2016

Backup Strategy for Failures in Robotic U-Shaped Assembly Line Systems

Alexander Gebel
University of Rhode Island, alexander_gebel@my.uri.edu

Follow this and additional works at: <https://digitalcommons.uri.edu/theses>

Recommended Citation

Gebel, Alexander, "Backup Strategy for Failures in Robotic U-Shaped Assembly Line Systems" (2016).
Open Access Master's Theses. Paper 904.
<https://digitalcommons.uri.edu/theses/904>

This Thesis is brought to you for free and open access by DigitalCommons@URI. It has been accepted for inclusion in Open Access Master's Theses by an authorized administrator of DigitalCommons@URI. For more information, please contact digitalcommons-group@uri.edu.

BACKUP STRATEGY FOR FAILURES IN ROBOTIC
U-SHAPED ASSEMBLY LINE SYSTEMS

BY
ALEXANDER GEBEL

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
SYSTEMS ENGINEERING

UNIVERSITY OF RHODE ISLAND

2016

MASTER OF SCIENCE THESIS

OF

ALEXANDER GEBEL

APPROVED:

Thesis Committee:

Major Professor Manbir Sodhi

Gretchen A. Macht

Lutz Hamel

Nasser H. Zawia
DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2016

ABSTRACT

The application of robotic U-shaped line layouts is becoming more important for manufacturing companies. Compared to straight assembly line layouts, U-shaped assembly lines result in cost savings, easier material handling and higher production rates. The reason for this is that U-shaped lines improve visibility and skill sharing between operators, increase production quality, reduce work in process inventory and facilitate problem-solving of appearing production failures which is shown in several researches. Key companies such as Toyota and Boeing are using U-shaped assembly lines to benefit from the advantages of U-shaped line layouts. However, few breakdown strategies are designed especially for U-shaped lines even though machine breakdowns are common. Breakdowns reduce the throughput rate and product quality and therefore strategies are needed which can ensure the targeted throughput and product quality of companies during breakdowns. In this thesis a breakdown strategy IS designed for a robotic U-shaped line which uses versatile backup robots on backup stations to cover the failures of workstation robots. Versatile backup robots are only considered in one prior study for a straight line layout and, in that study, the backup robots demonstrated a better performance than other breakdown strategies used for straight lines. The concept of backup stations with versatile robots is adapted to the robotic U-shaped line layout to identify whether backup robots can be an efficient breakdown strategy for robotic U-shaped lines. This adaptation is the placement of the backup stations between the arms of the U-shaped line layout. An automotive body shop assembly line configuration is selected for the U-shaped line layout. Ten workstations are used in the line configuration. Four positions exist for the placement of backup stations. Each

combination of workstations and placement positions have been analyzed to find the most efficient backup strategy for line configuration designed. The analysis starts with the one backup station, then considers two backup stations and finally three backup stations on the four possible placement options. The best option of the one, two and three backup stations are compared with four backup stations and the current breakdown strategies which are the usage of manual repair stations only and the workload reallocation of broken robots by working robots downstream the line. The criteria for the performance comparison are the cycle time and product quality which are generated for a 5%, 10% and 15%-line breakdown. For the generation of the criteria, a genetic algorithm is used which is modified from a straight line layout to the robotic U-shaped line backup strategy and current breakdown strategies. The analyses of the best placement options for the one, two, three and four backup stations options identify that the three and four backup stations options have the best cycle time and product quality for breakdowns, because they cover each workstation without the use of manual repair stations. It is shown that the three backup stations option is the best choice for the designed automotive body shop assembly line configuration. The three backup stations option has the same cycle time and product quality as the four backup stations option, but it uses one less backup station. Furthermore, the robotic U-shaped line backup strategy using three backup stations has a much better performance than the current breakdown strategies. Its cycle time for breakdowns is half as much as the cycle time of the current breakdown strategies and the robotic U-shaped line backup strategy does not use manual repair stations that generate a high product quality consciously. Due to these facts, the robotic U-shaped line backup strategy is an efficient breakdown strategy for

the robotic U-shaped line, because it ensures production with a smooth line flow, a continuously high product quality and the avoidance of work in process inventories for breakdowns. Nevertheless, the robotic U-shaped line backup strategy has three major disadvantages. The first disadvantage is that the backup robots have to be maintained after each operating period to ensure that they do not break down. The next disadvantage is the requirements of an intelligent conveyor system so that the backup station can be accessed without disrupting the material flow when a breakdown occurs. The last disadvantage is that the backup robots have to be equipped with several possibly costly tools, to cover the workstation robots. The final decision on which backup strategy to use is therefore conditional on the cost of equipment, but this study can easily be extended to include these factors when the data is available.

ACKNOWLEDGMENTS

First of all, I would like to thank my advisor Dr. Manbir Sodhi for his valuable support, comments, and suggestions. His advises motivated me to face this challenging research field, acquire new skills and increase my knowledge.

I would like to thank Arash Nasrolahi Shirazi for the assistance to learn the programming language Python, the offering of your genetic algorithm, and the hours of discussions to improve my research project.

Thank you to my committee members Dr. Grechten A. Macht, Dr. Lutz Hamel and Dr. David Freeman for reviewing my thesis and sharing your knowledge.

Additionally, I would like to thank everybody involved in the exchange program between The University of Rhode Island and Technical University of Braunschweig. Your involvement gave me the chance to be a student at URI, make new friends all over the world, and have a wonderful time in the United States of America.

Finally, I would like to thank my parents Viktor and Tamara Gebel as well as my siblings Andrej and Sergej Gebel. Your support made this awesome year in the USA possible. Thank you so much and it is an honor to be a part of such an extraordinary family.

TABLE OF CONTENTS

ABSTRACT	ii
ACKNOWLEDGMENTS	v
TABLE OF CONTENTS.....	vi
LIST OF TABLES	viii
LIST OF GRAPHS	ix
LIST OF FIGURES	x
CHAPTER 1	1
1 INTRODUCTION	1
1.1 Current Situation and Problem Statement	1
1.2 Objectives and Approach.....	4
CHAPTER 2	7
2 Line Balancing Problem.....	7
2.1 Industrial environments	8
2.2 Product design and process layout selection.....	9
2.3 Constraints and attributes leading to line balancing constraints	12
2.4 Objective function	16
2.5 Solution procedures	19
2.5.1 Exact methods.....	19
2.5.2 Heuristic procedures	25
CHAPTER 3	33
3 U-Shaped line layout.....	33
3.1 Lean Manufacturing.....	33
3.2 Functionality of a U-Shaped line	38
3.3 Breakdown strategies.....	45

CHAPTER 4	52
4 Robotic U-shaped line backup strategy design	52
4.1 Breakdown strategies evaluation	52
4.2 Backup strategy design and functionality	55
4.3 Genetic algorithm for the backup strategy	63
CHAPTER 5	89
5 Robotic U-shaped line backup strategy analysis	89
5.1 One backup station analysis	90
5.2 Two backup stations analysis	96
5.3 Three backup stations analysis	101
5.4 Breakdown strategies comparison	105
5.5 Breakdown strategies comparison using a five-time greater rework time	112
5.6 Breakdown strategies comparison using a two-time greater rework time	117
CHAPTER 6	123
6 Critical view on the rU-sbs & further research	123
CHAPTER 7	130
7 Summary	130
APPENDICES	133
Appendix A: Kahan et al. approach [50]	133
Appendix B: Müllers redundancy level approach [9]	136
Appendix C: Modified Genetic Algorithm	140
BIBLIOGRAPHY	141

LIST OF TABLES

Table 2.1: Goldberg's Population Fitness	28
Table 2.2: Goldberg's example offspring [41]	31
Table 4.1: Task execution time matrix.....	64
Table 4.2: Design of a capability matrix [50]	65
Table 4.3: Task precedence matrix	66
Table 4.4: Genetic algorithm replication test 1	72
Table 4.5: Genetic algorithm replication test 2.....	73

LIST OF GRAPHS

Graph 3.1: Comparison of Shirazi et al. [53]	50
Graph 5.1: Mean cycle time for the use of one backup station.....	90
Graph 5.2: Product quality analysis for the one backup station options	93
Graph 5.3: Mean cycle time for the use of two backup stations	96
Graph 5.4: Product quality analysis for the two backup stations options	99
Graph 5.5: Mean cycle time for the use of three backup stations	101
Graph 5.6: Product quality analysis for the three backup station options.....	103
Graph 5.7: Mean cycle time of the breakdown strategies considered.....	106
Graph 5.8: Product quality analysis for the breakdown strategies comparison	109
Graph 5.9: Mean cycle time of the breakdown strategies using a five-time greater manual rework time.....	113
Graph 5.10: Product quality analysis for the breakdown strategies comparison using a five-time greater manual rework time.....	116
Graph 5.11: Mean cycle time of the breakdown strategies using a two-time greater manual rework time.....	118
Graph 5.12: Product quality analysis for the breakdown strategies comparison using a two-time greater manual rework time	121

LIST OF FIGURES

Figure 1.1: Chapters of the Master thesis	6
Figure 2.1: Precedence graphs [5].....	9
Figure 2.2: Facility design factors [54]	10
Figure 2.3: Process Layouts [5]	11
Figure 2.4: Classification of ALB problems [15]	16
Figure 2.5: Branch and Bound illustration.....	23
Figure 2.6: Genetic algorithm concept [40]	28
Figure 2.7: Crossover example [40].....	30
Figure 2.8: Mutation of a binary string [42]	30
Figure 3.1: A simple U-Shaped line [6].....	38
Figure 3.2: Chase mode in U-Shaped lines [43]	39
Figure 3.3: Double-Dependent U-lines [43]	41
Figure 3.4: Multi lines in a single U-shapes layout [43].....	42
Figure 3.5: Embedded U-lines [43].....	42
Figure 3.6: Figure-eight-pattern U-line [43]	43
Figure 3.7: Multi-U-line facility [43].....	43
Figure 3.8: Breakdown strategy with inventories [49]	45
Figure 3.9: Balancing of uncompleted tasks [50]	46
Figure 3.10: Rebalancing of a whole assembly line [52].....	48
Figure 3.11: Schematic of Backup stations for high capability robots [53]	49
Figure 4.1: Self-generated robotic U-shaped line design.....	56
Figure 4.2: Work position of robots from Station 1 [9]	57

Figure 4.3: Four options for the robotic backup station placement	58
Figure 4.4: Backup station option A covers broken robots of workstation 1 and 2....	59
Figure 4.5: Backup station option A covers broken robots of workstation 9 and 10..	59
Figure 4.6: Use of manual repair stations in the robotic U-shaped line design	62
Figure 4.7: Framework of the modified algorithm	69
Figure 4.8: Randomly generated parent gene	75
Figure 4.9: Example of ten piece allocated to a specific workstation.....	78
Figure 4.10: Example of 12 pieces for the one backup station option A.....	79
Figure 4.11: Feasible number of tasks on each workstation	81
Figure 4.12: Allocation of each tasks to one workstation robot	82
Figure 4.13: Work time for each robot	83
Figure 5.1: Final robotic U-shaped line backup strategy design.....	112

CHAPTER 1

INTRODUCTION

1.1 Current Situation and Problem Statement

The methods of manufacturing have changed significantly over the centuries and these changes are described as Industrial Revolutions. In the 18th century, the first Industrial Revolution began with the use of powered machine tools [1]. The first Industrial Revolution resulted in a fundamental change from agricultural to industrial societies. Henry Ford pioneered the second Industrial Revolution by inventing mass production and assembly lines [1]. The third Industrial Revolution started in the nineteen-seventies with the development of automated manufacturing systems and programmable machines. In addition, manufacturing principles such as Lean Manufacturing have made the current production system more efficient by eliminating waste and continuous production process improvements [2, 1]. The National Science Foundation gives an American example of the consequences of ignoring manufacturing trends in their study [3]. In the nineteen-eighties the American market was overflowing with products coming from more efficient Japanese factories using the principles of Lean Manufacturing for an improved production process with less production waste, while the focus of American factories was to produce as many products as possible in the mass production flow lines. The elimination of waste in the production process leads

to a higher product quality and lower prices compared to the mass production flow lines. Thus the consumer preferred to buy the products from the Japanese factories [3]. The examples from the National Science Foundation about the American markets should demonstrate that manufacturing companies have to keep their eyes open for changing trends and production environments [3].

A study made by MHP – A Porsche Company presents that the fourth Industrial Revolution arises, which involves the use of Smart Factories. Smart Factories are companies connected intelligently with their production environment which includes the connection of human, machines and resources with each other. The continuous growth of the internet and information technologies provides factories and their resources with more information that leads to transparency of information. Figure 1 illustrates an example from the MHP – A Porsche Company study of Smart Factories connected over Computer Processing Systems with their environment which consist of Smart Logistic, Smart Buildings, Smart Products, Smart Grids and Smart Mobility [1].

Figure 1.1: Smart Factory [2]

Increasing individual customer needs, volatile global markets, scarcity of resources, ecological requirements and cost pressure are the current challenges for factories. The fourth Industrial Revolution will help to handle these challenges by providing factories more information about their environment that will help factories to react more flexibly to changes. According to the study of MHP – A Porsche Company, the ability to react to demand variability which includes time and value aspects, using resources efficiently, customer oriented product design and production will be important features resulting from flexibility [1]. The study of MHP – A Porsche Company is only a survey of the slowly increasing awareness of German companies about the upcoming trend of the fourth Industrial Revolution. Nevertheless, challenges such as individual customer needs, volatile global markets and cost pressure exist already and solutions have to be developed to handle these challenges [4]. Production optimization is one important step to increasing companies' efficiency [4]. Production flow lines exist in many manufacturing companies and they require high investment and running cost. These costs have a significant influence on economic performance of the company and therefore the line balancing problem is important for the production optimization [5]. Robotic assembly lines are highly automated systems to produce finished goods. Although much research has been done in the broad field of the line balancing problems, only a few papers consider robot breakdowns despite the fact that breakdowns are common. Another common topic in assembly line balancing problems considers U-shaped layouts. Compared with the straight assembly line layout, U-shaped assembly lines result in reduced cost, easier material handling and higher production rates. The reason for this is that U-shaped lines improve visibility skills between operators,

increase production quality, reduce work in process inventory and facilitate problem-solving of appearing production failures which is shown in several researches [6, 7]. Companies such as Toyota and Boeing are starting to use U-shaped assembly lines to become more efficient [8]. Therefore, the objective of this thesis is to design a breakdown strategy for a robotic U-shaped assembly line which ensures an efficient line flow for breakdowns.

1.2 Objectives and Approach

The objective of this thesis is to design a breakdown strategy for robotic U-shaped flow lines which ensures a production with a smooth line flow and a continuously high product quality. The approach for the breakdown strategy starts with a general description of the flow line balancing problem. Chapter 2 explains in detail the different constraints, optimization goals, and solution approaches which are essential to model and solve a flow line balancing problem.

Subsequently, a brief description of U-shaped lines will follow in Chapter 3. The advantages of U-shaped assembly lines compared to straight assembly lines will be discussed. Furthermore, the requirements of U-shaped assembly lines will be shown and which breakdown strategies already exist for U-shaped assembly lines and for straight assembly lines. The reason for the consideration of breakdown strategies for the straight assembly lines is that the research field of breakdown strategies is limited. The consideration of a wider breakdown research field will ensure that an efficient breakdown strategy can be designed for the robotic U-shaped assembly line.

Chapter 4 evaluates the existing breakdown strategies and it designs a U-shaped line configuration which is an adaptation of an automotive assembly line body shop. For this line configuration is a breakdown strategy design that uses the various number of one, two, three and four robotic backup station on the four possible placement options in the robotic U-shaped line layout configured. Furthermore, the functionality of the various design options of the robotic U-shaped line backup strategy are described.

Chapter 5 analyzes the various number of one, two, three and four backup station on the four possible placement options of the robotic U-shaped assembly line backup to identify the best performing option. In addition, the robotic U-shaped assembly line backup strategy is compared with current breakdown strategies based on its performance. The performance of the various design options for the robotic U-shaped line backup strategy and current breakdown strategies are investigated for 5%, 10% and 15%-line breakdown scenarios.

Afterwards, a critical view on the generated robotic U-shaped backup strategy follows in Chapter 6. This critique shows some drawbacks which have to be considered to make the robotic U-shaped backup strategy a useable implementation for factories. The critique leads to further research requirements in the assembly flow line breakdown strategy field. At the end of this thesis is a summary of the chapters and the generated results. Figure 1.2 illustrates the chapters of this thesis.

Figure 1.1: Chapters of the Master thesis

CHAPTER 2

LINE BALANCING PROBLEM

The Line Balancing Problem has been widely researched. It is important for researchers as well as practitioners, because flow lines require high investment cost and can lead to high running cost [9]. Furthermore, the line balancing problem consider various restrictions and constraints, which break the line balancing down and specify it. Especially the constraints could specify the line balancing problem to current challenges of factories. An example for line balancing problem dealing with current challenges of factories is in a journal article published by Chicaet et al. [10]. It deals with the optimization of an assembly line balancing problem considering the constraints varying work time, space and uncertain demand, which are equivalences to the aspects individual customer needs and volatile global markets as current challenges of factories [10]. This chapter will review the literature relevant to the restrictions and constraints. Subsequently, the general line balancing will be zoomed in to the robotic U-shaped assembly line balancing problem.

2.1 Industrial environments

The industrial environment specifies the general term line balancing problem by giving it a functionality. Common industrial environments in the academic research are machining, assembly and disassembly lines [5]. In machining lines, operations on parts such as drilling, welding, grinding and etc. are completed on several machines. Machining and assembly lines are highly automated and have to follow given precedence relations. Assembly lines produce final products and the significance is that several operations can be done simultaneously on a station with more than one machine or robots. Assembly configurations are also being investigated by disassembly line types. The research on disassembly lines is growing because of the rising governmental regulation for product recycling and therefore parts have to be recycled or reused as good as possible. Nevertheless, the most disassembly lines are manual and just reversing of the precedence relations of the assembly graph gives not a working disassembly graph [5].

Figure 2.1 illustrates from the literature, an example for precedence graphs in the typical industrial environments. The circles with numbers represent a task and the arrows are the relationships between tasks. As an example, in Figure 2.1, the assembly line task 4 can just start after tasks 2 and 3 have been completed, but task 5 has just to wait for task 3 to be proceeded. Companies decide their industrial environments by considering the products they are producing which leads to other two important aspects of the line balancing problem the product design and process selection.

Figure 2.1: Precedence graphs [5]

2.2 Product design and process layout selection

A product design translates into a set of tasks which have to be executed to produce a specific product. Therefore, tasks are the breakdown of the full production process into logical and small steps. Following these steps leads to the required product in the defined Quality. The steps generate the precedence relationship between each other. In assembly lines a final product just could be assembled after subassemblies and components of subassemblies has been done. The technology used is also an important consideration for the precedence relationships [9]. In new facilities, the set of tasks defines the production technology that has to be purchased and the product design creates the process sequences through the whole facility. Figure 2.2 shows that the 3 factors product, process and schedule design are defining the facility design.

Figure 2.2: Facility design factors [56]

Thereby the line balancing problem generates the schedule design using the defined constraints and objectives. A more detailed explanation about constraints and objectives will be given in section 2.3 and 2.4. However, when using existing production facilities, the designed product has to be completed with the existing line technology. In addition, existing production could have resulted in machines placed in a specific line layout. The line layout is crucial for the line direction and possible distribution of tasks to a special workstation. Typical line layouts are basic straight lines, straight lines with multiple workplaces, U-shaped lines and lines with a circular transfer which are shown in Figure 2.3

Figure 2.3: Process Layouts [5]

In basic straight lines, a workpiece runs through each workstation in the given order. Thereby the required set of tasks is done one after the other and the workpiece comes from the last station as a finished good, if the industrial environment is an assembly line [7].

A single straight assembly line works for simple products. Complex products and high production intensities require a straight line layout with multiple workplaces or a U-shaped line layout for a smooth production flow. In a straight line with multiple workplaces layout several tasks can be performed simultaneously at each station. This

is essential for a smooth line flow where a specific amount of subassembly has to be done before the workpiece can enter the next station [11].

Lines with circular transfers place their workstation around a rotating table as illustrated in figure 2.3. The table is used for loading and unloading the workstations with the required material to produce the finished good. A line layout with a circular transfer can be seen as being equivalent to a line balancing problem for a basic straight line and straight line with multiple workplaces. The frequency of the turn tables decides which optimization method could be used. A single turn is equivalent to basic straight lines and multi-turn for straight line with multiple workplaces [12].

As Figure 2.3 shows, U-shaped lines have their start and end point at the same place and operator could work inside the line layout. The literature mentioned several advantages of U-shaped line layouts compared to straight line layouts [8, 13], which will be detailed further in Chapter 3.

2.3 Constraints and attributes leading to line balancing constraints

Constraints construct a border for the line balancing problem in which the optimal task to workstation scheduling has to be found. Thereby constraints arise from logical, mathematical, practical conditions and from attributes of the objectives considered in the optimization [5].

A logical constraint mentioned in Section 2.1 is the precedence relationship between tasks, which have to be fulfilled to produce the required product. Another

logical constraint is the number of workstations. The line balancing optimization cannot schedule task to an eleventh workstation, if just ten workstations are given [14].

The cycle time is one of the most important constraints in the line balancing problem and it belongs to the mathematical constraints. In the literature two different definitions for the cycle time are given. The first definition describes the cycle time as the time needed to produce a finished product from the start to end of a production line in a facility. The second definition describes the cycle time as the amount of time given to each workstation to fulfill their scheduled tasks [15]. The second cycle time definition is more commonly used and the following formula shows how the upper bound of the cycle time could be calculated.

$$Cycle\ Time = \frac{Effective\ time\ available\ per\ shift}{Production\ volume/shift} \quad [15]$$

The line balancing problem should consider several attribute which influence optimization [5]. Each workstation has attributes which influence the distribution of tasks to a specific workstation. These attributes could be the type and number of workers and tools assigned to a specific workstation and the buffer capacity of each workstation [5]. In the literature these types of optimization problems are described as Assembly Line Design Problems (ALDP), because they try to set up workstations optimally for the assembly tasks [16].

