figure 58: Potential for throughput increment on Line 1 depending on amount of repair workers

The potential of improvement by prioritization decreases with an increasing number of repair workers, since with more workers, most repair request can be processed immediately. In order to maximize throughput, the best result can be achieved having three or more repair workers. The results of Figure 58 can be used for further research on the optimization of total costs of maintenance. Since prioritization can achieve a higher throughput with the same amount of workers, it shifts the cost-

optimum point of total costs of maintenance towards less costs due to breakdowns with equal personnel costs.

5.3 Conclusion

Concerning usability, validity and the potential for throughput improvement, the Average Active Period Method was the best performing prioritization policy. The two industrial cases have shown that there is potential for throughput increment by bottleneck prioritization. Both industrial use cases were very well-balanced and furthermore the bottlenecks of both lines were redundant machines. These were limiting factors for the performance of the prioritization policy. Still the results of this thesis recommend the implementation of the algorithm on production lines for the purpose of corrective maintenance task prioritization. The algorithm was validated successfully and can therefore be a reliable, real-time data driven decision support for the production and maintenance staff and provides useful information about the actual bottleneck situation of a production system.

6 Outlook

The prioritization algorithm for corrective maintenance tasks was realized using a data-driven bottleneck detection method. The bottleneck ranking was modified by introducing a Priority Increase Factor and by reducing the priority of redundant operating sequences. The result of this modification was the final priority ranking. Both of those modifications have still potential for further performance improvements. The Priority Increase Factor increases the priority of a machine depending on the time of how long a machine is already waiting for repair. Within this simulation study, a linear function with the same slope for all machines was used to model this behaviour. Using a machine depending Priority Increase Factor could bring further performance improvements, since the lower breakdown severity of redundant operating sequences compared to non-redundant operating sequences would be considered. In that case, redundant operating sequences would be assigned a lower Priority Increase Factor, whereas single machines would be assigned a higher Priority Increase Factor. The second modification, the reduction of redundant operating sequences priorities, was done by dividing the priority of each operating sequence, by the number of its redundant machines. This was necessary, since otherwise redundant operating sequences, which were detected as bottleneck machines, would have been prioritized over single non-bottleneck operating sequences which broke down at the same time. Even a redundant operating sequence can be a bottleneck of a system, it will always have a lower breakdown severity compared to a non-redundant operating sequence. Therefore, in the case of corrective maintenance, the priority of redundant operating sequence has to be reduced. How much the priority has to be reduced, is not only a matter of the number of redundancies, but also of the actual buffer levels and the structure of a

production system. Therefore, the findings of this thesis lead to further research on how the Priority Increase Factor and the reduction of redundant operating sequences priorities can be modelled in a mathematically more accurate way, in order to improve the proposed prioritization method further.

In the field of preventive maintenance, the scheduling of maintenance tasks is another possible application where a data-driven bottleneck detection could be used. Preventive maintenance tasks, require the stoppage of machines and reduce therefore the availability of those machines. Because of that, effective maintenance task scheduling has a big impact on the OEE of production facilities. Depending on the production system, the shift plan and the personnel resources of a company, not all preventive tasks can be scheduled in non-production shifts. Within the TPM 4.0 research project it was shown, that flexible maintenance windows, which occur due to a production line's dynamics and machine breakdowns, can be used for the purpose of preventive maintenance, without affecting the performance of the production system. Performing preventive maintenance tasks on bottleneck machines only during non-production shifts or flexible maintenance windows (Li *et al.*, 2009) could bring further OEE improvements.

Also besides maintenance there are possible fields of application for the bottleneck detection algorithm. Especially for assembly lines or job shop production layouts which are less automated compared to the two industrial use cases, a short-term bottleneck identification could be used to assign additional resources to the bottleneck stations. Furthermore, the buffer capacities around bottlenecks could be adjusted in order to reduce the balance-losses on a production line. Chang *et al.* (2007) have shown, that adjusting buffer capacities based on information about the bottleneck situation, can bring performance improvements and reduce balance-losses.