Worker distribution can also be considered in the line balancing problem and they too can be defined by specific attributes. A current optimization model of Ramezani et al. considers the different skill levels of workers and the amount of cost they cause with the scheduling to specific workstations [17].

Another important attribute mentioned in the context of the line balancing problem is that of the task attributes which could be constant, dynamic, uncertain or dependent on the assignment to a workstation. Dynamic and uncertain attributes or lead times make the line balancing problem very complex and increase the computing time required to find a feasible solution compared to that for constant and assignment dependent problems. On the other hand, dynamic and uncertain attributes reflect the practical manufacturing and even an attribute considered constant such as the task time, could become uncertain [18]. However, the optimization with constant attributes is needed to find optimal solution approximations and establish a foundation for further researches. For example, the Simple Assembly Line Balancing problem which considers constant task times, an upper bound of a given cycle time for every station and respects the precedence constraint between the tasks was introduced in 1955 and was used to minimize the number of the workstations used in a basic straight line design. Since this initial problem formulation, the related body of research has grown continuously and in just the period from 2007 to 2012, 267 scientific papers were published for this line balancing problem [5].

In practice, companies produce not just one product, but several models of a basic product and/or several different products. The literature defines the optimization problems which consider the number of products a Single-model lines, Mixed-model

lines and Multi-model lines. Single-model just considers one basic product, Mixed-model lines consider a similar products and the Multi-model line consider different products, which are usually produced in batches [5]. It is obvious that the complexity of such problems increases significantly with the number of the products and differences between them.

It is not just the number of products, that are considered in the line balancing problem. The number of flow lines is also a constraint because factories usually have more than one production line. The line balancing problem therefore includes cases with multiple lines (with identical or different configuration workstations) and workers assigned to more than one line and several parallel lines with crossover [19, 20]. Multiple lines are very complex to configure, because they must also consider constraints previously listed such as task and workstation attributes. Therefore, finding the solution becomes a time intensive and complex task [5]. However, considering every constraint mentioned above makes the line balancing problem too complex. Thus the literature has started to categorize the line balancing problem.

Figure 2.4 illustrates a classification of the Assembly line balancing problem. The single model assembly line with deterministic task times in a U-shaped line layout is the simplest mentioned research field for U-shaped lines. Nevertheless, it simplifies the functionality of a production line and offers a foundation for the complex backup strategy research. Therefore, the single model assembly line with deterministic task times in a U-shaped line layout will be considered in this research.

Figure 2.4: Classification of ALB problems [15]

2.4 Objective function

Another difference between line balancing problems is in the objective function. The first formulations of the assembly line balancing problem sought the minimum number of workstations to manufacture a product [21]. The types of the objective functions considered have since increased. Besides the minimization of the workstation, other objective functions are:

- Minimization of the cycle time.
- Maximization of the line efficiency.
- Maximization of the system utilization.
- Minimize the re- and configuration cost.
- Maximize the line profit.

The list of objective functions shows that constraints can become objective functions, because the cycle can be used as objective and constraint. The researcher defines his/her optimization goal and chooses the best fitting objective function for his purpose. Each objective function requires its own constraint variations and therefore the models in the literature vary considerably [5, 22]. Current research tries not just to optimize the production lines, but also to make them robust. In Xu et al. definition “robust approaches try to find a solution or a set of solutions that performs well across all scenarios and hedges against the worst of all possible scenarios” [23]. Taguchi introduced a methodology for robust optimization and defined three stages to attain robust design. The first stage is the systems design where the parameters of a product are defined in general. In the second stage these parameters are optimized to create quality requirements. These two steps are the usual steps of optimization problems which were mentioned above. The creation of a tolerance for the design parameters is the last step of this methodology [24]. Thereby tolerances are uncertainties and they could be deterministic, probabilistic and possibilistic. Deterministic tolerance gives an area in which a parameter for a task and/or workstation can vary. The second tolerance type works with probabilities in which an event changes the parameters to a specific value. Possibilistic tolerances are fuzzy measures in which probabilities could appear to change parameters into a plausible range [25].

Robust optimization increases the complexity of the line balancing problem dramatically. It considers the range of listed constraints in Section 2.3 and can consider additional parameters, while changing their values. Thus several options exist to solve robust optimization. Beyer et al. presented theoretical and practical solution options in

a survey. The theoretical methods for robust optimization such as the robust counterpart approach and the aggregation approach are not considered in the flow line balancing literature, because their complexity needs an enormous amount of computing power and time to generate a possible solution [25].

Practical methods to solve robust optimizations include evolutionary approaches such as genetic algorithms. Evolutionary approaches belong to approximate methods which do not give an optimal solution for a problem. Rather they generate a feasible solution for a problem in an acceptable computing time [5, 25]. A more detailed description about solution method will be given in the following section. Robust line balancing approaches are developed to handle uncertain data. Robust means also that the flow line should continue to operate even if one or more machines break down. Break downs are practical problems and should also be considered in robust design. Battaia et al. and Hazir et al. recommend the line balancing problem with robustness against break downs as further research [5, 26]. The literature of the assembly line balancing offers studies about break downs, because of their practical application. Therefore, in Section 3.3 the current state of the break down research will be given and especially which break down strategies could be adapted for robotic U-shaped assembly lines.

2.5 Solution procedures

After considering all the parameters above, the last step in the line balancing is to choose a solution procedure to solve the problem. The solution procedures have to find the best solution for the defined constraints and objective function. In addition, the solution procedure has to execute fast. In the line balancing problem, the performance of solution procedures is measured with the required time to find an optimal solution [27]. Another important factor in the performance is the solution value. Some solution procedures provide better solutions than other procedures and therefore the literature classifies the solution procedures in exact and heuristic methods.

2.5.1 Exact methods

As their name suggests, exact methods find the best solution for an optimization by considering a specific number of tasks and constraints. Therefore, the objective function and the constraints have to be defined in a mathematical model. The most common model for the assembly line balancing problem is the mixed integer program. To illustrate how a mixed integer program works, the simple assembly line balancing problem (SALBP) 1 and 2 will be taken as example. These two optimization problems are very simply defined. As mentioned in section 2.3 the SALBP-1 optimizes the number of used machines by considering the constraints cycle time and precedence relationship between tasks. The SALBP-2 is similar to the SALBP-1. It uses also a limited amount of constraints, but it optimizes the cycle time for a given number of

machines. The following parameters were assumed to generate a mathematical model for the multi integer program [28, 29]:

- N number of tasks
- K number of workstations
- t_i time to fulfill a task
- C_t cycle time
- P_i set of immediate predecessors of the task i
- W_k weight (cost) of an assigned workstation k
- E_i earliest possible workstation for task i
- L_i latest possible workstation for task i

SALBP – 1 [28]

$$\text{Minimize } \sum_{i=1}^N \sum_{k=1}^N w_{ik} * x_{ik} \quad (1.1)$$

$$\sum_{k=1}^N x_{ik} = 1 \quad \forall i = 1, \dots, N \quad (1.2)$$

$$\sum_{i=1}^N t_i * x_{ik} \leq C_t \quad \forall k = 1, \dots, N \quad (1.3)$$

$$x_{i_2 k_1} \leq \sum_{k=1}^{k_1} x_{i_1 k} \quad \forall i_2, k_1 = 1, \dots, N \text{ \& } i_2 \in P_{i_1} \quad (1.4)$$

$$w_{ik} = w_k \quad \forall k \quad (1.5)$$

$$N * w_k \leq w_{k+1} \quad (1.6)$$

$$x_{ik} \in \{0, 1\} \quad (1.7)$$

SALBP – 2 [29]

$$\text{Minimize } C_t \tag{2.1}$$

$$\sum_{k=E_i}^{L_i} x_{ik} = 1 \quad \forall i \tag{2.2}$$

$$\sum_{\forall i | k \in [E_i, L_i]} t_i * x_{ik} \leq C_t \quad \forall k \tag{2.3}$$

$$\sum_{k=E_i}^{L_i} k * x_{ik} \leq \sum_{k=E_j}^{L_j} k * x_{jk} \quad \forall (i, j) \in P \tag{2.4}$$

$$x_{ik} \in \{0, 1\}$$

The equation (1.1) is the mathematical formulation for the objective function of this optimization problem, which searches for the minimum number of used machines. The constraint (1.2) defines in a mathematical form, that each task could be just assigned to one machine. In the equation (1.3) is defined that the used work time of each machine has to be lower or equal to the cycle time to fulfill all the assigned tasks. In the next step the equations (1.4) defines the precedence constraint. A task i can be just done on a machine if its predecessor i_2 has been done on a previous or on the same machine. The SALBP - 1 has two equations more than the SALBP – 2. The equations (1.5) and (1.6) support the objective function by weighting the machines. Each time the number of machine used increases, the new machine gets a higher weighting than the previous machine. This weighting idea should support the objective function to keep the used number of machines as small as possible. In the end is the equation (1.7) which defines the values for the variable x_{ik} . Thereby x_{ik} has the value 1 if task i is assigned to the

machine k and otherwise it gets the value 0. The linear definition of constraints in a mathematical model and that x_{ik} could just take the values 0 and 1, makes this model to a mixed integer program solution procedure. The SALBP – 2 has some similar equations as the SALBP – 1. The equation (2.2) defines as well that each task could be just assigned to one machine and equation (2.3) sets the cycle time as upper bound work time for each time for each machine. Thereby it has to be mentioned, that the SALBP – 2 does not have a given cycle time. The interaction between the equations (2.1) and (2.3) is searching for the lowest cycle time by considering as well the other constraints. The equation (2.1) is the objective function of the SALBP-2, which is the cycle time minimization. This example should show, that an objective function can be a part of the constraints. The precedence relationship will be defined by equation (2.4), where a tasks j can just be done after its predecessor i . In the end the equation (2.7) makes the SALBP – 2 to a mixed integer program model such as equation (1.7) does it with the SALBP – 1. As mentioned in section 2.3 the line balancing problem can become very complex with the number of constraints used. Even if the set of task and machines is small, some constraints can make it impossible to construct a linearized mathematical model. In this case a nonlinear integer program could be used. A good example of this could be found by Hamta et al., who extended the SALBP-2 with flexible tasks times and a second objective function which consider the machine cost. These additional parameters made it impossible to create a linearized mathematical model and therefore a non-linear model was created to find an exact solution [30]. The linear and nonlinear integer programming model is used to create mathematical equations, which have to be solved to find the exact solution. It is practical to use special solver as Cplex, LINGO, ILOG to generate

a solution. These solvers follow the branch and bound algorithm to generate the exact solution [5]. The branch and bound algorithm can be seen as a tree diagram. It creates several levels of branches. At each level the algorithm compares the value of the branches and let just the branch grow, which has the best value. Here best means a low value if it is a minimization problem and a high value if is a maximization problem. The branch and bound algorithm creates as long level of branches as the entire of set parameters are considered in the tree diagram [31]. It is difficult to illustrate a line balancing problem in a branch and bound algorithm, because it becomes very huge even with a small set of parameters. Therefore, Figure 2.5 shows the basic idea of the branch and bound algorithm. In this example 4 task has to be ordered in an optimal position and from the start point the task 1 and 4 have the same value and a better value than task 2 and 3. Thus the branch and bound algorithm follow these branches to find the optimal solution.

Figure 2.5: Branch and Bound illustration

The possibility to find an exact solution is just one aspect of evaluating the performance of solution procedures. Another criterion is the required amount of time to find the exact solution. Thus the researchers try to modify their mathematical model with task specific bounds. A current example for a modified model is the branch, bound and remember algorithm from Sewell et al. which is branch and bound approach combined with dynamic programming [32]. The branch and bound part eliminates sub problems, which cannot offer a better solution than the current found branch solution. The dynamic program remembers all calculated solution and avoid that a solution option is calculated twice solve. Thus branch, bound and remember algorithm can solve the simple line balancing problem faster than any other exact algorithm [32].

Dynamic programming is a fast method to generate an exact solution. It divides a problem in sub-problem and generates solutions for the sub-problem. Afterwards the best solution is generated out of the sub-solutions by changing the sub-problems until the best solution is found for the initial problem [33].

The literature describes line balancing as NP-hard. Thus the required solution time increases exponentially with the parameters such as task size and the number of workstation used. With more parameters and uncertain data, exact solution may not be found. Therefore, approximate procedures have been developed to solve optimization problems with a large amount of sets, uncertain data and several objective function [10, 34].

2.5.2 Heuristic procedures

Heuristic procedures may not find the optimal solution for a line balancing problem, but they find acceptably good solutions even for complex problems in an acceptable amount of time [5]. The literature categorizes approximate procedures into simple heuristic and metaheuristic methods.

Simple heuristic methods use greedy algorithms or priority rules to generate a feasible solution for a large problem size in an acceptable time amount. Most priority rules are used for tasks or workstation attributes, which increase the complexity for the solution finding as mentioned in section 2.3 [35]. In addition, the user of simple heuristics methods can decide what an acceptable solution search time is. Therefore, they have to define how many iterations their method does until it stops and delivers the feasible solution. Needless to say that a low number of iterations do not usually generate near optimal solutions. Nevertheless, simple heuristic methods are often used to generate an upper bound for exact solution methods and these are used to find an optimal solution for a large problem size [35].

Metaheuristic methods are used for optimization problems with large problem sizes and complex constraints. Mostly a mathematical model cannot be created to solve such problems and metaheuristic methods are able to generate a near optimal solution. Metaheuristic methods are build up in a programming language as C, C+, Pascal or Python and follow specific algorithms.

Some of the heuristic approaches used in the literature for the solving of the line balancing are:

- Neighborhood methods [36]
- Evolutionary approaches [37]
- Swarm intelligence approaches [38]

The neighborhood methods are used for the optimization of multi-objective problems. The optimization starts by finding the best solution for the first objective. Afterwards, the second objective is considered and the neighborhood methods searches near the area (neighborhood) of the best solution for the first objective to find the best trade off solution for both objectives [36].

The swarm intelligence approaches base on the natural behavior of animal swarms in the food search process. In the optimization problems, the objective represents the food and a several number of search function, which is defined by the user of the swarm intelligence approach, represent the individuals in a swarm. The search functions start the solution search process simultaneously over the whole search area. After the finding of a good solution that has to be defined by the user in the initial phase of the swarm intelligence approach, all research functions concentrate on the area of this good solution generated, to find a better solution for the optimization problem [37].

The evolutionary approaches are based on natural behavior as well. The complexity of these methods makes it difficult to illustrate them. Thus the genetic algorithm will be used to demonstrate the evolutionary approaches. The genetic algorithm is a part of the evolutionary approaches and the most widely used metaheuristic method [5]. John

Holland introduced the genetic algorithm for the first time in 1975. The Genetic algorithm is an abstraction of the biological evolution process adapted in a computer system [39]. Figure 2.6 illustrates how the biological evolution process adapts in an algorithm. Thereby all stages will be explained with the biological logic in them and how this logic gets translated into an algorithm. The first step of a genetic algorithm is to initialize a population. These will be the first parents of several generations that follow. In the line balancing problem this population exist of a chosen amount of possible solution for an optimization problem. All the parameters of the considered constraints are genes and their connection build up a string called chromosome [40]. In programming languages all kind of alphabets can be used to design chromosomes. A binary alphabet will be used to show how the genetic algorithm work based on example of Goldberg [41]. Nevertheless, numerical and characters can also be used as alphabets. The process to choose an alphabet and design the chromosomes is called coding in the computer language. The second stage in the genetic algorithm is to evaluate the fitness of the population. Fitness is the value which gets generated by chosen chromosomes. A high value is good or bad depends on the objective function. If the objective is to minimize the cycle time, a lower cycle time is better than a higher. Goldberg choose in his example the function $f(x) = x^2$ in an interval from 0 to 31 and he wanted to maximize the value of $f(x)$. Therefore, he chooses randomly 4 numbers as population, programs them as chromosomes in a binary algorithm and evaluate their fitness as shown in Table 2.2 [41].

Figure 2.6: Genetic algorithm concept [40]

Parent No.	Real No.	Binary No.	Fitness $f(x)$
1	13	01101	169
2	24	11000	576
3	8	01000	64
4	19	10011	361

Table 2.1: Goldberg's Population Fitness

Constraints Satisfaction is the third stage of the genetic algorithm. In this stage it has to be proven whether all parents fulfill the defined constraints. In Goldberg's example the only constraint is that the number found should be in the interval between 0 and 31. All chosen parents in Goldberg's example are in this interval and there they all fulfill the given constraints [41]. However, the best solution for $f(x) = x^2$ in the interval between 0 and 31 has not been found yet. Therefore, the genetic algorithm has an additional constraint. This constraint is the number of iterations, which the algorithm has to do. Iterations means how many generations of possible solution the algorithm produce until it can stop. In the given example the algorithm is in it 0 iteration, because it is first generation of solution. The following stages will show how generations of solutions are created [39].

The fourth genetic algorithm stage is select survivors. The programmers decide how many parents of the population will be used in the next stage. This shows that genetic algorithms are individual designed to solve optimization problem and a general genetic algorithm does not exist [10, 39]. Nevertheless, it is logical to use the parents with the best fitness. Goldberg use 4 parents in his example. Therefore, he chose the best 3 parents to randomly vary individuals [41].

Randomly varying individuals is the next stage of the genetic algorithm. This is the reproduction stage in where the children of the survival population are made. Two methods exist to produce the next generation. These methods are crossover and mutation [40].

Crossover means that two parents generate two offerings. Each offspring has the genes of the two parents. The programmer of the genetic algorithm decides how many

genes of a parent go to an offspring. Thereby, the parent chromosome can be cut down in a chosen number of genes and distributed to the two offspring. Figure 2.7 shows a crossover with one cutting point.

Figure 2.7: Crossover example [40]

Mutations modify one or more genes in the created offspring. As in nature the mutation probability should be low in the genetic algorithm [40]. In the end the programmer has to decide which mutation probability is used or if the genetic algorithm should use only crossover to search for the best solution. Eiben et al. recommend to use mutation to find better solutions [42]. The crossover search consists only of solutions, which are combined of the two parents. Using mutation brings new information in the solution area and can identify much better solution, because the first parent generation is generated randomly. In Figure 2.8 is the mutation of a binary string shown.

Figure 2.8: Mutation of a binary string [42]

Goldberg's example use just crossover to create the first offspring generation, because the probability for mutation in the first iteration is very low. In addition, he uses the best three parents as survivors. As mentioned above two parents create two offspring. Therefore, Goldberg uses the best solution twice to create four offspring [41]. Table 2.3 illustrates Goldberg's first generation of offspring.

Parent			Offspring		
String	Value	Fitness	String	Value	Fitness
0110 1	13	169	01100	12	144
1100 0	24	576	11001	25	625
11 000	24	64	11011	27	729
10 011	19	361	10000	16	256

Table 2.2: Goldberg's example offspring [41]

The stage randomly varies individuals ends with the creation of the offspring and leads the algorithm to evaluate fitness and afterwards to the constraint satisfaction stage again, which work the same as illustrated above. In this stages the offspring created become the new parents and the stages will repeat until the defined amount of iterations has been completed. If this happens, the algorithm will go to the last stage to output results. In the last stage the current offspring is taken and the offspring with the best fitness is presented as the best solution for the problem. As mentioned above approximate solution procedures may not deliver the best possible solution but a near

optimal solution. In Goldberg's example a near optimal solution could be found even after first iteration (see Table 2.3).

The detailed explanation of the genetic algorithm should underline the key factors, which make the genetic algorithm to most used method for complex optimization methods:

- Solution search start from a population and not just from a single point
- Parent population can be generated randomly
- Using probabilities for creating offspring (mutation and cut points)
- User creates coding part individually to design the chromosome and to validate their fitness [41]

These criteria make the genetic algorithm to flexible optimization method which can be used for a large amount of optimization problems, because the user created coding part can be adapted to numerous optimization problems. The random research starting points offers the chance to find good solution in several solution areas, which allows to solve complex problems. Furthermore, the more iteration the genetic algorithm does, the merrier the response will be as shown in table 2.3. Thus the user can get a good solution even for a self-defined number of iterations.

CHAPTER 3

U-SHAPED LINE LAYOUT

Japanese factories started the use of U-shaped production layout to build up a just-in-time (JIT) production. Miltenburg underlies in his survey of U-shaped production lines, that some writers see the U-shaped line design as the most effective technique for a just-in-time production [43], which will be shown in this chapter. JIT belongs to the Lean Management principles. Therefore, the following chapter will give a short overview of Lean Management. Afterwards the idea and the advantages of U-shaped production lines will be presented. In the end of Chapter 3 an overview of breakdown strategies for assembly line will be given and shown which breakdown strategies are especially used for U-shaped production lines.

3.1 Lean Manufacturing

Lean Manufacturing based on the Toyota Production System, which development started in 1959 by Dr. Shigeo [44]. It is a continuous improvement in the production process to satisfy the customer requirements in terms of cost, quality and delivery times by reducing lead time, cost, improving the process flow and on the elimination of waste generated in the production environment and all activities that do not add value to the enterprise [45, 46]. Toyota proofed, that the principles of Lean Manufacturing are

successful and today Toyota is a benchmark for other manufacturing companies. The identification and elimination of seven types of wastes is one of the basic principles of Lean Manufacturing.

The first type of waste is the waste from producing defects. The later a defective product is detected, the more this defect will cost. A defect product identified by a customer has to be repaired or replaced and can lead to the loss of customer. But even if defects are detected as soon as possible, they lead to cost in detecting and repairing them. In the worst case the unfinished good has to be thrown away and this leads to additional cost, which the customers are not willing to pay. Therefore, the production process should be done right and every step of the productions should be defined in the product design phase correctly [47].

Another type of waste is the waste in transportation. Material has to move through different stations until it becomes a finished good. Thus the layout of the facilities and the routing sequence of operations should be optimized to deliver the minimum transportation cost as possible [47].

The third waste is the waste from inventory. "Toyota calls inventory the roof of all evil" [47]. Every item, which sits in the inventory, causes cost and binds money that could be invested in other opportunities. Moreover, inventory hides the company problems as inadequate market intelligence, instability and worse quality of the production process. To perform a better productions process, inventory should be eliminated [47].