References

- Ahuja, I. and Khamba, J.S. (2008), “Total productive maintenance. Literature review and directions”, *International Journal of Quality & Reliability Management*, Vol. 25 No. 7, pp. 709–756, DOI: 10.1108/02656710810890890.
- Balci, O. (1988), “The implementation of four conceptual frameworks for simulation modeling in high-level languages”, in *Proceedings of the 20th conference on Winter simulation, San Diego, California, United States*, ACM, New York, NY, pp. 287–295, DOI: 10.1145/318123.318204.
- Banks, J. (2005), *Discrete-event system simulation*, *Prentice-Hall international series in industrial and systems engineering*, 4. ed., Pearson Prentice Hall, Upper Saddle River, NJ, ISBN: 0131446797.
- Bellgran, M. and Säfsten, K. (2010), *Production development: Design and operation of production systems*, Springer, London, ISBN: 1848824955.
- Biedermann, H. (2008), *Ersatzteilmanagement: Effiziente Ersatzteillogistik für Industrieunternehmen*, 2. Auflage, Springer-Verlag, Berlin, Heidelberg, ISBN: 9783540008507.
- Borshchev, A. and Filippov, A. (2004), “From System Dynamics and Discrete Event to Practical Agent Based Modeling: Reasons, Techniques, Tools”, 22nd International Conference of the System Dynamics Society, July 25 - 29, 2004, Oxford, England, pp. 1–23.
- Chang, Q., Ni, J., Bandyopadhyay, P., Biller, S. and Xiao, G. (2007), “Supervisory Factory Control Based on Real-Time Production Feedback”, *Journal of Manufacturing Science and Engineering*, Vol. 129 No. 3, pp. 653-660, DOI: 10.1115/1.2673666.

- Chiang, S.-Y., Kuo, C.-T. and Meerkov, S.M. (2001), “C-Bottlenecks in serial production lines. Identification and application”, *Mathematical Problems in Engineering*, Vol. 7 No. 6, pp. 543–578, DOI: 10.1155/S1024123X01001776.
- Ebeling, C.E. (2010), *An introduction to reliability and maintainability engineering*, 2. ed., Waveland Press, Long Grove Ill., ISBN: 1577666259.
- EN 13306:2010-10-01 (2010), *Maintenance – Maintenance terminology*, 01.040.03; 03.080.10, Austrian Standards Institute.
- Furian, N. (2017), “HCDESLib”, available at: <https://github.com/nikolausfurian/HCDESLib/wiki> (accessed 30 March 2017).
- Furian, N., O'Sullivan, M., Walker, C. and Voessner, S. (2014), “HCCM - A Control World View For Health Care Discrete Event Simulation”, in *Proceedings / 28th European Conference on Modelling and Simulation ECMS 2014: May 27th - May 30th, 2014, Brescia, Italy*, ECMS, pp. 206–213, DOI: 10.7148/2014-0206.
- Furian, N., O'Sullivan, M., Walker, C., Vossner, S. and Neubacher, D. (2015), “A conceptual modeling framework for discrete event simulation using hierarchical control structures”, *Simulation modelling practice and theory*, Vol. 56, pp. 82–96, DOI: 10.1016/j.simpat.2015.04.004.
- Gopalakrishnan, M., Skoogh, A. and Laroque, C. (2014), “Simulation-based planning of maintenance activities by a shifting priority method”, in *Winter Simulation Conference (WSC), 2014: 7 - 10 Dec., Savannah, GA, Savannah, GA, USA*, IEEE, Piscataway, NJ, pp. 2168–2179, DOI: 10.1109/WSC.2014.7020061.
- Klenner, F., Lenze, D., Schwarzer, S., Deuse, J. and Friedrich, T. (2016), “Smart Data Analytics zur Identifikation dynamischer Engpässe in Flexiblen Fertigungssystemen”, in *Automatisierungstechnik*, Vol. 64 No. 7, DOI: 10.1515/auto-2016-0014.
- Kuo, C.-T., Lim, J.-T. and Meerkov, S.M. (1996), “Bottlenecks in serial production lines. A system-theoretic approach”, *Mathematical Problems in Engineering*, Vol. 2 No. 3, pp. 233–276, DOI: 10.1155/S1024123X96000348.