Waste from overproduction belongs also to the seven wastes of lean manufacturing. The production output of companies is much higher than the customers demand for a

product, which leads to inventories. Companies do this to keep their workers and machines busy and get low unit prices. But unsold units produce also cost, which has to be carried by the sold units. In the end each over production unit just leads to more cost. Furthermore, if everyone has to be busy, no one gets the chance see the emerging problems of the company [47].

The next waste in Lean Manufacturing is the waste of waiting time. This waste includes waiting for orders, parts, materials, items from preceding processes, or for equipment repairs [47]. It is a sign for a flawed process flow, if waiting appears. Moreover, waiting time increases the unit cost, for which the customer has to pay, and in addition the customer has to wait longer [48].

Another waste is the waste in processing. Every task, which doesn't add value to a product, should be eliminated. Additionally, each process should be improved, if the improvement makes the process more efficient [47].

The last waste of lean manufacturing is the waste of motion. This waste takes a deeper look on every step of a process and tries to eliminate each unnecessary movement to make the process much more efficient [48].

Eliminating the seven wastes of Lean Manufacturing results in an efficient production process with a high quality. For achieving this a Total Quality Management System is needed, which is also an important part of Lean Manufacturing.

The term Quality is defined it from the interaction of the customers and producer's perspective. Customers buy products to fulfill their needs, which have to be translated by the producers to the basic quality of their products. This translation occurs in form of the product design and manufacturing. By including the other departments as

engineering, manufacturing, marketing, sales and the suppliers the Total Quality Management gets defined. Thus everybody shares an idea of how their performance should be. The workers start to proof the work in process and give a feedback to their predecessor, because a failure performance of the predecessor cannot be recovered by the current station. In addition, everybody has to be watchful and flexible, because the needs of the customers are changing continuously and therefore the companies have to change and improve their understanding of quality continuously.

The elimination of the seven wastes and Total Quality Management stipulates, that only goods should be produced, which fulfill the quality and the demand of the customers. Hence it is logical, that Lean Manufacturing includes a process of controlling production and this starts with the customer [47]. Pull Production is the term for this method and it was developed in the 1950s by Toyota. American supermarkets first implemented these methods and they were adopted to the manufacturing industry. Each time a customer buys a product, the predecessor station is allowed to send an order to their predecessor station [48].

This procedure should ascertain, that only needed goods are produced and it deviates from the old form of Push Production. Push Production is the opposite production method, which try to produce as much goods as possible. The idea behind Push Production is to get low unit cost. Discounts should secure, that customers are buying this huge amount of products [48].

It is difficult to state in general terms whether pull or push production is better, because it depends on the products. If a product is standardized such as toothpicks, push production is better to get low unit cost and a huge volume of products. But if a products

get more specific, then the pull production is better suited to fulfill the needs of the customers [47].

To realize the concept of pull production, the two aspect of Lean Manufacturing needed are Just in Time and SMED.

Just in Time is a concept with the main idea being that a station gets its required material in the needed moment and in the right amount. This concept supports the idea of the elimination of inventory. To fulfill the requirements of Just in Time, the delivery of the material has to be optimized and the delivered material has to fulfill the standards of Total Quality Management [48].

SMED means Single Minute Exchange of Die. This concept is more a methodology developed by Shingo [47], which has the goal to reduce the time, where a machine or an operation has to be stopped to change a tool, to a single minute. In the practice a minute to exchange a tool could not be realized for a long time [47]. Today, automotive assembly line robots use more than one welding gun for their tasks execution and could change the welding guns used in a few seconds [9].

Lean manufacturing follows the concept of a continuous improvement, also called Kaizen, which makes it to a dynamic production strategy. Continuous improvement means that mistakes are analyzed in-depth to find and solve the reason for the mistake, learn from it and never repeat the mistake again. The concept of Kaizen also extends to the workers. They should get the chance to improve and raise their knowledge and skills through different projects and tasks to become an important part of the company.

3.2 Functionality of a U-Shaped line

In a U-Shaped production line are the producing machines arranged in a U form. Thereby the start and the end of the production line are at the same vicinity. Operators work inside the U-Shaped line design as illustrated in the following Figure [6]. It is possible to place the operators outside the line, if the machines allow an operation from both sides. Nevertheless, the literature places the operators in the middle of lines and this concept will be used in the following.

Figure 3.1: A simple U-Shaped line [6]

The movement of the Operator and the production flow can be clockwise or counterclockwise [6]. The flow direction can be decided by the line balancing decision and which balancing direction delivers the best line efficiency. In Miltenberg's description of the U-shaped line, no further material is allowed to enter the line while a product is still in work [43]. Thus the idea of Just-in-Time should be fulfilled to produce

just a product then it is needed. Additionally, it become easier to stop the machines if a problem with a product or machine appears. This production flexibility should be able to generate the required quality with zero product defects. Miltenberg defined, in his survey of U-shaped production lines, the chase mode. Originally the chase mode means that one operator works at a U-shaped line and convoys the products through all the workstations. More common is that two or three operators run a U-shaped line. In this scenario the operators are assigned to a specific section of the line and fulfill their scheduled tasks for each product. Figure 3.2 shows the chase mode with one operator and two operators.

Figure 3.2: Chase mode in U-Shaped lines [43]

One operator is able to run a whole production line, because there is a separation of work by the operator and the machine work. The operators work usually consists of:

1. bring the work in process part to the required machine
2. load the machine with the part and other requirements
3. start the machining process
4. wait a short to check if everything is alright
5. unload the work in process part
6. check the quality of the part [43]

The machine work is the automated part work with the machining functions as drilling, welding, assembly or other machining process which are needed to produce a product in the required quality [43]. One big advantage of U-shaped lines over straight lines is the better rebalancing possibilities. Rebalancing includes the following three functions:

- varying the production rate
- moving machines
- changing standard operations [43]

Varying the production rate means including operates to increase the production rate or removing operators to decrease it. Flexible and multi-skilled workers are needed to adapt the current production rate to the required demand. This leads to better educated workers and make the production job more interesting [43].

Moving machines and changing standard operation are necessary for new products and technological production innovation. As mentioned in section 2.2 the product design and production technology are the decisive part for the production tasks. With innovations, new tasks appear which requires additional precedence relationship and other machines. Therefore, moving machines and changing standard operations is essential for a smooth production and flexible production rates [43].

The current description of U-shaped production lines refers on the simple U-shaped line design. In Section 2.3 it was mentioned how complex the line balancing problem can become with multi-lines. In practice it is common to have more than one simple line. The following figures illustrate how other configurations of more complex U-shaped lines are discussed in the literature.

Figure 3.3: Double-Dependent U-lines [43]

Figure 3.4: Multi lines in a single U-shapes layout [43]

Figure 3.5: Embedded U-lines [43]

Figure 3.6: Figure-eight-pattern U-line [43]

Figure 3.7: Multi-U-line facility [43]

All the U-shaped lines above are designed for a practical use with multi-product production. It is not common and too laborious to reorganize the whole production line for product changes. In addition, each layout can vary their production rate by adding workers to the lines. Figure 3.7 is the most complex layout design, because it structures the whole facility in a U-shaped layout [43]. These complex U-shaped line designs illustrate the advantages of U-shaped lines.

The first advantage of the U-shaped layout is the increased visibility and communication in the production process. Thus the production quality increases and problems are solved much faster, because workers recognize problems faster and can help each other to solve it [6].

Another advantage is that workers become much more skilled. Workers are scheduled between workstations and different lines to vary the production rate. This work rotation makes the work more interesting and the workers learn many more tasks, which also helps them to react more efficiently to emerging problems [6].

The next advantage is the possibility of the line rebalancing. The flexible reaction to demand helps companies to fulfill the requirements of lean manufacturing to avoid inventory, overproduction and to increase the production quality [6].

The last advantage is that U-shaped lines requires fewer workstations than straight lines, because they offer more possibilities to schedule tasks. This leads to less investments for U-shaped lines and a higher production quality can be reached with less invested money [6].

3.3 Breakdown strategies

Battaia et al. and Hazir et al. refer in their surveys about the line balancing problems to considered machine breakdowns in further research [5, 22]. The research on breakdown strategies is limited and can be categorized in the following options:

- Inventories
- Balancing of uncompleted tasks
- Rebalancing of the whole line
- Backup robots

All of the breakdown strategies mentioned are mainly used for straight assembly lines. Only the inventory breakdown strategy was used for the U-Shaped line layout. Miltenburg compared the effectiveness of U-shaped lines with straight lines for machine breakdowns, which will be explained in the following [49].

Figure 3.8: Breakdown strategy with inventories [49]

Figure 3.8 shows Miltenburg's experiment design. It is simple designed with three workstations, seven tasks and a manageable amount of precedence relationships. Despite all their advantages are U-shaped lines are more efficient than straight lines, if inventories of work in process parts are placed after every workstation. If the placement of inventories after a workstation is not possible, a straight line layout should be used for proper high volume production even during breakdowns [49].

Another breakdown strategy is the balancing of uncompleted tasks. If a machine breakdown appears, the scheduled tasks of the broken machine cannot be performed. Kahan et al. introduced a mixed integer program formulation for the rebalancing of tasks from broken machines [50]. Appendix A includes the mixed integer formulation of Kahan et al. They used the design of an automotive body shop, where a car body runs through workstations and welding robots add parts until it is completed. Each station has several robots, which can perform several task simultaneous [50]. The following figure illustrates the experiment design of Kahan et al.

The problem illustration: robot R4 in station A fails and is replaced by robot R4 in station B.

Figure 3.9: Balancing of uncompleted tasks [50]

In addition to the mixed integer formulation, Kahan et al. tested what the best distribution for the uncompleted task is. In their test the objective function is to minimize the cycle with the reallocation of the uncompleted tasks to working stations. The response of this test is very clear. Manual repair stations had the worst performance with the highest cycle time. On manual repair stations, workers do to all the required work manually to complete the tasks of the broken machine. The best performance has the breakdown strategy with the reallocation of uncompleted tasks to working stations downstream the flow line [50].

The reallocation of tasks from a broken to a working workstation is possible if the working workstation has the same equipment or capabilities as the broken one. Additionally, the precedence relationships between tasks have to be respected. The redundancy level maximization of assembly line with a specific number of tools at workstations is a possible way to resolve breakdowns. Furthermore, tool redundancy simplifies task reallocation. Müller formulated a mixed integer program with the objective function of tool redundancy to handle robotic breakdowns in assembly lines [9]. Müllers optimization model consists of two steps. The first step finds the minimum cycle time for the user researched assembly line balancing problem. Afterwards, the found cycle time is taken as an upper bound work time for all workstations and a second optimization tries to find the best redundancy level, which should ensure the line performance for breakdowns. Müllers mixed integer optimization model can be seen in appendix B [9].

Shin et al. created a decision tool for uncompleted task reallocation. This decision tool has been designed as a practical approach based on variables as repair time and inventory size. The objective of the tool is to decide if the uncompleted part should be allocated to a working station or a manual station to secure the required throughput [51].

The third breakdown strategy is the Rebalancing of the whole line without the consideration of the broken workstation. While Kahan et al. just wanted to reallocate the task of the broken station to deliver a fast possibility of task reallocation, other authors have balanced all task to the working stations. A current research of Sanci presents a branch and bound algorithm, which balance all tasks in a feasible way after a breakdown [52]. The objective function is to minimize the cycle time to secure a smooth production flow even with fewer working machines. Figure 3.10 shows the basic idea of Sanci's branch and bound algorithm.

Figure 3.10: Rebalancing of a whole assembly line [52]

Sanci's branch and bound algorithm is the Rebalancing of a whole flow line, but just with fewer workstations. The goal is to make balancing algorithm faster and to increase their reaction time on breakdowns. In addition, the Rebalancing of all tasks requires a tool redundancy at all stations, which is similar to the breakdown strategy balancing of uncompleted tasks.

The last breakdown strategy is the use of backup robots, which was introduced by Shirazi et al. and realized in a multi-objective genetic algorithm [53]. This algorithm starts with a regular line optimization by scheduling tasks to the given number of workstation under the objective of a cycle time minimization. The special feature in this algorithm is the availability of additional backup stations for workstation with a high capability [53]. Stations with a high capability have a higher probability to break down than stations with an average and low capability. Therefore, just high capability stations need a backup station to avoid breakdowns. Shirazi et al. [53] tested his algorithm in the scenario of an automotive body shop assembly line, which is illustrated in the following Figure 3.11.

Figure 3.11: Schematic of Backup stations for high capability robots [53]

The backup stations support the high capability stations in a normal situation without a breakdown by taking some tasks away from the high capability stations. In Figure 3.11 workstation robots R1 and R4 have a high capability and hence they get supported by the backup stations 1 robots. Then a robot breaks down, the backup station just performs the tasks of the broken robot to keep a smooth production flow [53]. The objective of the multi-objective genetic algorithm of Shirazi et al. was to create a solution method, which does not require a high tool redundancy. A high tool redundancy leads to high investment cost and an inefficient production layout. Therefore, Shirazi et al. compared their solution method with other tool redundant solution methods [53]. The following graph illustrates the comparison between the multi-objective genetic algorithm of Shirazi et al. and the tool redundancy methods.

Graph 3.1: Comparison of Shirazi et al. [53]

Graph 3.1 uses the criteria cycle time and group of tasks performed on manual repair station. The lower the cycle time is the better is the performance, because a low cycle time ensure a high production throughput. Furthermore, robots offer a higher product quality than manual repair stations and therefore the fewer tasks are executed on manual repair stations, the higher is the product quality [53]. Graph 3.1 shows that the breakdown strategy with backup stations have nearly the same performance as a tool redundancy breakdown strategy with a six level redundancy. A Redundancy level is the average number of robots, which can perform one task [9]. The backup station solution of Shirazi et al. has also a 2.5 level of redundancy [53]. A six level redundancy requires much more investment than just a 2.5 level of redundancy, because the cost for tools can become very high. Nevertheless, both solutions need manual stations for breakdowns. In the method of Shirazi et al. the user has to decide what a high capability for workstations is and where a backup station has to be placed. If the capability on some workstations is not high, the user will not place a backup station there. In this case are manual stations needed for the low capability stations to secure a continuous production flow for a robot breakdown [53].

CHAPTER 4

ROBOTIC U-SHAPED LINE BACKUP STRATEGY DESIGN

The previous Chapters gave a detailed explanation of the line balancing problem and U-shaped line layout. The goal of this Chapter is to design a backup strategy for the robotic U-shaped line, which is efficient and fulfills the requirements of lean manufacturing. Therefore, the breakdown strategies mentioned in Section 3.3 will be validated for their adaption ability to robotic U-shaped line. Afterwards, the functionality of the backup strategy designed will be explained. At the end of this chapter will be an explanation of the methodology to investigate the performance of the robotic U-shaped line backup strategy designed.

4.1 Breakdown strategies evaluation

Section 3.3 presented the breakdown strategies for the line balancing problem. The U-shaped line layout was designed as an effective technique for a just-in-time production as a part of the lean manufacturing principles. Only Miltenburg's inventory breakdown strategy is especially designed for U-shaped production lines, because the research field for breakdown strategies is very limited and the most breakdown strategies are designed for straight lines. At this point appears a discrepancy between the inventory breakdown strategy and the lean manufacturing principles of waste

elimination in form of inventories. This discrepancy has to be considered in the generation of an efficient breakdown strategy which supports the basic idea of U-shaped lines to improve the production with lean manufacturing.

The breakdown strategies balancing of uncompleted tasks and rebalancing of the whole line are mainly used for straight production lines. These breakdown strategies fulfill the principles of lean manufacturing much better than the inventory breakdown strategy, because they generate a smooth line flow without additional inventories. Thus it could be an opportunity to transfer one of these breakdown strategies for the U-shaped line layout. However, it has to be accepted that the balancing of uncompleted tasks and rebalancing of the whole line breakdown strategies need a tool redundancy on the used workstations. This fact is as well a discrepancy with the lean manufacturing principles mentioned in Section 3.1. Tool redundancy is a waste in processing, because during normal production conditions without breakdowns, the workstations are equipped with much more tools than needed. This leads to higher investment cost and binds money in the production lines which could be used more efficient in other departments. Furthermore, the working workstations take the workload of the broken ones. Thus the workload increases of the working workstations. A higher workload claims the workstations for larger durations and the probability of a breakdown increases. Consequently, these breakdown strategies make the production line more unstable during breakdowns. Due to this fact, the balancing of uncompleted tasks and rebalancing of the whole line breakdown strategies belong not to the best options for a U-shaped line backup strategy.

The breakdown strategy of Shirazi et al. is designed for straight lines and uses versatile backup robots. The use of backup robots has two major advantages. The first advantage is the backup robots support of workstations with an above average workload for a normal production without breakdowns. Thus breakdowns can be avoided with the reduction of high workloads. Furthermore, the cycle time can be reduced which increases the production throughput. During breakdowns the backup robots only focus on the tasks of the broken robots. This leads to a small increase of the cycle time and a high throughput can be still reached. Additionally, other workstations will not be influenced by a breakdown [53]. These advantages make the breakdown strategy of Shirazi et al. the best of the options considered and thus it offers the potential for an adaption to the U-shaped line layout. Nevertheless, the breakdown strategy of Shirazi et al. has some disadvantages. The user has to decide what a high capability for the used workstations is and how many robotic backup workstations the algorithm should use. Additionally, manual repair stations are needed for workstation which do not get a robotic backup station. Manual repair stations have two great disadvantages compared with robotic backup stations:

- a work time which is up to three times greater as reported in Kahan's et al. research [50]
- a lower product quality.

These disadvantages affect the line performance dramatically, because the cycle time gets much higher and the product quality decreases. Due to these facts, the U-shaped line requires a breakdown strategy which can avoid manual repair stations, has an efficient number of backup stations and can handle breakdowns with an adequate cycle time increase. A robotic U-shaped line has the ability to provide these factors for an efficient breakdown strategy. In Section 3.2, it was noted that the place between the top and the bottom part of the U-shaped line is used by operators. A robotic U-shaped line works automatically and does not need operator to perform on the workstations [8]. Thus, robotic U-shaped lines have unused space between their arms. The following Section will demonstrate how the unused space could be used to generate an efficient breakdown strategy.

4.2 Backup strategy design and functionality

An average U-shaped line consist of ten workstations [43]. This thesis will use the average number of ten workstations to design a practical breakdown strategy. The first step for the breakdown strategy design is to illustrate the placement and the design of the chosen number of ten workstations in a robotic U-shaped line. Figure 4.1 shows the self-generated robotic U-shaped line layout which is used for a breakdown strategy design.

Figure 4.1: Self-generated robotic U-shaped line design

The self-generated robotic U-shaped line design is an adaption of an automotive body shop assembly line, which is also used by Shirazi et al. to test the performance of their breakdown strategy [53]. An automotive body shop assembly line has four robots on each workstation. Each robot has a number of tasks which have to be performed on a specific position of the car body. As example, workstation 1 will be taken from the self-generated robotic U-shaped line design to demonstrate the performance of each workstation robot. The robot with the number 1 performs the tasks on the left and rear position of the car body. Robot 2 performs the tasks on the right and rear position and robot 3 perform the tasks on the front and left position. The last robot with the number 4 perform the tasks on the front and right position. The following figure summarizes the description of workstation 1.

Figure 4.2: Work position of robots from Station 1 [9]

Figure 4.2 shows that each robot on a workstation concentrates on a specific side and position of the car body. Thus robots can break down on a workstation and the other robots on the same workstation are still able to execute their tasks. This leads to the assumption that the precedence relationship between tasks constraint exist not between robots on the same workstation. The precedence relationship between tasks constraint has to be only considered within a robot and between workstations. This assumption offers the opportunity to design a breakdown strategy with an efficient number of backup stations. The reason for this is that the robotic U-shaped line offers four options for a backup station placement and the usage of all options allows the coverage of the whole robotic U-shaped line. Figure 4.3 illustrates the four options for the robotic backup stations placement.

Figure 4.3: Four options for the robotic backup station placement

Each backup station consists of two backup robots. The placement of backup stations between the arms of the U-shaped line allows the coverage of four workstations by one backup station:

- Option A covers workstation 1,2,9 and 10
- Option B covers workstation 2,3,8 and 9
- Option C covers workstation 3,4,7 and 8
- Option D covers workstation 4,5,6 and 7

The backup station option A will be used as example to demonstrate how one backup station is able to cover four workstations. Figure 4.4 and 4.5 illustrates the coverage of four broken workstation robots by backup station option A.

Figure 4.4: Backup station option A covers broken robots of workstation 1 and 2

Figure 4.5: Backup station option A covers broken robots of workstation 9 and 10

In the Figure 4. and 4.5 the robots 1 and 4 on the workstation 1, robot 3 on workstation 9 and robot 2 on workstation 10 are broken. All these workstation robots are covered by the backup robots on backup station option A. Thereby, the car body enters the backup station twice. The first entry is after workstation 1 where backup robot 1 is covering workstation robot 1 and backup robot 2 is covering workstation robot 2. The next entry of backup station option A is after workstation 9. In the process backup robot 1 covers robot 3 of workstation 9 and backup robot 2 covers robot 2 of workstation 10. The coverage of two different workstations with only one backup station entry is possible, because of the precedence relationship between tasks constraint mentioned above. This constraint assumes that the tasks precedence relationship consists within a robot and between workstations. Thus backup station option A can cover the broken robot from workstation 9 and additionally it can execute the tasks from the broken robot of the following workstation 10. In this scenario backup station 1 has an above average workload, because each backup robot covers two workstation robots. Therefore, the cycle time of this backup station is the sum of the first and second entrance.

The usage of all robotic backup station options allows the coverage of the whole U-shaped line. Thus manual stations are not needed, which leads to a constant high product quality. Furthermore, operators do not have to wait on manual repair stations until a breakdown occurs. Rather operators can concentrate on the line performance and react much faster on breakdowns to repair the broken robots. The operator can enter the inner of the U-shaped line configuration while the workstations execute tasks. The conveyor systems of automotive body shop uses overhead conveyors which means that the flow line can be crossed while the workstations execute tasks [54]. Consequently,

the robotic U-shaped line backup strategy designed with four robotic backup stations fulfill the requirements mentioned above of an efficient backup strategy which avoid manual repair stations, use an efficient number of backup stations and handle breakdowns with an adequate cycle time increase. Nevertheless, the question appears whether four robotic backup stations are really necessary to design a breakdown strategy with a good performance. Industrial robots are expensive and the usage of more backup robots than needed would be a waste in processing which is against the principles of lean manufacturing. Hence, this thesis will test all the possible options of backup stations placements to find the best backup strategy for the robotic U-shaped line. The possible options of the backup stations placements are:

- One backup station on the places A, B, C or D
- Two backup stations on the places AB, AC, AD, BC, BD or CD
- Three backup stations on the places ABC, ABD, ACD or BCD
- Four backup stations on the places ABCD

The usage of only one backup station do not allow to cover each workstation. Due to this fact the backup station options, which are not able to cover each workstation, get a manual repair station. Figure 4.6 uses the one backup station option A to demonstrate the use of manual repair stations.

Figure 4.6: Use of manual repair stations in the robotic U-shaped line design

In this scenario robot 1 on workstation 3 and robot 2 on workstation 4 are broken. Furthermore, the backup station option A is used which can cover the workstation 1,2,9 and 10. The broken robots are out of the reach of the used backup station. Due to this fact the broken robots have to be covered by manual repair stations. The manual repair stations are placed at the end of each workstation which is an adaption of the manual repair station use of Kahan et al. [50]. For a realistic comparison between the backup station options, the manual repair station usage has been extended. In Kahan et al. design the manual repair stations cover only the broken robots of the previous workstation. The robotic U-shaped line design has the assumption that the precedence relationship between tasks constraint has to be only considered within a robot and between workstations. Thus the manual repair station 3 in Figure 4.6 is able to cover the failure robots of workstation 3 and 4. Furthermore, manual repair station 4 is also place in Figure 4.6 to cover the failure robots of workstation 4. The placement of a manual repair station after each workstation should support an acceptable line performance, although

their performance is much worse compared to robotic backup stations. The disadvantages of manual repair stations compared with robotic backup stations is mentioned in the previous section. Nevertheless, manual repair stations are needed for the options which cannot cover each workstation. The comparison between the different options in Chapter 5 will show whether the use of manual repair station will decrease the line performance dramatically. Before the comparison between the different options of the robotic U-shaped line backup strategy can be done, a line balancing problem has to be generated and a solution approach has to be chosen. This will be done in the next Section.

4.3 Genetic algorithm for the backup strategy

The robotic U-shaped line backup strategy designed has 40 workstation robots and therefore at least the number of 40 tasks is needed, because it makes no sense to let workstation robots unutilized. For the performance research of the backup strategy options designed, this thesis will use 72 tasks. Such there is the opportunity to utilize some workstation robots with two or three tasks which can be seen as above average workload compared to robots with only one task. Each task needs a time to be executed. This execution time will vary in the range from 20 to 41 seconds based on the task elimination time of Kahan et al. which was taken for the performance research of their breakdown strategy [50]. The allocation of the execution time to each task is noted in a task time matrix. Table 4.1 illustrates a section of the used task time matrix.

Task	Execution Time in seconds
1	33
2	24
3	22
4	22
5	28
6	33
7	24
8	22
9	22
10	28

Table 4.1: Task execution time matrix

In the illustrated section of the task time execution matrix, task 1 requires an execution of 33 seconds, task 2 requires an execution of 24 seconds and so on. Afterwards, it has to be defined which task can be executed by which robots. This is implemented in a capability matrix. A capability matrix is a simple matrix with zeros and ones. A one indicates that a robot can perform a specific task and a zero means the robots cannot perform this specific task. Table 4.2 illustrates a capability matrix.

		Robot													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
Group of spots	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	3	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	4	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	5	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	6	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	7	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	8	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	9	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	10	0	0	0	0	0	0	0	1	0	0	0	0	0	0
	11	0	0	0	0	0	0	0	1	0	0	0	0	0	0
	12	0	0	0	0	0	0	0	0	1	0	0	0	0	0
	13	0	0	0	0	0	0	0	0	0	1	0	0	0	0
	14	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	15	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	16	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	17	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	18	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	19	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 4.2: Design of a capability matrix [50]

In this example tasks 8 and 9 can be only performed by robot 7 and tasks 15, 16 and 17 can be only performed by robot 12. The next step is the design of the precedence relationship between task matrix. An example of such a matrix is illustrated in the following table.

		Tasks									
Tasks		1	2	3	4	5	6	7	8	9	10
	1	0	0	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0	0	0	0
	5	1	0	0	0	0	0	0	0	0	0
	6	1	0	0	0	1	0	0	0	0	0
	7	0	1	0	0	0	0	0	0	0	0
	8	0	0	1	0	0	0	0	0	0	0
	9	0	0	1	0	0	0	0	1	0	0
	10	0	0	0	1	0	0	0	0	0	0

Table 4.3: Task precedence matrix

The task precedence matrix defines which task as to be executed before another task can be started. In Table 4.3 the first four tasks do not have a predecessor, because there are just zeros in the matrix and a zero means that no precedence relationships exist between the tasks. On the other hand, task 6 has two predecessors. The task 6 can be performed only, after the tasks 1 and 5 have been executed. If Table 4.3 is referred to the robotic U-shaped line designed, the first four tasks will be executed on the first workstation and each of these tasks will be executed on a different workstation. Which task is executed by which robot is considered in the capability matrix as mentioned above. The tasks 5 to 10 will be executed on the second workstation, where the tasks 5 and 6 will be executed on one robot and the tasks 9 and 10 will be executed on one robot. Thereby, the task 5 has to be executed before task 6 can be performed and task 9 has to be executed before task 10 can be performed.

All the matrices described above define constraints which have to be considered for a feasible solution. An objective function is needed for the performance comparison between the different options of the robotic U-shaped line backup strategy designed. Graph 3.1 in Section 3.3 has shown that backup strategies in the literature use the cycle time as performance comparison. Hence, the cycle will be taken as well as objective function to investigate the performance of the robotic backup strategy designed.

A solution approach is needed to investigate the performance of the robotic U-shaped line backup strategy options. The use of 72 tasks and 40 workstation robots makes the robotic U-shaped line backup strategy to a large size problem compared to backup strategies from the literature which use around 19 to 40 tasks and 14 to 20 robots [9, 50]. As mentioned in Section 2.5, exact solution approaches require a long time to find a solution for large size problems. Furthermore, the different options of the robotic U-shaped line backup strategy require multiple options for the task allocation which makes this breakdown strategy to a complex design. For these reasons the use of a genetic algorithm will be best choice to investigate the performance of robotic U-shaped line backup strategy.

Genetic algorithms can be constructed in a computer programming languages as Java, C, C++, Pascal, Python and so on [41]. Section 2.5.2 describes the basic idea of genetic algorithms in a simple way, but the coding of an efficient genetic algorithm is complex. The generation of an original genetic algorithm will exceed the scope of this thesis. Thus an existing genetic algorithm will be adapted to the robotic U-shaped line backup strategy design. Shirazi generated for his research a fast genetic algorithm which

has given good solutions to many different problems.¹ This algorithm is realized in the programming language Python and it will be used for the adaption to the requirements of the U-shaped line backup strategy designed. In the discussion that follows, the most important steps of the adapted genetic algorithm will be shown for the U-shaped line backup strategy. Each step of the modified algorithm will be explained in detail. Furthermore, the differences between the original genetic algorithm of Shirazi and the modified algorithm will be explained. The explanation of the differences is important, because Shirazi's algorithm is designed for his backup robots breakdown strategy in a straight line. The following Figure illustrates the framework of the modified algorithm. Before the algorithm starts, the user has to decide which of the backup options mentioned above the algorithm has to investigate for the U-shaped line backup strategy.

¹ Shirazi's genetic algorithm research is still in progress. He presents in April the current standing of his algorithm research and proofed the efficiency of his algorithm. The literature will be published end of the summer 2016.

Figure 4.7: Framework of the modified algorithm

The first step of the modified algorithm starts with a verification whether the algorithm has done enough solution replication. In the literature is the number of breakdowns for the line balancing problem not specified [9]. Due to this fact this thesis will consider a 5%, 10% and 15%-line breakdown. For each breakdown scenario 40 solutions will be generated to find an average and significant cycle time. Hence, the algorithm will end after 121 solution replications which is the step 15 in Figure 4.7. The first run is to find the cycle time for the robotic U-shaped line for a line flow without breakdowns. The precedence relationship between tasks, capability and the task time matrices are the same for the one, two, three and four backup stations options for the robotic U-shaped assembly line backup strategy and therefore the cycle time for a line flow without breakdowns is the same for each backup strategy option. Furthermore, the cycle time for a line flow without breakdowns is the lower bound for the performance comparison, because the cycle time for breakdown scenarios cannot be lower than the cycle time for line flow without breakdowns. Thus the cycle time for a line without breakdowns is an indicator for the performance of the backup strategy options investigated. The remaining 120 solution replications are for the investigation of the 5%, 10% and 15%-line breakdown scenarios. Each of the 5%, 10% and 15%-line breakdown scenario is replicated 40 times to configure a significant mean cycle time for the scenario. As mentioned in the capability matrix, the robots will execute one, two or three tasks and therefore the cycle time will be higher for breakdowns, if a robot with three or two allocated tasks breaks down compared to the robots with one tasks. Due to this fact a significant mean cycle time is required for the comparison of the breakdown

strategy. 40 replications for each breakdown scenario generate a signification mean cycle time which will be shown in the following analysis part.

The step 2 defines the scenario which the algorithm has to consider in one run. The scenario definition means that the algorithm validates the following variables:

- Are workstation robots broken?
- Which workstation robots are broken?
- Are backup robots used?
- Can the backup robots cover all the broken robots?
- If the backup robots cannot cover each broken workstation robot, how many manual repair stations should be used?

After the validation of these variables, the algorithm imports the precedence relationship between tasks, capability and the task time matrices. The matrix import is the end of step 2 and at this point the genetic algorithm can search for the best cycle time in the scenario generated. Thereby, the scenario changes with each solution replication, because the algorithm defines randomly which workstation robots are broken. Only the number of broken workstation robots changes after 40 solution replications beginning with 2 (5%), going to 4 (10%) and ending with 6 (15%) broken robots. It has to mentioned that the variables validated are a fixed value until the algorithm goes back to step 2 again. For examples, if robot 7 and 20 break down for the 5%-line breakdown, the algorithm will search for the lowest cycle while robot 7 and 20 are broken. Only if the algorithm goes back to step 2, two other robots will break down

randomly for the 5%-line breakdown scenario and the algorithm starts to search for the lowest cycle time while these two robots are broken.

The third step is the start of the solution search process. The solution search process starts with a verification whether the following genetic algorithm part replicated itself often enough to generate a good solution for the scenario existing. During the tests of the modified genetic algorithm, it was shown that 40 replications of the genetic algorithm process are usually sufficient to generate a good solution for one genetic algorithm run.

GA Replication	Scenario																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	109	128	118	120	129	144	139	128	146	146	196	179	203	215	199	233	215	230	224	234
2	111	127	107	134	105	144	144	144	151	169	198	190	197	209	208	209	231	233	223	227
3	109	113	118	124	124	132	136	136	149	150	197	190	212	207	216	216	209	240	213	204
4	109	111	106	118	108	140	128	138	155	163	217	200	211	192	207	210	211	233	208	228
5	100	124	109	104	120	146	144	149	142	143	206	186	209	195	195	218	215	233	224	231
6	109	130	117	140	133	139	133	129	144	170	215	198	203	215	204	234	219	220	225	208
7	118	120	105	142	118	132	133	141	137	146	190	198	219	204	216	209	205	239	206	223
8	111	102	98	91	124	134	131	131	144	140	198	186	202	189	199	217	239	230	213	229
9	110	127	110	124	100	144	132	136	146	162	185	175	195	204	223	221	240	255	229	210
10	109	123	118	98	130	133	132	151	155	138	210	197	212	198	201	228	238	230	205	232
11	110	110	100	116	117	136	140	134	140	154	179	187	203	192	198	233	196	226	230	234
12	114	127	110	130	105	134	133	129	142	155	198	185	206	209	200	234	234	230	224	208
13	110	98	108	118	129	132	133	141	142	139	212	181	197	184	231	214	243	230	208	227
14	119	120	110	129	98	139	142	143	137	148	184	190	208	219	209	217	208	224	223	210
15	107	124	110	118	118	132	133	142	150	164	195	186	212	192	201	236	239	240	223	228
16	100	111	112	100	132	139	133	137	142	141	182	190	208	200	208	223	227	238	210	238
17	109	102	109	115	98	133	133	144	147	170	179	187	199	187	216	225	229	255	230	208
18	110	125	114	118	122	139	133	138	143	145	215	179	210	203	192	209	239	233	220	229
19	110	123	118	116	118	129	132	133	142	134	208	198	211	191	240	235	223	215	205	234
20	98	131	106	118	123	132	133	137	140	164	190	190	195	219	216	220	236	233	227	228
21	112	122	109	98	124	144	133	140	146	146	216	198	199	204	201	209	230	230	205	204
22	115	127	119	126	117	134	132	153	140	166	184	187	189	216	240	212	222	225	210	227
23	100	110	118	134	134	127	136	144	150	134	208	186	212	199	216	237	196	230	221	232
24	109	127	91	104	131	136	136	137	142	167	219	195	195	191	235	204	229	238	208	208
25	107	102	108	132	126	144	133	143	151	147	217	198	208	216	216	216	221	230	223	223
26	113	120	100	123	91	139	133	144	142	164	206	177	197	187	225	222	231	215	208	228
27	114	116	110	98	123	133	132	139	148	129	179	198	189	208	223	221	205	240	224	228
28	107	112	112	124	107	132	132	144	150	161	218	186	199	206	220	213	237	235	206	229
29	91	124	110	118	118	137	134	144	142	157	217	184	208	184	205	225	206	227	228	227
30	110	132	118	127	129	144	133	135	134	139	193	187	197	203	234	210	227	230	229	234
31	107	102	114	118	130	133	136	128	146	141	185	175	203	204	203	234	208	230	224	210
32	115	127	98	142	98	132	136	131	148	147	199	198	210	192	241	214	199	246	201	227
33	100	110	101	118	126	134	133	144	152	158	215	198	203	209	212	216	217	230	210	229
34	109	124	110	141	91	140	129	140	134	163	189	177	217	218	195	218	196	240	223	204
35	117	120	111	100	100	146	132	145	148	159	200	198	206	199	221	209	208	222	205	238
36	118	91	111	130	122	140	140	141	144	165	174	193	197	208	189	233	215	235	224	228
37	110	122	109	129	128	140	133	148	131	149	198	184	203	218	209	229	238	230	222	218
38	109	91	110	130	100	147	142	132	145	164	194	198	210	215	217	214	222	238	224	231
39	110	128	128	118	131	151	132	141	144	139	208	198	203	219	210	209	244	240	229	236
40	91	128	104	127	118	134	133	135	153	165	187	184	204	185	205	220	240	249	228	228

Table 4.4: Genetic algorithm replication test 1

41	100	128	117	122	135	138	132	137	140	143	192	185	206	189	191	225	200	230	223	239
42	109	122	109	131	118	145	136	141	142	135	190	198	209	192	199	224	226	237	227	227
43	109	130	118	139	129	134	133	135	153	152	210	191	200	197	192	233	196	240	223	235
44	114	123	107	135	130	133	133	141	144	148	221	187	203	204	200	221	244	215	221	228
45	109	122	119	120	139	134	131	144	142	143	217	205	198	219	212	233	208	230	222	210
46	110	122	118	127	123	139	136	138	137	150	196	187	203	216	202	219	226	233	223	234
47	110	128	110	123	129	133	133	148	131	157	214	185	209	204	189	221	224	250	227	238
48	110	127	118	130	118	141	132	143	151	159	198	198	217	191	218	217	237	220	227	228
49	107	122	114	137	133	132	133	136	152	167	208	177	199	209	201	222	202	228	223	230
50	110	120	110	118	120	136	136	144	144	156	192	195	189	189	206	239	228	230	229	228
51	109	105	91	139	91	134	139	131	146	145	208	198	202	223	219	217	233	228	229	208
52	109	117	100	135	118	127	133	141	144	140	179	176	212	219	216	231	198	232	220	229
53	111	91	110	131	130	132	132	132	145	167	217	185	217	215	200	209	219	215	221	228
54	110	127	117	125	137	133	133	141	142	149	193	190	200	203	235	229	213	230	223	231
55	109	122	109	100	129	131	133	132	144	134	174	203	212	211	216	212	231	227	229	228
56	117	120	110	129	139	140	132	136	140	166	219	175	199	219	204	216	238	229	201	234
57	109	98	109	104	118	144	133	136	151	147	218	187	217	216	195	221	208	230	224	232
58	111	116	114	134	129	139	132	140	146	157	212	198	212	184	223	216	229	222	224	231
59	109	127	107	140	134	136	132	133	142	148	198	190	195	211	240	218	238	233	225	234
60	117	128	109	129	107	144	132	144	153	129	185	198	213	223	216	216	206	248	223	223
61	110	123	109	118	95	132	142	128	144	149	208	202	197	216	230	232	225	240	221	228
62	111	127	114	118	118	132	133	139	142	163	198	184	202	220	217	209	212	230	224	227
63	106	115	119	104	122	132	133	131	150	166	196	177	203	194	215	217	223	223	220	234
64	109	116	109	123	118	140	144	131	137	144	213	198	206	185	234	222	196	249	224	222
65	109	127	110	130	122	133	143	141	147	149	217	186	204	199	209	204	239	233	212	210
66	112	120	110	91	123	129	132	142	142	139	205	186	212	209	204	233	202	240	216	227
67	110	110	109	139	122	136	133	145	144	169	213	198	189	203	210	218	223	249	216	231
68	110	110	118	118	107	134	132	142	150	153	187	198	203	223	218	239	212	253	222	228
69	100	116	117	129	117	132	142	134	145	166	198	194	204	199	224	221	229	228	224	234
70	117	91	110	100	91	136	142	140	152	157	203	185	203	216	232	209	220	233	230	214
71	111	122	118	124	111	133	136	137	150	163	216	198	197	187	208	227	200	248	210	227
72	110	122	114	134	100	133	133	145	140	159	189	177	212	216	195	228	224	248	230	227
73	115	105	110	122	139	129	133	136	152	144	215	198	217	203	225	233	243	235	222	228
74	118	128	91	104	134	133	133	138	144	129	205	195	204	217	216	232	206	215	224	229
75	110	120	106	128	124	133	134	141	148	151	196	197	195	197	238	217	237	230	213	234
76	113	110	109	98	134	140	133	136	140	140	208	200	203	201	190	210	223	238	229	204
77	119	120	100	118	120	136	133	146	152	152	193	175	203	218	240	224	208	230	223	223
78	109	124	114	129	122	135	136	139	144	169	179	187	199	190	189	222	220	222	208	229
79	107	105	118	142	131	144	132	144	131	134	196	186	204	192	227	205	239	233	216	210
80	110	116	110	118	118	132	142	134	137	147	203	193	210	203	205	216	212	238	220	222

Table 4.5: Genetic algorithm replication test 2

Table 4.4 and 4.5 illustrates 80 genetic algorithm replications for 20 scenarios. The variables are fixed within a scenario which means that the algorithm searches 80 times for the lowest cycle time for specific broken robots. For example, if the robots 7 and 24 break down, which is a scenario, the algorithm will search 80 times for the lowest cycle time while robots 7 and 24 are broken. The first five scenarios searches for the lowest

cycle time for a line flow without breakdowns and they use in each scenario a different backup strategy option of the one, two, three and for backup stations options generated. In total, 400 genetic algorithms replications are performed to find the best cycle time for line flow without breakdowns and the lowest cycle time is 91 seconds in the first five scenarios. 400 genetic algorithm replications are a large size of replications and therefore 91 seconds is the lowest cycle time for a line flow without breakdowns by using the generated precedence relationship between tasks, capability and the task time matrices as constraints. The lowest cycle is used to validate the number of genetic algorithm required to find an acceptable trade of between a good solution for a scenario and the computing time required to generate a good solution. Table 4.4 shows that a good solution is found in the first 10 and 20 genetic algorithm replication, which are highlighted yellow and the best solution found for a scenario are highlighted green. It is shown that the best solution is mostly find between the number of 20 and 40 solution replications for the first time and sometimes the best solution is even found in the first 20 genetic algorithm replications as shown in the scenarios 4,7,12,17 and 20. Table 4.5 shows that in the 41 to 80 genetic algorithm replications a better solution than in the first 40 replications could not be found. Due to these facts more than 40 are not necessary to find a good solution for a scenario in an acceptable amount of computing, even if 16 hours are required for one backup strategy option investigation to find the solutions for the 121 scenarios generated by using 40 genetic algorithm replications.

However, if the genetic algorithm part does 40 replications, it will usually have the best solution for the scenario existing which is the step 14 in Figure 4.7. Afterwards, the algorithm will go back to the step 1 to verify whether enough solution replications are

done and if not enough solution replications are done, the algorithm will generate a new scenario.

The fourth step is the start of the genetic algorithm which is adapted from Shirazi and modified for the U-shaped line backup strategy. In this step, the solution search process starts with the generation of the initial population size. In Section 2.5.2, an example from literature is given for the genetic algorithm which uses the population size of four parents. This thesis uses a population size of 100 parents for the solution search. The reason for this large population size is to find a solution which is the best or at least near the best solution for the scenario investigated in an acceptable search time. A large population size offers more starting points for the solution search and increases the chance of finding the best solution. Section 2.5.2 mentioned that one of the major advantages of the genetic algorithm is the random solution starting point. Shirazi uses this advantage in his genetic algorithm. The algorithm generates 72 randomly distributed numbers in the range from 1 to 72. Each number represents one task which has been defined in the task time, task precedence relationship and capability matrices. Figure 4.8 illustrates an example of a randomly generated parent gene.