- Law, A.M. and Kelton, W.D. (2000), *Simulation modeling and analysis*, McGraw-Hill series in industrial engineering and management science, 3. ed., McGraw-Hill, Boston, ISBN: 0070592926.
- Leidinger, B. (2014), *Wertorientierte Instandhaltung: Kosten senken, Verfügbarkeit erhalten*, Springer Gabler, Wiesbaden.
- Li, L. (2009), “Bottleneck detection of complex manufacturing systems using a data-driven method”, *International Journal of Production Research*, Vol. 47 No. 24, pp. 6929–6940, DOI: 10.1080/00207540802427894.
- Li, L., Ambani, S. and Ni, J. (2009), “Plant-level maintenance decision support system for throughput improvement”, *International Journal of Production Research*, Vol. 47 No. 24, pp. 7047–7061, DOI: 10.1080/00207540802375705.
- Li, L., Chang, Q., Ni, J., Xiao, G. and Biller, S. (2007), “Bottleneck Detection of Manufacturing Systems Using Data Driven Method”, in *IEEE International Symposium on Assembly and Manufacturing, 22 - 25 July 2007, Ann Arbor, USA*, IEEE Operations Center, Piscataway, NJ, pp. 76–81, DOI: 10.1109/ISAM.2007.4288452.
- Matyas, K. (2011), *Taschenbuch Instandhaltungslogistik: Qualität und Produktivität steigern*, 4. Aufl., Carl Hanser Fachbuchverlag, ISBN: 3446423761.
- Matyas, K. (2016), *Instandhaltungslogistik: Qualität und Produktivität steigern, Praxisreihe Qualitätswissen*, 6. Auflage, ISBN: 3-446-44614-1.
- Moubray, J. (2000), *Reliability-centered maintenance*, 2. ed., Industrial Press, New York, NY, ISBN: 978-0831131463.
- Nakajima, S. (1988), *Introduction to TPM: Total productive maintenance*, Productivity Press, Portland, ISBN: 0915299232.
- Neubacher, D., Furian, N., Gutsch, C. and Voessner, S., “A Hierarchical Control Simulation Model to Support Maintenance Planning in Flexible Production Systems”, in *European Simulation and Modelling Conference 2016*.
- Pascual, D.G. and Kumar, U. (2016), *Maintenance audits handbook: A performance measurement framework*, CRC Press LLC, Boca Raton, ISBN: 9781466583924.

- Pawellek, G. (2016), *Integrierte Instandhaltung und Ersatzteillogistik: Vorgehensweisen, Methoden, Tools*, 2. Auflage, Springer Vieweg, Berlin, Heidelberg.
- Reichel, J., Müller, G. and Mandelartz, J. (Eds.) (2009), *Betriebliche Instandhaltung, VDI-Buch*, Springer, Berlin, ISBN: 978-3-642-00501-5.
- Roser, C. and Nakano, M. (2015), “A Quantitative Comparison of Bottleneck Detection Methods in Manufacturing Systems with Particular Consideration for Shifting Bottlenecks”, in *Advances in production management systems: Innovative production management towards sustainable growth; September 7-9, 2015*; Vol. 460, Springer, pp. 273–281, DOI: 10.1007/978-3-319-22759-7_32.
- Roser, C., Nakano, M. and Minoru Tanaka, “Detecting Shifting Bottlenecks”, *International Symposium on Scheduling*, Vol. 2002, pp. 59–62.
- Roser, C., Nakano, M. and Tanaka, M. (2001), “A practical bottleneck detection method”, in *Proceedings of the 2001 Winter Simulation Conference: Crystal Gateway Marriott, Arlington, VA, U.S.A., 9 - 12 December 2001*, Assoc. for Computing Machinery, New York, NY, pp. 949–953, DOI: 10.1109/WSC.2001.977398.
- Roser, C., Nakano, M. and Tanaka, M. (2002), “Shifting bottleneck detection”, in Yücesan, E. (Ed.), *Proceedings of the 2002 Winter Simulation Conference: Manchester Grand Hyatt San Diego, U.S.A., December 8 - 11, 2002*, Association for Computing Machinery, New York, pp. 1079–1086, DOI: 10.1109/WSC.2002.1166360.
- Roser, C., Nakano, M. and Tanaka, M. (2003), “Comparison of bottleneck detection methods for AGV systems”, in Chick, S.E. (Ed.), *Proceedings of the 2003 Winter Simulation Conference: Fairmont Hotel, New Orleans, LA, U.S.A., December 7 - 10, 2003*, Association for Computing Machinery, New York, pp. 1192–1198, DOI: 10.1109/WSC.2003.1261549.
- Schenk, M. (2010), *Instandhaltung technischer Systeme: Methoden und Werkzeuge zur Gewährleistung eines sicheren und wirtschaftlichen Anlagenbetriebs*, Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg.
- Schröder, W. (2010), *Ganzheitliches Instandhaltungsmanagement: Aufbau, Ausgestaltung und Bewertung*, Zugl.: Leoben, Montanuniv., Diss., 2009, *Gabler research Techno-ökonomische Forschung und Praxis*, 1. Aufl., Gabler Verlag / GWV Fachverlage GmbH Wiesbaden, Wiesbaden.

Wang, H. (2002), “A survey of maintenance policies of deteriorating systems”,
European Journal of Operational Research, Vol. 139 No. 3, pp. 469–489,
DOI: 10.1016/S0377-2217(01)00197-7.

Yang, Z., Chang, Q., Djurdjanovic, D., Ni, J. and Lee, J. (2007),
“Maintenance Priority Assignment Utilizing On-line Production
Information”, *Journal of Manufacturing Science and Engineering*, Vol. 129
No. 2, p. 435, DOI: 10.1115/1.2336257.