[54,55,28,16,29,10,25,32,6,58,57,71,14,11,1,53,3,1,2,30,59,23,56,42,45,24,20,63,52,12,41,65,22,68,38,46,31,13,43,47,72,67,15,7,21,70,27,19,9,50,60,62,64,34,49,18,36,48,69,39,35,17,44,3,37,33,5,40,26,66,61,4,8]

Figure 4.8: Randomly generated parent gene

The parent gene in Figure 4.8 has a different structure as the parent example in Section 2.5.2, which consist only of zero and ones. The reason for this is that the example in Section 2.5.2 uses a binary code structure numbers and the genetic algorithm

allows the use of numerical numbers. Nevertheless, the randomly generated parent does not fulfill the constraints of the precedence relationship between task and capability matrices. Thus the randomly generated population size has to be modified. This will be done the following two steps.

The step 5 puts the population size in a feasible order which fulfill the requirements of the precedence constraints. This is implemented by a function called ***Order*** which was designed by Shirazi. The function takes each of the generated parents separately and validates their structure. If the function identifies a task that is in front of its predecessor, the ***Order*** function will take this task and place it after its predecessor. This operation will be repeated until each task is placed after their predecessor and such the whole population size will consist of parents which fulfill the requirements of the precedence constraint. The fulfillment of the capability matrix constraint is realized in the cycle time of the search process. This process is the step 6 in the modified algorithm.

The step 6 consists of three function which are necessary to generate a feasible cycle time for the scenario investigated. The first function is called ***makeStations*** and it cuts randomly each parent in a specific number of pieces. This specific number of pieces represents the workstations used and the length of each piece represents the number of tasks which should be executed by the workstation. Each of these cuts is done randomly to underline the basic ideas of genetic algorithms to start a random solution search, but the sum of the length of each piece randomly generated has to be 72 to match the initialization requirements of 72 tasks. Furthermore, the number of these pieces varies in each U-shaped line backup strategy option. Each option has the same number of ten workstations for a line flow without failures, but for breakdowns the backup stations

have to be entered to execute the tasks of the broken robots. This entry has to be considered as an additional station in the genetic algorithm and therefore the number of pieces varies for the robotic U-shaped line backup strategy options for breakdowns.

Here appears the question why the backup robots are not used to support the line flow. In Section 4.1 is mentioned that one advantage of backup robots is the support of workstations with an above average workload for a line flow without breakdowns. This advantage cannot be considered in this research. The reason for this is that the support of workstations would lead to an unequal comparison. For a line flow without breakdowns the option using four backup stations would have 48 robots and the option using one backup station only would have 42 robots. Consequently, the four backup stations option would have a lower cycle time for the line flow without breakdowns, because it would have more robots than the other options. Furthermore, if the backup robots are used the whole time, they will also break down. Thus the backup robots would not be an efficient breakdown strategy. Due to these facts the backup robots will be used only for the coverage of broken robots.

Nevertheless, the genetic algorithm designed by Shirazi has to be modified to consider the several backup options for the U-shaped line. The modified algorithm generates for a line flow without breakdowns ten pieces which represents the ten workstations. Figure 4.9 illustrates an example of a parent which is randomly cut in ten pieces for the allocation to a specific workstation. The first piece is allocated to workstation 1, the second piece is allocated to workstation 2 and so on. Furthermore, each piece consists of a fixed number of a fixed number tasks which have to be done by the allocated workstation. In Figure 4.9 the first piece contains the tasks 1,2,3,4,5 and is

allocated to the first workstation. The following function will verify whether the workstations are able to execute all the allocated tasks.

1st piece is allocated to workstation 1: [1, 2, 3, 4, 5]
2nd piece is allocated to workstation 2: [6, 7, 8, 9, 10, 11, 12, 13, 14]
3rd piece is allocated to workstation 3: [15, 16, 17, 18, 19, 20, 21, 22]
4th piece is allocated to workstation 4: [21, 22, 23, 24]
5th piece is allocated to workstation 5: [25, 26, 27, 28, 32, 33]
6th piece is allocated to workstation 6: [34, 35, 36, 29, 30, 31, 37, 38, 39, 40, 41, 42, 43, 44]
7th piece is allocated to workstation 7: [45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]
8th piece is allocated to workstation 8: [56, 57, 58]
9th piece is allocated to workstation 9: [64, 59, 60, 61, 62, 63]
10th piece is allocated to workstation 10: [65, 66, 67, 68, 69, 70, 71, 72]

Figure 4.9: Example of ten piece allocated to a specific workstation

Figure 4.9 illustrates ten pieces which are allocated to one workstations only. In the previous Section it is mentioned that the backup station can be entered twice. The *makeStations* function realizes the double entry of a backup station by generating for each backup station used two additional cuts in the parent gene. Thus the *makeStations* function cuts the following number of pieces for the robotic U-shaped line backup strategy options for breakdowns:

- One backup station: 12 pieces
- Two backup stations: 14 pieces
- Three backup stations: 16 pieces
- Four backup stations: 18 pieces

1st piece is allocated to workstation 1: [1, 2, 3, 4, 5]

2nd piece is allocated to backup station A: [6,7,8]

3rd piece is allocated to workstation 2: [9, 10, 11, 12, 13, 14]

4th piece is allocated to workstation 3: [15, 16, 17, 18, 19, 20, 21, 22]

5th piece is allocated to workstation 4: [21, 22, 23, 24]

6th piece is allocated to workstation 5: [25, 26, 27, 28 32, 33]

7th piece is allocated to workstation 6: [34, 35, 36, 29, 30, 31 37, 38, 39, 40, 41, 42, 43, 44]

8th piece is allocated to workstation 7: [45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]

9th piece is allocated to workstation 8: [56, 57, 58]

10th piece is allocated to workstation 9: [64, 59, 60, 61, 62, 63]

11th piece is allocated to backup station A: [65,66]

12th piece is allocated to workstation 10: [67, 68, 69, 70, 71, 72]

Figure 4.10: Example of 12 pieces for the one backup station option A

Figure 4.10 shows an example of 12 pieces for the one backup station option A. The backup station A can be entered after workstation 1 and 9. Thus the *makeStations* allocates the second and the eleventh piece to the backup station A. It is not verified that the allocated tasks are from the broken robots. This verification will be done by the second function of step 6.

The usage of additional cuts to handle the double entry of a backup station, allows the use of Shirazi's genetic algorithm for the robotic U-shaped line backup strategy. Hence, the algorithm has to be extended that the work time of backup stations are summed, which are used twice. This extension is realized in the end of the of this step where the cycle time is calculated. Manual repair stations do not get additional cuts. If manual repair stations have to been used, the modified algorithm places a manual repair

station in the workstation which requires a manual repair station. At the end the work time of the workstation and manual repair station will be summed. Thus a manual repair station can be placed after each workstation and the genetic algorithm of Shirazi is still usable.

The next function used in the step 6 is the self-generated *fescheck* function. This function fulfills the requirements of the capability matrix. Each piece randomly generated by the *makeStation* is allocated to a specific workstation and consist of several tasks. The self-generated *fescheck* function takes each piece separately and validates whether the number of tasks can be executed by this specific workstation. If some of the allocated task cannot be executed by their allocated workstation, the function will allocate these task to a nearby workstation, which can perform these task: Thus the precedence relationship between tasks constraint is still fulfilled and additionally the constraint is fulfilled that each task is allocated to a robot, which can execute the allocated tasks. At the end of the *fescheck* function, each generated station has a feasible number of tasks.

1st piece is allocated to workstation 1: [6, 3, 7, 1, 8, 4, 2, 5]
2nd piece is allocated to workstation 2: [11, 10, 13, 9, 14, 12]
3rd piece is allocated to workstation 3: [19, 16, 20, 17, 18, 15]
4th piece is allocated to workstation 4: [25, 22, 28, 24, 21, 23, 26, 27]
5th piece is allocated to workstation 5: [36, 31, 32, 30, 35, 29, 34, 33]
6th piece is allocated to workstation 6: [40, 42, 44, 41, 37, 43, 38, 39]
7th piece is allocated to workstation 7: [51, 47, 45, 49, 46, 52, 48, 50]
8th piece is allocated to workstation 8: [53, 57, 55, 54, 56, 58]
9th piece is allocated to workstation 9: [64, 62, 61, 59, 63, 60]
10th piece is allocated to workstation 10: [67, 68, 72, 69, 65, 71, 70, 66]

Figure 4.11: Feasible number of tasks on each workstation

Figure 4.11 illustrates the ten workstations scheduled with a feasible number of tasks for a line flow without breakdowns. It can be seen that each workstation executes the number of six or eight tasks. Furthermore, the tasks fulfill the precedence constraints. It seems that the tasks are more randomly structured than in the Figure 4.9, because task 6 on workstation is listed in front of the tasks 1,2,3,4 and 5. Nevertheless, the last function will demonstrate that all the tasks in Figure 4.11 are in a feasible order.

The last function of the step 6 is named as ***LengthPiece*** and it counts the work time of each workstation. The ***LengthPiece*** function uses the allocated number of tasks for each workstation and schedules each task to one robot which is able to execute the task. The robot with the highest work time defines the work time of a workstation. The ***LengthPiece*** function is modified for the robotic U-shaped line backup strategy options

compared to the initial *LengthPiece* function of Shirazi's genetic algorithm. This modification includes a verification whether a task can be executed by several robots and chooses only a robot of the current workstation for which the work time is counted. The verification is necessary, because some workstations can be covered by two backup stations or by two manual repair station which is shown in the previous Section. Thus the verification ensures that each task is only done by one robot on the current workstation. Figure 4.12 shows the allocation of each tasks to one workstation robot only for a line flow without breakdowns.

```

Workstation 1: {Robot 1: [1, 4], Robot 2: [2, 5], Robot 3: [6], Robot 4: [3, 7, 8]}
Workstation 2: {Robot 5: [9, 14], Robot 6: [12], Robot 7: [11], Robot 8: [10, 13]}
Workstation 3: { Robot 9: [15, 17], Robot 10: [16], Robot 11: [18, 20], Robot 12: [19]}
Workstation 4: {Robot 13: [21, 23], Robot 14: [25, 26, 27], Robot 15: [24] Robot, 16: [22, 28]}
Workstation 5: {Robot 17: [34], Robot 18: [33, 29], Robot 19: [35, 30], Robot 20: [32, 36, 31]}
Workstation 6: {Robot 21: [37, 43], Robot 22: [38, 39], Robot 23: [41], Robot 24: [40, 42, 44]}
Workstation 7: {Robot 25: [45, 46, 48], Robot 26: [49, 50], Robot 27: [47, 52], Robot 28: [51]}
Workstation 8: {Robot 29: [54], Robot 30: [55, 56], Robot 31: [58], Robot 32: [53, 57]}
Workstation 9: {Robot 33: [64], Robot 34: [59, 60], Robot 35: [61], Robot 36: [62, 63]}
Workstation 10: {Robot 37: [70, 72], Robot 38: [65], Robot 39: [66, 67, 68], Robot 40: [69, 71]}

```

Figure 4.12: Allocation of each tasks to one workstation robot

It can be seen that the most robots execute two tasks. Some robots execute one task and a few robots executes three tasks which is an above average workload in this scenario. Furthermore, it can be seen that robot 3 executes the task 6 only. Thus the task 6 could be placed in front of the tasks 1,2,3,4 and 5 in the previous *fescheck* function and a feasible order was still ensured.

After the *LengthPiece* function, each of the stations have a work time. At this point it has to be analyzed whether a backup station is used twice. If a backup robot is used twice, their work time will be summed to generate the real work time for this backup station. Furthermore, the station with the highest work time defines the cycle time for the investigated parent. The cycle time is the objective function for the modified genetic algorithm and the lower the cycle time of a parent is, the better is its performance. Step 6 in the algorithm ends with the computation of the cycle time. Figure demonstrates the work time of each robot. The yellow underlined robots define the worktime for a workstation, because they have the highest work time and the red underlined robot 4 on workstation 5 fine the cycle time of 91 seconds for the solution, because it is the robot with the highest work time.

Figure 4.13: Work time for each robot

The seventh step is the verification whether the genetic algorithm generated enough offspring generations for one genetic algorithm replication. This modified algorithm uses 100 offspring generations for one genetic algorithm run. The combination of a population size of 100 parents, 100 offspring generation and 40 genetic algorithm replication is able to generate a good solution in an acceptable computing time. The scope of this thesis is not to improve the genetic algorithm of Shirazi, but to generate a backup strategy for the U-shaped line. Thus it is acceptable to use the combination of a population size of 100 parents, 100 offspring generation and 40 genetic algorithm replication, even if a faster combination exists to generate a good cycle time for each scenario. Nevertheless, if the genetic algorithm generates 100 offspring generations, one replication of the genetic algorithm will be done and the modified algorithm goes back to the step 3. If the algorithm generates less than 100 offspring generations, it will go to the step 8.

In this step the genetic algorithm uses only the five parents with the lowest cycle time to generate an offspring generation. The use of five parents only to generate an offspring is an adaption of Shirazi's genetic algorithm which has the best performance by using five parents only for the offspring generation. The parents are generated in a list called ***Pop*** which structures the 100 parents generated by their cycle time. On the first position in the ***Pop*** list is the parent with the lowest cycle, followed by the parent with the second lowest cycle time and so on. The offspring generated are saved and structured in the ***Pop*** list as well which allows to use the five parents with the lowest cycle time in each offspring generation.

The step 9 is the offspring generation. The offspring generation bases on the initial genetic algorithm of Shirazi and use a fast method to perform crossover and mutation. This fast methods of the offspring generation will not be explained here, but a detailed explanation can be found in Shirazi's research [53] and they will be explained in his Ph.D. Thesis, which will be published in the end of August 2016. Nevertheless, the basic idea how crossover and mutation are performed is illustrated in Section 2.5.2. After the offspring generation, each offspring has to be structured in a feasible order.

The structure of the offspring in a feasible order is step 10 in the modified algorithm. This step uses the same *Order* function as the step 5 to structure the offspring. Afterwards, the step 11 follows which generate a cycle time for each offspring.

The step 11 uses the same three *makeStations*, *fescheck* and *LengthPiece* functions as the step 6 to generate a cycle time for the offspring. The offspring or parent with the lowest cycle time is the current best solution for one offspring generation and the step 12 in the modified algorithm.

The step 12 saves the cycle time and the structure of the solution with the lowest cycle time, which is shown in Figure 4.12, in a list called *cbresponse*. During the first offspring generation, the *cbresponse* list is empty and it saves the solution from the first place in the *Pop* list which is structured and has the solution with the lowest on the first place. During the remaining 99 offspring generations, it is validated if the solution on the first position of the *Pop* list is smaller than the solution of the *cbresponse* list. If the solution on the first position of the *Pop* list is smaller than the solution of the *cbresponse* list, the *cbresponse* list overwrite its current value with the value of the solution on the first position of the *Pop* list. Otherwise, the *cbresponse* list keeps its value.

After the step 12 the algorithm goes back to the step 7 to verify whether enough offspring generations have been generated. The algorithm uses the number of the current offspring generation and verifies whether this number equals 100, which is the required number of offspring generations for one genetic algorithm replication. If less than 100 offspring generations have been generated, the genetic algorithm uses the current five best solutions, which can consist of the initial population or previous offspring generations, for the next offspring generation. If 100 offspring generations have been generated, the genetic algorithm reports the parent or offspring with the lowest cycle time which is the best solution for one genetic algorithm replication and the step 13 in Figure 4.7.

The step 13 works in the same ways as the step 12. It consists of a list called *bestopt* which saves the value of the *cbresponse* list for the first run. Afterwards, it validates for the remaining 39 genetic algorithm replication if the cycle time of the solution in *cbresponse* list is smaller than the cycle time of the solution in the *bestopt* list. If the cycle time of the solution in *cbresponse* list is smaller than the cycle time of the solution in the *bestopt* list, the *bestopt* list will other write its solution with the solution of the *cbresponse* list otherwise the *bestopt* list will keeps its solution. The solution of the *cbresponse* list changes in each genetic replication run, because the step 13 leads to the step 3 which is the verification whether enough genetic algorithm replications have been done. If not enough genetic algorithm replications have been done, the values of the *Pop* list, *cbresponse* list will be deleted and the modified algorithm will go to the step 4 otherwise it will go to the step 14.

The Step 14 is the solution for the current scenario investigated which is saved in the *cbresponse* list. At this point, the *cbresponse* list has the best solution for one scenario generated. This solution is printed to illustrate the cycle time and the structure of the best solution for the generated scenario. Furthermore, the number of the broken robots is printed which are broken in the investigated scenario and here ends the generated scenario. The algorithm goes automatically to the Step 1 which verifies whether enough solution replications have been done. As mentioned above, the modified algorithm investigates for each backup strategy option the breakdown scenario of a 5%, 10% and 15%-line breakdown. Each scenario has 40 replications and the cycle time for line flow without breakdowns will be investigated as well. Thus the modified algorithm does 121 solution replications. If not enough solution replications have been done, the modified algorithm will go to the step 2 otherwise it will go to the step 15 which is the end of the algorithm. In the step 2 a new scenario is generated there the robots break down randomly and the algorithm starts to search for the lowest cycle time while these robots are broken. Figure 4.7 illustrates only the framework of the modified algorithm for the U-shaped line backup strategy research. The whole algorithm is shown in Appendix 3.

The modified algorithm is able to investigate each option of the robotic U-shaped line backup strategy. Afterwards, the cycle times of each option can be compared with each other to find the best option for the robotic U-shaped line backup strategy which will be done in Chapter 5. Nevertheless, the question arises whether the U-shaped line backup strategy generated is better than existing backup strategies. Hence, the robotic U-shaped line backup strategy will be compared with other breakdown strategies.

The inventory breakdown strategy has a discrepancy with the basic idea of a U-shaped line layout as a part of lean manufacturing and will not be used in this methodology.

Kahan et al. uses several breakdown strategies in their research which offers the potential for a cycle time comparison [50]. One of these breakdown strategies is the coverage of each broken workstation with manual repair station only [50]. Another breakdown strategy of Kahan et al. is the workload reallocation of a broken robot by working robots downstream the line. As mentioned in Section 3.3, the workload reallocation downstream the line has the best performance in research of Kahan et al. [50]. Due to this fact the manual repair stations only and workload allocation downstream the line breakdown strategies will be used for a comparison with the robotic U-shaped line backup strategy. These breakdown strategies could be realized in the modified algorithm and therefore a realistic comparison can be done.

The breakdown strategy designed by Shirazi et al. will not be considered in the comparison, because their breakdown strategy is still in research. The improved breakdown strategy of Shirazi et al. will be available in the end of August 2016 and afterwards a comparison between the robotic U-shaped backup strategy and the robotic straight line backup strategy can be performed.

CHAPTER 5

ROBOTIC U-SHAPED LINE BACKUP STRATEGY ANALYSIS

In the previous chapter, the robotic U-shaped line backup strategy is described in detail. There are four options for the placement of backup stations. This chapter will identify the best option and the number of backup stations for the robotic U-shaped line backup strategy. For this purpose, this chapter will consist of six analyses:

1. The first analysis identifies the best placement for one backup station.
2. The second analysis identifies the best placement for two backup station.
3. The third analysis identifies the best placement for three backup stations.
4. The fourth analysis compares the backup station options with current breakdown strategies.
5. The fifth analysis uses a higher task execution time on manual repair stations to shows the performance of the breakdown strategies for more complex products.
6. The sixth analysis uses a lower task execution time on manual repair stations to shows the performance of the breakdown strategies for standardized products.

The last analyses use the best placement for one, two and three backup stations to compare them with the performance of four backup stations. Furthermore, the most efficient robotic U-shaped line backup strategy will be compared with the breakdown strategies of Kahan et al. which are the usage only manual repair stations of and the

workload reallocation of broken robots by working robots downstream the line [50]. These last analyses should illustrate the potential of the robotic U-shaped backup strategy in comparison with current line balancing breakdown strategies. The performance of each breakdown strategy will be evaluated by using essential criteria. As mentioned in the previous chapter, the first criterion is the cycle time which is used in the literature as benchmark for breakdown strategies. The second criterion is the generated product quality from the considered breakdown strategies. It is difficult to describe the generated product quality with the modified genetic algorithm, because the optimization goal is the cycle time minimization only. Therefore, the product quality will be described by the number of tasks which are executed by the manual repair stations.

5.1 One backup station analysis

Graph 5.1: Mean cycle time for the use of one backup station

Graph 5.1 shows the mean cycle time and the standard deviation of the mean for the use of one backup station on the four placement options in the robotic U-shaped line backup strategy design. The mean cycle time and its standard deviation are calculated for the 5%, 10% and 15%-line breakdown scenarios. Each breakdown scenario was replicated 40 times and the robot failures were generated randomly. Out of the 40 replications, the mean cycle time and the standard deviation of the mean cycle time are calculated. This procedure is applied for the following analyses as well. The cycle time for the line flow without breakdowns is the optimum cycle for each breakdown strategy, because the initial data is the same for each breakdown strategy to generate a realistic comparison. Due to this fact, the cycle time for a line flow without breakdowns will be presented in this analysis only and it has the amount of 91 seconds.

The one backup station option A has the mean cycle time of 224 seconds with a standard deviation of 8.01 seconds, the option B has the mean cycle time of 204 seconds with a standard deviation of 11.07 seconds, the option C has the mean cycle time of 201 seconds with a standard deviation of 11.18 seconds and the option D has the mean cycle time of 193 seconds with a standard deviation of 7.02 seconds for the 5%-line breakdown scenario. Afterwards, the option A has the mean cycle time of 239 seconds with a standard deviation of 7.37 seconds, option B has the mean cycle time of 265 seconds with a standard deviation of 8.09 seconds, option C has the mean cycle time of 246 seconds with a standard deviation of 7.78 seconds and option D has the mean cycle time of 208 seconds with a standard deviation of 5.91 seconds for the 10%-line breakdown scenario. For the 15%-line breakdown scenario, the option A has the mean cycle time of 256 seconds with a standard deviation of 5.75 seconds, option B has the

mean cycle time of 267 seconds with a standard deviation of 5.99 seconds, option C has the mean cycle time of 272 seconds with a standard deviation of 6.29 seconds and option D has the mean cycle time of 222 seconds with a standard deviation of 5.43 seconds.

It is striking that the cycle time increases dramatically from the without breakdowns line flow to the 5%-line breakdown scenario. This is due the fact that the one backup station options require the use of manual repair stations which need three times longer to execute tasks. The error bars show that option B, C, D are similar and option A has a significant higher mean cycle time than the options B, C, D for the 5%-line breakdown. For the 10% and 15%-line breakdown, the option D has a significant lower mean cycle time than the option A, B, C and therefore option D performs as best of the various one backup station options. It has a 13.8% lower cycle time than the option A, a 5.7% lower cycle than the option B and a 4.0% lower cycle time than the option C for the 5%-line breakdown scenario. The cycle time of option D increases slightly to 208 seconds for the 10%-line breakdown scenario and performs 13.0% better than the option A, 21.5% better than the option B and 15.5% better than the option C. The option D performs also better than the other options for the 15%-line breakdown scenario. It has a 15.3% lower cycle time than the option A, a 16.9% lower cycle time than option B and a 18.4% lower cycle time than option C. The reason for the lowest cycle time of option D is that it is the only backup station which can cover workstation 5. The workstation 5 has the highest work time of all workstations in the generated line balancing problem and therefore it defines the cycle time for a line flow without breakdown. Thus a failure of the workstation 5 robots has the highest impact on the cycle time. A higher number of robot failures increases the probability that a robot of workstation 5 will break which

explains the increasing cycle time of the option D for the 10% and 15%-line breakdown. Nevertheless, a more than twice higher cycle time for the 5%-line breakdown compared to the line flow without breakdowns validates the worse performance of manual repair stations. Whether the option D has the best performance of the one backup station options, can be identified after the analysis of the generated product quality. Graph 5.2 illustrates the mean number of tasks and the standard deviation of the mean number of tasks which are executed on the manual repair stations for the breakdown scenarios considered.

Graph 5.2: Product quality analysis for the one backup station options

The mean number of tasks and their standard deviation are calculated in the same way as the mean cycle time and its standard deviation. Each breakdown scenario was replicated 40 times and out of the 40 replications the mean number of tasks and the standard deviation of mean of tasks, which are executed on the manual repair stations, is calculated. For a line flow without breakdowns no tasks have to be executed on the

manual repair stations. Due to this fact the line flow without breakdowns will not be considered in following product quality analyses.

For the one backup station case, option A results in the mean number of 2.4 tasks with a standard deviation of 0.20 on the manual repair stations, the option B has 2.1 tasks with a standard deviation of 0.25, the option C has 2.0 with a standard deviation of 0.23 tasks and the option D has the mean number of 2.4 tasks with a standard deviation of 0.33 tasks on the manual repair stations for the 5%-line breakdown scenario. For the 10%-line breakdown, option A has the mean number of 4.4 tasks with a standard deviation of 0.31 tasks, option B has the mean number of 4.4 tasks with a standard deviation of 0.33 tasks, option C has the mean number of 4.0 tasks with a standard deviation of 0.32 tasks and option D has the mean number of 3.9 tasks with a standard deviation of 0.24 tasks on the manual repair stations. For the 15%-line breakdown scenario, option A has the mean number of 6.9 tasks with a standard deviation of 0.36 tasks, option B has the mean number of 6.7 tasks with a standard deviation of 0.43 tasks, option C has the mean number of 6.9 tasks with a standard deviation of 0.38 tasks and option D has the mean number of 6.0 tasks with a standard deviation of 0.35 tasks on manual repair stations.

The error bars show that the options A, B, C, D are similar to each other in the 5%-line breakdown scenario. For the 10%-line breakdown scenario, the options A, B, C are similar to each other and the options B, C, D are similar to each other, but the option D has a significant lower mean cycle time than the option A. For the 15%-line breakdowns, the error bars show that the options A, B, C are similar to each other and the option D has a significant lower cycle time than the options A, B, C and therefore the options D

offers the best product quality of the one backup stations options. For the 5%-line breakdown scenario, the option C executes 16.8 % fewer tasks than the option A, 6.0% fewer tasks than the option B and 15.1% fewer tasks than the option D on the manual repair stations. This changes in the following scenarios where the option D executes 11.4 % fewer tasks than the option A, 11.4% fewer tasks than the option B and 2.5% fewer tasks than the option C on the manual repair stations for the 10%-line breakdown scenario. For the 15%-line breakdown scenario, the option D executes 13.0 % fewer tasks than the option A, 10.4% fewer tasks than the option B and 14.3% fewer tasks than the option C on the manual repair stations.

The option D has best product quality of the various one backup station, because it covers the workstation with the highest workload as mentioned above. The capability matrix was designed in the way that several robots execute the number of two and three tasks. Hence, the workstation with the highest workload has some of highly loaded robots. Thus the manual repair stations of option D covers in the most breakdown scenario the robots with only one or two allocated tasks. The poor performance of the option D in the 5%-line breakdown is an outlier where the failing robots randomly generated had an above average workload. Option D performs in the cycle time and the product quality analyses as the best of the one backup station options for the robotic U-shaped line backup strategy designed. For this purpose, the option D will be taken as the one backup station option for the last analysis in this chapter.

5.2 Two backup stations analysis

Graph 5.3: Mean cycle time for the use of two backup stations

Graph 5.3 illustrates the mean cycle time and the standard deviation of the mean cycle time for the use of two backup stations on the four placement options in the robotic U-shaped line backup strategy design. The option AB has the mean cycle time of 181 seconds with a standard deviation of 12.61 seconds, option AC has the mean cycle time of 164 seconds with a standard deviation of 13.19 seconds, option AD has the mean cycle time of 126 seconds with a standard deviation of 6.72 seconds, option BC has the mean cycle time of 224 seconds with a standard deviation of 12.71 seconds, option DB has the mean cycle time of 150 seconds with a standard deviation of 13.16 seconds and option CD has the mean cycle time of 161 seconds with a standard deviation of 9.53 seconds for the 5%-line breakdown scenario. Afterwards, the option AB has the mean cycle time of 238 seconds with a standard deviation of 8.94 seconds, option AC has the mean cycle time of 182 seconds with a standard deviation of 12.63 seconds, option AD

has the mean cycle time of 171 seconds with a standard deviation of 6.32 seconds, option BC has the mean cycle time of 262 seconds with a standard deviation of 8.51 seconds, option DB has the mean cycle time of 177 seconds with a standard deviation of 11.46 seconds and option CD has the mean cycle time of 195 seconds with a standard deviation of 10.07 seconds for the 10%-line breakdown scenario. For the last 15%-line breakdown scenario, the option AB has the mean cycle time of 260 seconds with a standard deviation of 5.05 seconds, the option AC has the mean cycle time of 238 seconds with a standard deviation of 9.12 seconds, option AD has the mean cycle time of 175 seconds with a standard deviation of 4.09 seconds, option BC has the mean cycle time of 277 seconds with a standard deviation of 6.25 seconds, option DB has the mean cycle time of 230 seconds with a standard deviation of 8.41 seconds and option CD has the mean cycle time of 220 seconds with a standard deviation of 6.03 seconds. The error bars show that the option AD has a significant lower mean cycle time than the options AB, AC, BC, BD, CB for the 5% and 15%-line breakdown scenario. Thus the option AD performs as best of the two backup stations options. For the 10%-line breakdown scenario, the options AD, AC, BD are similar to each other and they have a significant lower mean cycle time than the options AB, BC.

It is worth noting that the options which use a backup station on position D have a better performance than the options without a placement on the position D. This is due the fact that the backup station on position D covers the workstation with the highest workload which is mentioned in the previous section. Thereby, the option AD has the best cycle time of all the two backup stations options. It has a 30.4% lower cycle time than option AB, a 23.6% lower cycle time than option AC, a 43.8% lower cycle time

than option BC, a 16% lower cycle time than option BD and a 21.7% lower cycle time than option CD for the 5%-line breakdown scenario. Afterwards, the option BD has a 25.6% lower cycle time than option AB, a 6.0% lower cycle time than option AC, a 35.5% lower cycle time than option BC, a 3.4% lower cycle time than option BD and a 12.3% lower cycle time than option CD for the 10%-line breakdown scenario. For the 15%-line breakdown scenario, the option BD has a 32.7% lower cycle time than option AB, a 26.5% lower cycle time than option AC, a 36.9% lower cycle time than option BC, a 24.1% lower cycle time than option BD and a 20.4% lower cycle time than option CD.

The comparison between the best option AD and the other options shows that the option AC has comparable cycle times as the options BD and CD without the usage of a backup station on the position D. This is due the fact that the option AC is able to cover eight of the 10 workstations and therefore it can reach still a cycle time on the same level as the options BD and CD. On the contrary, options AC, AB and BC performs badly, because they can cover six workstations only. Nevertheless, the following product quality analysis will show whether option AD is the best of the two backup station options.

Graph 5.4: Product quality analysis for the two backup stations options

Graph 5.4 shows the mean number of tasks and the standard deviation of the mean number of tasks which are executed on the manual repair stations for the two backup stations options. The two backup stations option AB has the mean number of 3 tasks with a standard deviation of 0.31 tasks, option AC has 0.9 tasks with a standard deviation of 0.19 tasks, option AD has 0.6 tasks with a standard deviation of 0.14 tasks, option BC has 2.1 tasks with a standard deviation of 0.23 tasks, option BD has 0.9 tasks with a standard deviation of 0.21 tasks and option CD has 1.4 tasks with a standard deviation of 0.21 tasks on the manual repair stations for the 5%-line breakdown scenario. Afterwards, the option AB has the mean number of 3.8 tasks with a standard deviation of 0.32 tasks, option AC has 1.2 tasks with a standard deviation of 0.21 tasks, the option AD has 1.4 tasks with a standard deviation of 0.21 tasks, option BC has 3.8 tasks with a standard deviation of 0.30 tasks, option BD has 1.2 tasks with a standard deviation of 0.22 tasks and option CD has 2.5 tasks with a standard deviation of 0.29

tasks on the manual repair stations for the 10%-line breakdown scenario. For the 15%-line breakdown scenario, the option AB has the mean number of 4.9 tasks with a standard deviation of 0.28 tasks, option AC has 2.5 tasks with a standard deviation of 0.32 tasks, option AD has 1.7 tasks with a standard deviation of 0.27 tasks, option BC has 4.9 tasks with a standard deviation of 0.33 tasks, option BD has 2.4 tasks with a standard deviation of 0.28 tasks and option CD has 4.3 tasks with a standard deviation of 0.33 tasks on the manual repair stations. The error bars show that the option AD has significant fewer tasks on manual repair stations than the options AB, AC, BC, BD, CB for the 5% and 15%-line breakdown scenario. Thus the option AD offers the best product quality of the two backup stations options. For the 10%-line breakdown scenario, the options AD, AC, BD are similar to each other and they have significant fewer tasks on manual repair stations than the options AB, BC.

It can be seen that option AD has the best performance in the product analysis as well. The option AD executes 81.7% fewer tasks than option AB, 40.5% fewer tasks than option AC, 73.5% fewer tasks than option BC, 38.9% fewer tasks than option BD and 59.3% fewer tasks than option CD on the manual repair stations for the 5%-line breakdown scenario. Afterwards, the option AD executes 17.9% more tasks than the options AC and BD on the manual repair station, but it executes still 63.2% fewer tasks than option AB, 62.9% fewer tasks than option BD and 43.4% fewer tasks than option CD on the manual repair station for the 10%-line breakdown scenario. The better of performance of the options AC and BD validates the importance of using the backup stations to cover as many workstations as possible. Nevertheless, the option AD has the best performance for the 15%-line breakdown again. It executes 66.2% fewer tasks than

option AB, 34.0% fewer tasks than option AC, 66.5% fewer tasks than option BC, 30.5% fewer tasks than option BD and 61.6% fewer tasks than option CD on the manual repair stations.

This lead to the conclusion that the option AD performs as best of the two backup stations options. Hence, the two backup stations option AD will be used in the last analyses to find the best option for the robotic U-shaped line backup strategy designed.

5.3 Three backup stations analysis

Graph 5.5: Mean cycle time for the use of three backup stations

Graph 5.5 illustrates the mean cycle time and the standard deviation of the mean cycle time for the use of three backup stations on the four placement options in the robotic U-shaped line backup strategy design. The three backup stations option ABC has the mean cycle time of 161 seconds with a standard deviation of 14.12 seconds, option ABD has the mean cycle time of 92 seconds with a standard deviation of 0.43 seconds, option ACD has the mean cycle time of 94 seconds with a standard deviation of 1.51 seconds and option BCD has the mean cycle time of 141 seconds with a standard deviation of 11.71 seconds for the 5%-line breakdown scenario. Afterwards, the option ABC has the mean cycle time of 221 seconds with a standard deviation of 13.89 seconds, option ABD has the mean cycle time of 108 seconds with a standard deviation of 3.74 seconds, option ACD has the mean cycle time of 106 seconds with a standard deviation of 3.24 seconds and option BCD has the mean cycle time of 168 seconds with a standard deviation of 12.99 seconds for the 10%-line breakdown scenario. For the 15%-line breakdown scenario, the option ABC has the mean cycle time of 227 seconds with a standard deviation of 11.38 seconds, option ABD has the mean cycle time of 139 seconds with a standard deviation of 5.32 seconds, option ACD has the mean cycle time of 122 seconds with a standard deviation of 5.40 seconds and option BCD has the mean cycle time of 224 seconds with a standard deviation of 10.05 seconds.

The error bars show that the options ABD, ACD are similar to each other and they have a significant lower mean cycle time than the options ABC, BCD for the 5%, 10% and 15%-line breakdown. Thus the options ABD and ACD perform much better than the options ABC and BCD. The option ABD has a 43.3% lower cycle time than option ABC, a 2.1% lower cycle time than option ACD and a 34.9% lower cycle time than option BCD.

BCD for the 5%-line breakdown scenario. Afterwards, the option ACD performs better. It has a 52.3% lower cycle time than option ABC, a 2.2% lower cycle time than option ABD and a 37.2% lower cycle time than option BCD for the 10%-line breakdown scenario. For the 15%-line breakdown scenario, the option ACD has a 46.1% lower cycle time than option ABC, a 12.0% lower cycle time than option ABD and a 45.4% lower cycle time than option BCD.

The reason for the good performance of the options ABD and ACD is that these options are able to cover breakdowns without the use manual repair stations. Furthermore, the cycle time increase of option ACD from 94 seconds for the 5%-line breakdown to 122 second for the 15%-line breakdown is still near the initial 91 seconds for line flow without breakdowns. This fact evidences the efficiency of backup robots for line breakdowns compared to the use manual repair stations. The following analysis will show how often the options ABC and BCD has to use the manual repair stations.

Graph 5.6: Product quality analysis for the three backup station options

Graph 5.6 shows the mean number of tasks and the standard deviation of the mean number of tasks which are executed on the manual repair stations for the three backup stations options. The three backup stations option ABC has the mean number of 0.9 tasks with standard deviation of 0.20 tasks, option ABD has 0 tasks, the option ACD has 0 tasks and option BCD has the mean number of 0.7 tasks with standard deviation of 0.17 tasks on the manual repair stations for the 5%-line breakdown scenario. Afterwards, the option ABC has the mean number of 1.9 tasks with standard deviation of 0.25 tasks, option ABD has 0 tasks, option ACD has 0 tasks and option BCD has the mean number of 1.6 tasks with standard deviation of 0.33 tasks on the manual repair stations for the 10%-line breakdown scenario. For the 15%-line breakdown scenario, the option ABC has the mean number of 2.6 tasks with standard deviation of 0.33 tasks, option ABD has 0 tasks, option ACD has 0 tasks and option BCD has the mean number of 2.3 tasks with standard deviation of 0.28 tasks on the manual repair stations.

The error bars show that the options ABD, ACD are similar to each other and they have significant fewer tasks on manual repair stations than the options ABC, BCD for the 5%, 10% and 15%-line breakdown. Thus the options ABD and ACD offer a higher product quality than the options ABC and BCD, because they do not have to use the manual repair stations. The options ABC and BCD may be able to offer a good product quality as well, because the mean number of tasks on the manual repair station is 1 for the 5%, 2 for the 10% and 3 for the 15%-line breakdown scenario.

The number of 3 tasks executed by manual repair stations should be still be able to offer an acceptable product quality for the most products. Nevertheless, the number of 3 tasks executed by manual repair stations may be too much to offer an acceptable for

products which require a premium quality. Furthermore, the execution of one task on the manual repair station increases the cycle time dramatically which is shown in Graph 5.5 for the options ABC and BCD. Hence, the option ACD will be used in the following analysis, because it has the best performance of the three backup stations options for the robotic U-shaped line backup strategy.

5.4 Breakdown strategies comparison

This section will identify the best backup station option for the robotic U-shaped line backup strategy designed. Furthermore, the best robotic U-shaped line backup strategy option will be compared with current breakdown strategies from the research of Kahan et al. which are the usage of manual repair stations only and the workload reallocation of broken robots by working robots downstream the flow line [50]. Graph 5.7 illustrates the mean cycle time and the standard deviation of the mean cycle time for the various backup stations options in the robotic U-shaped line backup strategy designed and the breakdown strategies from Kahan's et al. research.

Graph 5.7: Mean cycle time of the breakdown strategies considered

The backup station option D has the mean cycle time of 193 seconds with a standard deviation of 7.02 seconds, option AD has the mean cycle time of 126 seconds with a standard deviation of 6.72 seconds, option ACD has the mean cycle time of 94 seconds with a standard deviation of 1.51 seconds, option ABCD has the mean cycle time of 93 seconds with a standard deviation of 1.55 seconds, the breakdown strategy workload reallocation downstream the line has the mean cycle time of 130 seconds with a standard deviation of 8.02 seconds and the breakdown strategy usage of manual repair stations only has the mean cycle time of 227 seconds with a standard deviation of 6.48 seconds for the 5%-line breakdown scenario. Afterwards, the option D has the mean cycle time of 207 seconds with a standard deviation of 5.91 seconds, option AD has the mean cycle time of 171 seconds with a standard deviation of 6.32 seconds, option ACD has the mean cycle time of 106 seconds with a standard deviation of 3.24 seconds, option ABCD has the mean cycle time of 100 seconds with a standard deviation of 3.21

seconds, the breakdown strategy workload reallocation downstream the line has the mean cycle time of 173 seconds with a standard deviation of 11.37 seconds and the breakdown strategy usage of manual repair stations only has the mean cycle time of 250 seconds with a standard deviation of 5.68 seconds for the 10%-line breakdown scenario. For the last 15%-line breakdown scenario, the option D has the mean cycle time of 222 seconds with a standard deviation of 5.43 seconds, option AD has the mean cycle time of 175 seconds with a standard deviation of 4.09 seconds, option ACD has the mean cycle time of 122 seconds with a standard deviation of 5.40 seconds, option ABCD has the mean cycle time of 108 seconds with a standard deviation of 4.37 seconds, the breakdown strategy workload reallocation downstream the line has the mean cycle time of 230 seconds with a standard deviation of 11.25 seconds and the breakdown strategy usage of manual repair stations only has the mean cycle time of 263 seconds with a standard deviation of 5.17 seconds.

The error bars show that the options ACD, ABCD are similar to each other and they have a significant lower mean cycle time than the options D, AD, the breakdown strategies workload reallocation downstream and usage of manual repair stations only for the 5% and 10%-line breakdown. For the 15%-line breakdown scenario, the four backup station option ABCD has even a significant lower mean cycle time than the three backup station ACD. It should be pointed out that the robotic U-shaped line backup strategy with all four backup stations ABCD obviously has the best performance of all the backup station options and breakdown strategies considered. It has a 51.8% lower cycle time than option D, a 26.0% lower cycle time than option AD, a 0.6% lower cycle time than option ACD, a 28.2% lower cycle time than the breakdown strategy workload

reallocation downstream the line and a 59.0% lower cycle time than the breakdown strategy usage of manual repair stations only for the 5%-line breakdown scenario. Afterwards, the option ABCD has a 52.0% lower cycle time than option D, a 41.6% lower cycle time than option AD, a 5.7% lower cycle time than option ACD, a 42.4% lower cycle time than the breakdown strategy workload reallocation downstream the line and a 60.2% lower cycle time than the breakdown strategy usage of manual repair stations only for the 10%-line breakdown scenario. For the 15%-line breakdown scenario, the option ABCD has a 51.5% lower cycle time than option D, a 38.4% lower cycle time than option AD, a 11.9% lower cycle time than option ACD, a 53.1% lower cycle time than the breakdown strategy workload reallocation downstream the line and a 58.9% lower cycle time than the breakdown strategy usage of manual repair stations only.

The comparison of the four backup stations option ABCD with the other backup stations options showed that the three backup stations ACD has similar performance to the four backup stations option. This is due the fact that both options can cover each workstation and therefore manual repair stations are not needed. Furthermore, the backup stations options ACD and ABCD have a much better performance than the breakdown strategies workload reallocation downstream the line and usage of manual repair stations only. Thereby, the breakdown strategy usage of manual repair stations only has the highest cycle time for each breakdown scenario.

The breakdown strategy workload reallocation downstream the line has quite the same cycle time has the two backup stations option AD for the 5% and 10%-line breakdown scenarios. For the 15%-line breakdowns scenario, the cycle time of breakdown strategy

workload reallocation downstream the line increases to the same level as the one backup station option D. The reason for this is that the breakdown strategy needs manual repair stations as well. The workload of the last workstation cannot be reallocated downstream of the line, because there does not exist another workstation. Furthermore, the workload of a broken robots is allocated to the following workstation only. If the following robot breaks, the workload has to be executed by the manual repair station. Thus the breakdown strategy workload reallocation downstream the line performs on the same level as the one and two backup stations options. The following product quality analysis illustrates the number tasks which has to be executed on the manual repair stations.

Graph 5.8: Product quality analysis for the breakdown strategies comparison

Graph 5.8 shows the mean number of tasks and the standard deviation of the mean number of tasks which are executed on the manual repair stations for the backup stations options and breakdown strategies. As mentioned above, the backup stations options ACD and ABCD can cover each workstation and therefore the manual repair stations are not used. The breakdowns strategy usage of manual repair stations only has to use the manual repair each time when a robot fails. Thus it has the highest number of tasks on the manual repair stations for breakdowns which are the mean number of 3.5 tasks with a standard deviation of 0.15 tasks for the 5%-line breakdown scenario, 7.3 tasks with a standard deviation of 0.20 tasks for the 10%-line breakdown scenario and 10.6 tasks with a standard deviation of 0.22 tasks for the 15%-line breakdown scenario.

The error bars show that the breakdown strategy workload reallocation downstream the line preforms on the same level as the two backup stations option AD. It has the mean number of 0.3 tasks with a standard deviation of 0.12 tasks for the 5%-line breakdown scenario, 1.0 task with a standard deviation of 0.20 tasks for the 10%-line breakdown scenario and 2.3 tasks with a standard deviation of 0.22 tasks for the 15%-line breakdown scenario on the manual repair stations. As mentioned in Section 3.3, the breakdown strategy workload reallocation downstream the line has the best performance in the research of Kahan et al. [50].

The present results show that the workload reallocation downstream the line performs better than the breakdown strategy usage of manual repair stations only. Nevertheless, the backup stations options ACD and ABCD performs much better than the breakdown strategy workload reallocation downstream the line. The backup stations option ACD and ABCD offer a good product quality continuously, because they do not

have to use the manual repair station. Furthermore, both backup stations options have a cycle time increase to around 100 second for the three breakdown scenario considered. A cycle time of 100 seconds is near the initial cycle time of 91 seconds for a line flow without breakdowns which evidences the good performance of the options ACD and ABCD as backup strategies for the robotic U-shaped line.

In the previous chapter it is mentioned that the robotic U-shaped line requires a breakdown strategy which can avoid manual repair stations, has an efficient number of backup stations and can handle breakdowns with an adequate cycle time increase. The backup stations option ACD fulfills all requirements of the robotic U-shaped line backup strategy. It is able to cover each workstation and therefore manual repair stations are not needed. The cycle time of option ACD increases slightly compared to the initial cycle time for each breakdown scenario. Furthermore, the option ACD uses one workstation less than the option ABCD and both options have quite the same performance. Thus the three backup stations option ACD is the best choice for robotic U-shaped line backup strategy. Figure 5.1 illustrates the final design of the robotic U-shaped line backup strategy which uses three backup station on the places A, C and D.

Figure 5.1: Final robotic U-shaped line backup strategy design

The following analyses will use various work times for the manual repair stations to verify the efficiency of the robotic U-shaped line backup strategy option ACD.

5.5 Breakdown strategies comparison using a five-time greater rework time

This breakdown strategies comparison will use a five-time greater manual rework time for manual repair stations. A five times higher work time for manual repair stations should demonstrate a scenario where the tasks are more complicated. Thus operator require much more time to execute the tasks manually.

Graph 5.9: Mean cycle time of the breakdown strategies using a five-time greater manual rework time

Graph 5.9 illustrates the mean cycle time and the standard deviation of the mean cycle time for the various backup stations options in the robotic U-shaped line backup strategy designed and the breakdown strategies from Kahan's et al. research using a five-time greater manual rework time. The backup station option D has the mean cycle time of 278 seconds with a standard deviation of 13.63 seconds, option AD has the mean cycle time of 187 seconds with a standard deviation of 15.30 seconds, option ACD has the mean cycle time of 94 seconds with a standard deviation of 1.51 seconds, the breakdown strategy workload reallocation downstream the line has the mean cycle time of 181 seconds with a standard deviation of 21.54 seconds and the breakdown strategy usage of manual repair stations only has the mean cycle time of 347 seconds with a standard deviation of 11.09 seconds for the 5%-line breakdown scenario. Afterwards,

the option D has the mean cycle time of 319 seconds with a standard deviation of 12.03 seconds, option AD has the mean cycle time of 215 seconds with a standard deviation of 12.04 seconds, option ACD has the mean cycle time of 106 seconds with a standard deviation of 3.24 seconds, the breakdown strategy workload reallocation downstream the line has the mean cycle time of 230 seconds with a standard deviation of 19.56 seconds and the breakdown strategy usage of manual repair stations only has the mean cycle time of 384 seconds with a standard deviation of 8.89 seconds for the 10%-line breakdown scenario. For the last 15%-line breakdown scenario, the option D has the mean cycle time of 328 seconds with a standard deviation of 8.82 seconds, option AD has the mean cycle time of 252 seconds with a standard deviation of 7.97 seconds, option ACD has the mean cycle time of 122 seconds with a standard deviation of 5.40 seconds, the breakdown strategy workload reallocation downstream the line has the mean cycle time of 330 seconds with a standard deviation of 19.01 seconds and the breakdown strategy usage of manual repair stations only has the mean cycle time of 388 seconds with a standard deviation of 8.93 seconds.

The error bars show the robotic U-shaped line backup strategy option ACD has a significant lower cycle time than the backup stations options and breakdown strategies considered for the 5%, 10% and 15%-line breakdowns. It has a 66.3% lower cycle time than option D, a 49.8% lower cycle time than option AD, a 48.1% lower cycle time than the breakdown strategy workload reallocation downstream the line and a 73.0% lower cycle time than the breakdown strategy usage of manual repair stations only for the 5%-line breakdown scenario. Afterwards, the option ACD has a 66.9% lower cycle time than option D, a 50.8% lower cycle time than option AD, a 54.0% lower cycle time than

the breakdown strategy workload reallocation downstream the line and a 72.5% lower cycle time than the breakdown strategy usage of manual repair stations only for the 10%-line breakdown scenario. For the 15%-line breakdown scenario, the option ACD has a 62.6% lower cycle time than option D, a 51.5% lower cycle time than option AD, a 62.9% lower cycle time than the breakdown strategy workload reallocation downstream the line and a 68.5% lower cycle time than the breakdown strategy usage of manual repair stations only.

The comparison of the robotic U-shaped line backup strategy option ACD with the other breakdown strategies shows the efficiency of backup robots for tasks which are more complicated. The cycle time of the rU-sbs is nearly on the same level as the cycle time for a line flow without breakdowns. Conversely, the use of manual repair stations increases the cycle time dramatically. The breakdown strategy workload reallocation downstream the line has the same performance as the backup strategy option AD for the 5% and 10%-line breakdown. Their cycle times are twice higher than the cycle time of the option ACD for the 5%-line breakdown. The breakdown strategy workload reallocation downstream the line has even a three times higher cycle time than the option ACD for the 15%-line breakdown. The worst cycle time has the breakdown strategy usage of manual repair stations only. It has a three times higher cycle time than the backup strategy option ACD in each breakdown scenario. Thus it should be avoided to use manual repair station for complex tasks.

In the previous section a product quality analysis has been done for the breakdown strategies considered. It was shown that the backup strategy option offers the best product quality, because manual repair stations are not used. Nevertheless, a product

quality analysis will be presented in the following to demonstrate the number of tasks, which increase cycle time dramatically.

Graph 5.10: Product quality analysis for the breakdown strategies comparison using a five-time greater manual rework time

Graph 5.10 shows the mean number of tasks and the standard deviation for mean number of tasks with a five-time greater manual rework time on the manual repair stations for the backup stations options and breakdown strategies. As mentioned above, the backup stations option ACD covers each workstation and therefore the manual repair stations are not used. The breakdowns strategy usage of manual repair stations only has to use the manual repair each time when a robot fails. Thus it has the highest number of tasks on the manual repair stations for breakdowns which are the mean number of 3.5 with a standard deviation of 0.15 tasks for the 5%-line breakdown scenario, 7.3 tasks

with a standard deviation of 0.20 tasks for the 10%-line breakdown scenario and 10.5 tasks with a standard deviation of 0.25 tasks for the 15%-line breakdown scenario.

The error bars show that the breakdown strategy workload reallocation downstream the line preforms on the same level as the two backup stations option AD. It has the mean number of 0.6 tasks with a standard deviation of 0.18 tasks for the 5%-line breakdown scenario, 1.0 tasks with a standard deviation of 0.20 tasks for the 10%-line breakdown scenario and 2.5 tasks with a standard deviation of 0.30 tasks for the 15%-line breakdown scenario on the manual repair stations. Although, one task is executed only on the manual repair station for the 5% and 10%-line breakdown scenario, the cycle time increase is huge. This fact validates that manual repair station should be avoided for complex tasks. Hence, the robotic U-shaped line backup strategy is more efficient for complex tasks and should be used to ensure a smooth line flow and a good product quality continuously.

Here the question arises whether the use of the robotic U-shaped line backup strategy generates a benefit for standardized products. This question will be investigated in the following section.

5.6 Breakdown strategies comparison using a two-time greater rework time

This breakdown strategies comparison will use a two-time greater manual rework time for manual repair stations. A two-time greater manual rework for manual repair stations should demonstrate a scenario where the tasks are standardized. Thus operator require less time to execute the tasks manually.

Graph 5.11: Mean cycle time of the breakdown strategies using a two-time greater manual rework time

Graph 5.11 illustrates the mean cycle time and the standard deviation for the various backup stations options in the robotic U-shaped line backup strategy designed and the breakdown strategies from Kahan's et al. research using a two-time greater manual rework time. The backup station option D has the mean cycle time of 136 with a standard deviation of 4.76 seconds, option AD has the mean cycle time of 111 seconds with a standard deviation of 3.45 seconds, option ACD has the mean cycle time of 94 seconds with a standard deviation of 1.51 seconds, the breakdown strategy workload reallocation downstream the line has the mean cycle time of 134 seconds with a standard deviation of 5.80 seconds and the breakdown strategy usage of manual repair stations only has the mean cycle time of 181 seconds with a standard deviation of 5.15 seconds for the 5%-line breakdown scenario. Afterwards, the option D has the mean cycle time of 167 seconds with a standard deviation of 4.83 seconds, option AD has the mean cycle

time of 135 seconds with a standard deviation of 4.52 seconds, option ACD has the mean cycle time of 106 seconds with a standard deviation of 3.24 seconds, the breakdown strategy workload reallocation downstream the line has the mean cycle time of 166 seconds with a standard deviation of 7.31 seconds and the breakdown strategy usage of manual repair stations only has the mean cycle time of 191 seconds with a standard deviation of 4.10 seconds for the 10%-line breakdown scenario. For the last 15%-line breakdown scenario, the option D has the mean cycle time of 179 seconds with a standard deviation of 4.16 seconds, option AD has the mean cycle time of 156 seconds with a standard deviation of 3.76 seconds, option ACD has the mean cycle time of 122 seconds with a standard deviation of 5.40 seconds, the breakdown strategy workload reallocation downstream the line has the mean cycle time of 185 seconds with a standard deviation of 7.33 seconds and the breakdown strategy usage of manual repair stations only has the mean cycle time of 202 seconds with a standard deviation of 3.67 seconds.

The error bars show the robotic U-shaped line backup strategy option ACD has a significant lower mean cycle time than the backup stations options and breakdown strategies considered. It has a 31.1% lower cycle time than option D, a 15.3% lower cycle time than option AD, a 30.0% lower cycle time than the breakdown strategy workload reallocation downstream the line and a 48.3% lower cycle time than the breakdown strategy usage of manual repair stations only for the 5%-line breakdown scenario. Afterwards, the option ACD has a 36.7% lower cycle time than option D, a 21.8% lower cycle time than option AD, a 36.5% lower cycle time than the breakdown strategy workload reallocation downstream the line and a 44.7% lower cycle time than

the breakdown strategy usage of manual repair stations only for the 10%-line breakdown scenario. For the 15%-line breakdown scenario, the option ACD has a 31.5% lower cycle time than option D, a 21.7% lower cycle time than option AD, a 34.0% lower cycle time than the breakdown strategy workload reallocation downstream the line and a 39.4% lower cycle time than the breakdown strategy usage of manual repair stations only.

The comparison of the robotic U-shaped line backup strategy option ACD with the other breakdown strategies shows the efficiency of backup robots even for standardized tasks. Conversely, the use of manual repair stations performs poorly. Its cycle is in each breakdown scenario twice higher than the cycle time of robotic U-shaped line backup strategy option ACD.

The breakdown strategy workload reallocation downstream the line has a much better performance in this analysis than in the previous ones. Its cycle time is still much higher than the cycle time of the rU-sbs option ACD, but it is not twice higher as in the previous analyses. Nevertheless, the breakdown strategy workload reallocation downstream the line has only the same performance as the one backup station option D. Standardized products require a high volume production and therefore this breakdown strategy performs poorly for breakdowns, because it has a twice higher cycle time for breakdowns than the initial cycle time of 91seconds for a line flow without failures. Furthermore, it increases the workload of working robots for breakdowns. An increasing workload leads to a higher probability of breakdowns which is a major disadvantage of the breakdown strategy workload reallocation downstream the line. These facts verify

the good performance of the robotic U-shaped line backup strategy option ACD even for standardized products.

Standardized products do not require a high quality. Nevertheless, a product quality analysis will be presented in the following to demonstrate the generated product quality.

Graph 5.12: Product quality analysis for the breakdown strategies comparison using a two-time greater manual rework time

Graph 5.12 shows the mean number of tasks and the standard deviation of the mean number of tasks with a two-time greater manual rework time on the manual repair stations for the backup stations options and breakdown strategies. As mentioned above, the backup stations option ACD covers each workstation and therefore the manual repair stations are not used. The breakdowns strategy usage of manual repair stations only has to use the manual repair each time when a robot fails. Thus it has the highest number of

tasks on the manual repair stations for breakdowns which are the mean number of 3.7 tasks with a standard deviation of 0.16 tasks for the 5%-line breakdown scenario, 7.4 tasks with a standard deviation of 0.21 tasks for the 10%-line breakdown scenario and 10.5 tasks with a standard deviation of 0.23 tasks for the 15%-line breakdown scenario. Standardized products do not require a high quality, but the number of 7.4 and 10.5 tasks could be overcharged for the required product quality of standardized products.

The error bars show that the breakdown strategy workload reallocation downstream the line performs on the same level as the two backup stations option AD. It has the mean number of 0.5 tasks with a standard deviation of 0.15 tasks for the 5%-line breakdown scenario, 1.4 tasks with a standard deviation of 0.25 tasks for the 10%-line breakdown scenario and 2.6 tasks with a standard deviation of 0.34 tasks for the 15%-line breakdown scenario on the manual repair stations. The execution of one task offers the required product quality of standardized products. Although, the breakdown strategy workload reallocation downstream the line fulfills the product quality requirements, its cycle time is too high to ensure a high volume production for breakdowns. Thus the rU-sbs should be used for standardized products as well, because it ensures a smooth and high volume production continuously.

CHAPTER 6

CRITICAL VIEW ON THE RU-SBS & FURTHER RESEARCH

Chapter 5 shows that use of three backup stations on the places A, C, and D is the best option for the robotic U-shaped assembly line backup strategy, because it offers several advantages.

The first advantage is that three backup stations on the places A, C, and D ensures a low cycle time for breakdowns which is near the cycle of 91 seconds for a line flow without breakdowns. Furthermore, its cycle time is half as much as the cycle time of the current best breakdown strategy workload reallocation of broken robots by working robots downstream from the study of Kahan et al. which is investigated for an automotive body shop [50]. Automotive body shops target a high volume production and therefore a continuous low cycle time is essential for a smooth line flow and the performance of assembly lines.

Another advantage of the robotic U-shaped assembly line backup strategy using three backup stations on the places A, C, and D is that it offers a high product quality continuously. The current breakdown strategies of Kahan et al. requires manual repair stations to ensure that the line flow does not stop for breakdown [50]. On manual repair stations operators execute the tasks of the broken robots manually and the quality of tasks executed manually is worse than the quality of those executed by industrial robots, because industrial robots are programmed to execute tasks precisely [53]. Automotive

companies have to ensure a high product quality continuously, because failure in their products leads to repair cost, can harm their reputation, and can even lead to lives being lost [55]. Due to these facts it is important for automotive body shops to offer a high product quality continuously.

The last advantage of the robotic U-shaped line backup strategy is that it can eliminate waste in the form of inventories. The only breakdown strategy in the U-shaped line balancing problem research is the use of inventories between each workstation [49]. In Section 3.1, it is shown that work in process inventories belong to the waste in production, because they require a lot of space and bind money in each production line which can be used better in other departments. The elimination of work in process inventories makes the robotic U-shaped assembly line backup strategy the most efficient in the U-shaped line breakdown strategy research field. Nevertheless, the robotic U-shaped line backup strategy has several disadvantages.

The first disadvantage is the assumption that the backup robots cannot break down, because they are used only to cover broken robots and therefore they have a too short operating period to break down. This assumption is not realistic. The backup robots can break down even if they have short operating periods. This disadvantage may be repealed, if the backup robots are maintained after each operating period. The maintenance after each operation period ensure that the backup robots are not breaking down and thus they are able to be breakdown strategy which can be used in factories. However, the maintenance of the backup robots after each operating period could be a costly solution to ensure the efficiency of the robotic U-shaped line backup strategy.

The second disadvantage is that the rU-sbs requires an intelligent conveyor system. The backup stations are able to cover four workstations, because their placement allows to use them twice during the line flow on top and bottom part of the U-shaped assembly line layout. This dual entering of a backup station has to be realized intelligently by the conveyor system for the generation of a smooth line flow or otherwise it would be not possible to reach the good performance of the U-shaped line backup strategy. The algorithm in Section 4.3 considers the dual entering of backup stations by summing the work time of the first and second entrance. Thus the dual entrance of a backup station is a better breakdown strategy than the current breakdown strategies which is validated in the Chapter 5. Nevertheless, an intelligent conveyor system for the rU-sbs could be much more expensive than the regular ones for the current breakdown strategies and this has to be considered to find the best breakdown strategies for factories.

The last disadvantage is the requirement of several welding guns on the backup robots for the coverage of four workstations. Müller presented in his research that newer industrial robots are able to hold several welding guns and it needs only a few seconds to change a welding gun [9]. The breakdown strategy of the workload reallocation of broken robots by working robots downstream the line has the same disadvantage. The downstream robots need the same welding gun as their previous robot for the workload coverage. Even the breakdown strategy usage of the manual repair stations only needs additional welding gun to cover robot failures. Furthermore, if several robots break down, the manual repair stations will need several operators to cover the robots in an adequate cycle time increase. Each current breakdown strategy has the disadvantage of the requirement of additional welding guns. However, the rU-sbs uses backup robots

only for the coverage of broken robots and the breakdown strategies with manual repair station need several operators for breakdowns. Thus it has to be analyzed which breakdown strategy is the most cost efficient. Furthermore, the current production trend goes to individual consumer products. Individual products require a consumer demand based production volume. Thus, the optimization goal lies in the generation of a flexible production which has a consumer satisfying production time and quality.

The main goal of each manufacturing company is the maximization of profit. Cost and the right product demand belong to most important indicators. Therefore, a profit maximization research is needed to find out which breakdown strategy has the best performance for automotive companies. This research should consider a number of workstations and welding guns which are necessary to execute the tasks in an automotive body shop. Furthermore, other important cost factors have to be considered in the profit maximization which are human resources, equipment for the manual repair stations, the required conveyor system and a realistic customers demand to avoid over production. The generation of a profit optimization, which considers the cost factors of automotive body shop, could be done in cooperation with automotive body factory or an intensive cost research of the cost of automotive body shops has to be done. An optimization for a minimum cycle time was a good basis to start the backup strategy research for the line balancing problem, but the economical goal of companies is to reach an optimal profit. Hence, the line optimization with profit as objective function offers also a good research field as recommend by Hazir et al. for a further research field [22].

The disadvantages and the cost part are not the only limitations of this thesis. In Section 2.3, the constraints of the line balancing problem are presented. It is shown that factories have multiple assembly lines which interact with each other to manufacture products. Furthermore, it is common that more than one product is manufactured on one assembly line, because the product portfolio of factories increases to fulfill the individual customer needs [19, 20]. The constraints of multiple lines and several products on one assembly are not considered in this thesis as they increase the complexity of the line balancing problem and the consideration would be out of the scope of a master thesis. Nevertheless, the consideration of the multiple lines and several products constraints is essential for the implementation of the robotic U-shaped assembly line backup strategy in factories.

The last important limitation of this thesis is that the constraints considered as tasks time, production layout and capability matrix are adapted from the study of Kahan et al. which concentrates on automotive assembly lines only [50]. For a general evaluation of the performance of backup robots as breakdown strategy for U-shaped lines, further research is required which uses the task times, precedence and capability of products from various factories. Machining flow lines are widely disseminated throughout factories and they are automated to ensure a high volume production. Thus, it is essential to investigate how backup robots perform in an automated machining flow line and for this investigation is an intensive research in the machining production field required. The research in the machining production field should offer constraints which represent realistic data of machining flow lines. A U-shaped machining line breakdown

optimization using realistic data will ensure backup robots are the most efficient breakdown strategy for U-shaped lines.

Nevertheless, the used data in the robotic U-shaped assembly line backup strategy has an impact on further researches in the breakdown strategy research field. As mentioned in Section 3.3, the research field of breakdown strategies is limited. The current breakdown strategy of Kahan et al. is investigated in cooperation with the research and development department of General Motors which shows the interest of companies in the research of breakdown strategies [50]. As mentioned above, the robotic U-shaped assembly line backup strategy adapted the constraints from the study of Kahan et al. and the rU-sbs has a much better performance than the best breakdown strategy in Kahan's et al. study which is the workload reallocation of broken robots by working robots downstream the line [50]. Due to this fact backup robot has to be considered as a new breakdown strategy which has the potential to ensure smooth line flow and a continuously high product quality for breakdowns.

Furthermore, this thesis has shown that a genetic algorithm can be used in practice for the analysis of breakdown strategies. In Section 4.3, the genetic algorithm, which is used for the analyses of the rU-sbs, is described and it is shown that 16 hours are required for the investigation of one robotic U-shaped line backup strategy option. Sixteen hours is a long investigation period and therefore the modified is not an online solution which can offer the best breakdown strategy instantly. Nevertheless, the research for the best breakdown strategy for each flow line layout has to be done during the line designing period. The line designing period usually takes several month or even years when a new facility is build up and therefore 16 hours is not a long time period to

investigate the performance of a breakdown strategy. In addition, the genetic algorithm can be used as a solution approach for many different optimization problems which is shown in Section 2.5.2. Due to these facts, the genetic algorithm has the potential to be used in the practice as a solution approach for the investigation of the performance of breakdown strategies in the line layout designed.

CHAPTER 7

SUMMARY

This Master's thesis starts with a review of current configurations of manufacturing companies. A literature research follows to provide an explanation of the line balancing problem in Chapter 2. The explanation includes usual flow line layouts, constraints, optimization goals and solution methods which are used in the literature.

Chapter 3 has a brief description about U-shaped production lines and their advantages compared to straight production lines. Furthermore, Chapter 3 develops the connection to lean manufacturing. U-shaped lines are a part of Just-in-Time tools which should generate a smooth production with less production wastes. In the end of Chapter 3, breakdown strategies for the line balancing problem are presented and the inventory based backup strategy is the only breakdown strategy especially used for U-shaped lines. Work in process inventories are discordant with lean manufacturing principles of waste elimination. Other flow line breakdown strategies have been researched to find a breakdown strategy which fits the advantages of U-shaped lines and the lean manufacturing principles. Thereby, the backup stations breakdown strategy of Shirazi et al. offers the best potential as a lean breakdown strategy by using versatile backup robots for high capability workstations.

Chapter 4 explains the functionality of the chosen robotic U-shaped line design for which the backup strategy will be generated. The concept of backup stations with versatile robots was adapted to the robotic U-shaped line layout. This adaption is the placement of the backup stations between the arms of the U-shaped line layout. A specific configuration is selected for analysis and it is shown that four positions exist for the placement of backup stations for it. The goal of this thesis is to find the most efficient backup strategy for this robotic U-shaped line and therefore each possible number of workstations and placement positions have been analyzed. The criteria for the performance comparison are the cycle time and product quality which are generated for a 5%, 10% and 15%-line breakdown. For the generation of the criteria is the genetic of Shirazi used which is modified for the U-shaped line backup strategy and current breakdown strategies.

Chapter 5 analyzes the various placement positions for robotic U-shaped line backup strategy. It starts with the one backup station, proceeding to two backup stations and three backup stations options. At the end of Chapter 5, the best option of the one, two and three backup stations are compared with four backup stations and the current breakdown strategies which are the usage only manual repair stations of and the workload reallocation of broken robots by working robots downstream the line. It is shown that the three backup stations option on the placements A, C and D has the most efficient performance and it performs much better than the current breakdown strategies. The three backup stations option on the placements A, C and D has the cycle time increase from 94 seconds for the 5%-line breakdown to 122 seconds for the 15%-line breakdown scenario which is a good performance, because the initial cycle time for a

line flow without breakdowns is 91 seconds. Due to this fact this backup strategy ensures a smooth and high volume production in each breakdown scenario. Furthermore, the robotic U-shaped line backup strategy option ACD does not use manual repair stations and therefore it offers a good product quality continuously. On the contrary, the current breakdown strategies perform much worse than the robotic U-shaped line backup strategy option ACD. They have at least a twice higher cycle time than the rU-sbs option ACD for breakdowns and they have to execute several tasks on manual repair stations which decrease the product quality.

Chapter 6 reviews critically the robotic U-shaped line backup strategy designed and the disadvantages of the rU-sbs are shown. These disadvantages are the maintenance of the backup robots after each operation period, the requirement of an intelligent conveyor system and additional welding guns for the backup robots. Thus the robotic U-shaped line backup strategy could be a more cost-intensive option than the current breakdown strategies. However, the recommendation is given that a profit optimization is necessary to decide whether the robotic U-shaped line backup strategy offers the best practical performance of the current breakdown strategies.

APPENDICES

Appendix A: Kahan et al. approach [50]

Sets and parameters:

I = set of total group of spots

I^W = set of total groups of spots assigned to working robots $I^W \subseteq I$

I^F = set of total groups of spots assigned to failed robots $I^F = I / I^W$

R = set of all the robots located in the assembly line

R^W = set of working robots $R^W \subseteq R$

R^F = set of failed robots $R^F = R / R^W$

R_{Max} = maximum number of backup robots

t_i = performance time of group of spots I (including setup)

$I P_i^W$ = set of immediate predecessors of group i assigned to working robots

$I S_i^W$ = set of immediate successors of group i assigned to working robots

$I P_i^F$ = set of immediate predecessors of group i assigned to failed robots

$I S_i^F$ = set of immediate successors of group i assigned to failed robots

IM_{ir} = Initial Matrix

RCM_{ir} = Robot Capability Matrix

$$IM_{ir} = \begin{cases} 1, & \text{if group of spots } i \text{ is performed by robot } r \text{ in the initial state} \\ 0, & \text{otherwise.} \end{cases}$$

$$RCM_{ir} = \begin{cases} 1, & \text{if group of spots } i \text{ can be performed by robot } r \\ 0, & \text{otherwise.} \end{cases}$$

C = Cycle time

rm_{ir} = Recovery Matrix

$$rm_{ir} = \begin{cases} 1, & \text{if group of spots } i \text{ is performed by robot } r \text{ in the initial state} \\ 0, & \text{otherwise.} \end{cases}$$

$$z_r = \begin{cases} 1, & \text{if robot } r \text{ is a backup robot} \\ 0, & \text{otherwise.} \end{cases}$$

linear programming equations:

$$\text{Minimize } C \tag{A.1}$$

Subject to:

$$\sum_{i \in I^W} t_i * IM_{ir} + \sum_{i \in I^F} t_i * rm_{ir} \leq C \quad \forall r \in R^W \tag{A.2}$$

$$\sum_{r \in R^W} rm_{ir} = 1 \quad \forall i \in I^F \tag{A.3}$$

$$z_r \geq rm_{ir} \quad \forall i \in I^F, \forall r \in R^W \tag{A.4}$$

$$rm_{ir} \geq RCM_{ir} \quad \forall i \in I^F, \forall r \in R^W \tag{A.5}$$

$$\sum_{k \in R^W} k * IM_{hk} \leq \sum_{l \in R^W} l * rm_{il} \quad \forall i \in I^F, \forall h \in IP_i^W \quad A.6$$

$$\sum_{k \in R^W} k * rm_{ik} \leq \sum_{l \in R^W} l * IM_{gl} \quad \forall i \in I^F, \forall g \in IS_i^W \quad A.7$$

$$\sum_{k \in R^W} k * rm_{hk} \leq \sum_{l \in R^W} l * rm_{il} \quad \forall i \in I^F, \forall h \in IP_i^F \quad A.8$$

$$\sum_{k \in R^W} k * rm_{ik} \leq \sum_{l \in R^W} l * rm_{hl} \quad \forall i \in I^F, \forall h \in IS_i^F \quad A.9$$

$$\sum_{r \in R^W} z_r \leq R_{Max} \quad A.10$$

$$rm_{ir} \in 0,1 \quad \forall i \in I^F, \forall r \in R^W \quad A.11$$

$$z_r \in 0,1 \quad \forall r \in R^W \quad A.12$$

$$C \geq 0 \quad A.13$$

Appendix B: Müllers redundancy level approach [9]

Sets and parameters:

I = set of all groups of spots ($i = 1, 2, \dots, I$)

$I_{R,L}$ = set of groups that have to be done in a rear position on the left side

$I_{R,R}$ = set of groups that have to be done in a rear position on the right side

$I_{F,L}$ = set of groups that have to be done in a front position on the left side

$I_{F,R}$ = set of groups that have to be done in a front position on the right side

$I_{R,E}$ = set of groups that have to be done in a rear position on either side

$I_{F,E}$ = set of groups that have to be done in a front position on either side

$I_{E,L}$ = set of groups that have to be done on the left side in either position

$I_{E,R}$ = set of groups that have to be done on the right side in either position

J = set of welding guns ($j = 1, 2, \dots, J$)

K = set of stations ($k = 1, 2, \dots, K$)

R = set of robots ($r = 1, 2, \dots, R$)

$R_{k,r}$ = the r^{th} robot in the k^{th} station

IP_i = set of immediate predecessors of task i

t_i = Process time of group of spots i

$C_{\min}$ = Minimum cycle time

M = Very large positive number

WCM_{ij} = Welding gun Capability Matrix

C = cycle time of the system

M = A very large positive integer number

$x_{ikr} = \begin{cases} 1, & \text{if group of spots } i \text{ is assigned to robot } r \text{ in station } k \\ 0, & \text{otherwise.} \end{cases}$

$y_{jkr} = \begin{cases} 1, & \text{if welding gun } j \text{ is assigned to robot } r \text{ in station } k \\ 0, & \text{otherwise.} \end{cases}$

RCM_{ikr} = Robot Capability Matrix

$RCM_{ikr} = \begin{cases} 1, & \text{if group of spots } i \text{ can be performed by robot } r \text{ in station } k \\ 0, & \text{otherwise.} \end{cases}$

linear programming equations:

first part:

Minimize C B.1

Subject to:

$$\sum_{k=1}^K \sum_{r=1}^R x_{ikr} = 1 \quad \forall i \in I \quad \text{B.2}$$

$$\sum_{j=1}^J y_{jkr} = 1 \quad \forall k \in K, r \in R \quad \text{B.3}$$

$$x_{ikr} \leq \sum_{j=1}^J y_{jkr} * WCM_{ij} \quad \forall i \in I, k \in K, r \in R \quad \text{B.4}$$

$$\sum_{l=1}^L \sum_{r=1}^R x_{hlr} * h \leq \sum_{k=1}^K \sum_{r=1}^R x_{ikr} * k \quad \forall i \in I, h \in IP(i) \quad \text{B.5}$$

$$\sum_{k=1}^K \sum_{r=1}^R x_{ikr} * t_i \leq C \quad \forall i \in I \quad \text{B.6}$$

$$\sum_{k=1}^K \sum_{r=1}^R x_{ikr} * r \bmod 2 = 1 \quad \forall i \in I_{E,L} \quad \text{B.7}$$

$$\sum_{k=1}^K \sum_{r=1}^R x_{ikr} * r \bmod 2 = 0 \quad \forall i \in I_{E,R} \quad \text{B.8}$$

$$\sum_{k=1}^K \sum_{r=1}^R x_{ikr} * r \leq 2 \quad \forall i \in I_{R,E} \quad \text{B.9}$$

$$\sum_{k=1}^K \sum_{r=1}^R x_{ikr} * r \geq 3 \quad \forall i \in I_{F,E} \quad \text{B.10}$$

$$\sum_{k=1}^K \sum_{r=1}^R x_{ikr} * r = 1 \quad \forall i \in I_{R,L} \quad \text{B.11}$$

$$\sum_{k=1}^K \sum_{r=1}^R x_{ikr} * r = 2 \quad \forall i \in I_{R,R} \quad \text{B.12}$$

$$\sum_{k=1}^K \sum_{r=1}^R x_{ikr} * r = 3 \quad \forall i \in I_{F,L} \quad \text{B.13}$$

$$\sum_{k=1}^K \sum_{r=1}^R x_{ikr} * r = 4 \quad \forall i \in I_{F,R} \quad \text{B.14}$$

$$x_{ikr}, y_{ikr} \in \{0, 1\} \quad \forall i \in I, j \in J, k \in K, r \in R \quad \text{B.15}$$

$$C \geq 0 \quad \text{B.16}$$

Second part:

$$\text{Maximize } \sum_{i=1}^I \sum_{k=1}^K \sum_{r=1}^R RCM_{ikr} \quad \text{B.17}$$

$$\sum_{i=1}^I x_{ikr} * t_i \leq C_{min} \quad \forall k \in K, r \in R \quad \text{B.18}$$

$$RCM_{ikr} \leq \sum_{j=1}^J WCM_{ij} * y_{jkr} \quad \forall i \in I, k \in K, r \in R \quad \text{B.19}$$

$$RCM_{ikr} \leq \sum_{j=1}^J y_{jkr} * r \bmod 2 \quad \forall i \in I_{R,L} \cup I_{F,L} \cup I_{E,L}, k \in K, r \in R \quad \text{B.20}$$

$$RCM_{ikr} \leq 1 - \left(\sum_{j=1}^J y_{jkr} * r \bmod 2 \right) \quad \forall i \in I_{R,R} \cup I_{F,R} \cup I_{E,R}, k \in K, r \in R \quad \text{B.21}$$

$$\sum_{j=1}^J y_{jkr} * r \leq 2 + (1 - RCM_{ikr}) * M \quad \forall i \in I_{R,R} \cup I_{R,L} \cup I_{R,E}, k \in K, r \in R \quad \text{B.22}$$

$$\sum_{j=1}^J y_{jkr} * r \geq 3 - (1 - RCM_{ikr}) * M \quad \forall i \in I_{F,R} \cup I_{F,L} \cup I_{F,E}, k \in K, r \in R \quad \text{B.23}$$

$$\sum_{k=1}^K \sum_{r=1}^R x_{ikr} * k \leq \left(\sum_{j=1}^J y_{jlr} * l \right) + 1 - (RCM_{ilt}) * M \quad \forall i \in I, l \in K, t \in R \quad \text{B.24}$$

$$x_{ikr}, y_{ikr}, RCM_{ikr} \in \{0, 1\} \quad \forall i \in I, j \in J, k \in K, r \in R \quad \text{B.25}$$

Appendix C: Modified Genetic Algorithm

See the supplementary file “Modified Genetic Algorithm”

BIBLIOGRAPHY

- [1] MHP – A Porsche Company, "Die Studie „Industrie 4.0 – Eine Standortbestimmung der Automobil-," ESB Business School, Reutlingen University, Reutlingen, 2014.
- [2] F. O. Overheil, Layout Design in Automated Environments, Kingston: University of Rhode Island, 2009.
- [3] National Science Foundation, "Manufacturing the form of things unknown," [Online]. Available: <https://www.nsf.gov/about/history/nsf0050/pdf/manufacturing.pdf>. [Accessed 26 01 2016].
- [4] A. M. A. E. H. Youssef, "Assessment of manufacturing systems reconfiguration smoothness," *The International Journal of Advanced Manufacturing Technology*, vol. 30, pp. 174-193, 2006.
- [5] O. Battai and A. Dolgui, "A taxonomy of line balancing problems and their solution approaches," *International Journal of Production Economics*, vol. 143, p. 259–277, 2013.
- [6] G. . J. Miltenburg and J. Wijngaard, "The U-Line Line Balancing Problem," *Institute for Operations Research and the Management Sciences*, vol. 40, no. 10, pp. 1378-1388, 1994.

- [7] Y. Kara, C. Özgüven, N. Yalcin and Y. Atasagun, "Balancing straight and U-shaped assembly lines with resource dependent task times," *International Journal of Production Research*, vol. 49, no. 21, p. 6387–6405, 2011.
- [8] J. M. Nilakantan and S. G. Ponnambalam, "Robotic U-shaped assembly line balancing using particle swarm optimization," *Engineering Optimization*, vol. 48, no. 2, pp. 231-252, 2016.
- [9] C. Müller, *Fault-Tolerant Flow-Line Design: An example from an automotive body shop*, Kingston: University of Rhode Island, 2013.
- [10] M. Chica, J. Bautista, Ó. Cordon and S. Damas, "A multiobjective model and evolutionary algorithms for robust time and space assembly line balancing under uncertain demand," *Omega The International Journal of Management Science*, vol. 58, p. 55–68, 2015.
- [11] H. Gökcen, Y. Kara and Y. Atasagun, "Integrated line balancing to attain Shojinka in a multiple straight line facility," *International Journal of Computer Integrated Manufacturing*, vol. 23, no. 5, pp. 402-411, 2010.
- [12] O. Battaia, A. Dolgui, N. Guschinsky and G. Levin, "A decision support system for design of mass production machining lines composed of stations with rotary or mobile table," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 6, p. 672–680, 2012.

- [13] R. K. Hwang and H. Katayama, "A multi-decision genetic approach for workload balancing of," *International Journal of Production Research*, vol. 47, no. 14, p. 3797–3822, 2009.
- [14] M. S. Sodhi and B. R. Sarker, "Configuring flexible flowlines," *International Journal of Production Research*, vol. 41, no. 8, p. 1689–1706, 2003.
- [15] P. Sivasankaran and P. Shahabudeen, "Literature review of assembly line balancing problems," *The International Journal of Advanced Manufacturing Technology*, vol. 73, p. 1665–1694, 2014.
- [16] J. Oesterle and L. Amodeo, "Efficient Multi-Objective Optimization Method for the Mixed-Model-Line Assembly Line Design Problem," *Procedia CIRP*, vol. 47, p. 82 – 87, 2014.
- [17] R. Ramezani and A. Ezzatpanah, "Modeling and solving multi-objective mixed-model assembly line balancing and worker assignment problem," *Computers & Industrial Engineering*, vol. 87, p. 74–80, 2015.
- [18] Ö. Hazır and A. Dolgui, "Assembly line balancing under uncertainty: Robust optimization models and exact solution method," *Computers & Industrial Engineering*, vol. 65, pp. 261 - 267, 2013.
- [19] A. Azzi, D. Battini, M. Faccio and A. Persona, "Sequencing procedure for balancing the workloads variations in case of mixed model assembly system

with multiple secondary feeder lines," *International Journal of Production Research*, vol. 50, p. 6081–6098, 2012.

- [20] M. Rabbani, M. Moghaddam and N. Manavizadeh, "Balancing of mixed-model two-sided assembly lines with multiple U-shaped layout," *The International Journal of Advanced Manufacturing Technology*, vol. 59, no. 9, pp. 1191-1210, 2012.
- [21] M. E. Salveson, "The assembly line balancing problem," *The Journal of Industrial Engineering*, vol. 6, no. 3, p. 18–25, 1955.
- [22] Ö. Hazir, X. Delorme and A. Dolgui, "A Survey on Cost and Profit Oriented Assembly Line Balancing," in *Preprints of the 19th World Congress The International Federation of Automatic Control*, Cape Town, 2014.
- [23] W. Xu and T. Xiao, "Robust balancing of mixed model assembly line," *The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, vol. 28, pp. 1489-1502, 2009.
- [24] T. Genichi, *Introduction to Quality Engineering: Designing Quality Into Products and Processes*, Michigan: American Supplier Institute, 1986.
- [25] H.-G. Beyer and B. Sendhoff, "Robust optimization – A comprehensive survey," *Computer Methods in applied mechanics and engineering*, vol. 196, pp. 3190 - 3218, 2007.

- [26] Ö. Hazır, X. Delorme and A. Dolgui, "A review of cost and profit oriented line design and balancing problems and solution approaches," *Annual Reviews in Control*, vol. 40, pp. 14-24, 2015.
- [27] R. Esmailbeigi, B. Naderi and P. Charkhgard, "New formulations for the setup assembly line balancing and scheduling problem," *Operations Research & Decision Theory*, pp. 1 - 26, 2016.
- [28] B. Rekiek, A. Dolgui, A. Delchambre and A. Bratcu, "STATE OF ART OF OPTIMIZATION METHODS FOR ASSEMBLY LINE DESIGN," *Annual Reviews in Control*, vol. 26, pp. 163-174, 2002.
- [29] R. Pastor and L. Ferrer, "An improved mathematical program to solve the simple assembly line balancing problem," *International Journal of Production Research*, vol. 47, no. 11, p. 2943–2959, 2009.
- [30] N. Hamta, S. F. Ghomi, F. Jolai and U. Bahalke, "Bi-criteria assembly line balancing by considering flexible operation times," *Applied Mathematical Modelling*, vol. 35, p. 5592–5608, 2011.
- [31] J. Clausen, "Branch and Bound Algorithms - Principles and Examples," 12 03 1999. [Online]. Available:
<http://www.imada.sdu.dk/Employees/jbj/heuristikker/TSPtext.pdf>. [Accessed 09 03 2016].

- [32] E. C. Sewell and S. H. Jacobson, "A Branch, Bound, and Remember Algorithm for the Simple Assembly Line Balancing Problem," *INFORMS Journal on Computing*, vol. 24, no. 3, p. 33–442, 2012.
- [33] J. Bautista and J. Pereira, "A dynamic programming based heuristic for the assembly line balancing problem," *European Journal of Operational Research*, vol. 194, p. 787–794, 2009.
- [34] O. Guschinskaya and A. Dolgui, "Comparison of exact and heuristic methods for a transfer line balancing problem," *International Journal of Production Economics*, vol. 120, pp. 276-286, 2009.
- [35] O. Battaia and A. Dolgui, "Reduction approaches for a generalized line balancing problem," *Computers & Operations Research*, vol. 39, p. 2337–2345, 2012.
- [36] A. Otto and A. Scholl, "Incorporating ergonomic risks into assembly line balancing," *European Journal of Operational Research*, vol. 212, p. 277–286, 2011.
- [37] W. Zhang, W. Xu and M. Gen, "Hybrid Multiobjective Evolutionary Algorithm for Assembly Line Balancing Problem with Stochastic Processing Time," *Procedia Computer Science*, vol. 36, p. 587 – 592, 2014.

- [38] I. Sabuncuoglu, E. Erel and A. Alp, "Ant colony optimization for the single model U-type assembly line balancing problem," *International Journal of Production Economics*, vol. 120, p. 287–300, 2009.
- [39] M. Mitchell, "Genetic Algorithms: An Overview," *Complexity*, vol. 1, no. 1, p. 31{39, 1995.
- [40] J. Hofstede, Beasley, Bull and Martin, "Optimization Techniques Genetic Algorithms And other approaches for similar applications," 2002. [Online]. Available: <http://www.slideshare.net/rifkismith/0101genetic-algorithm-48395879>. [Accessed 17 3 2016].
- [41] D. E. Goldberg, *Genetic Algorithm in Search, Optimization, and Machine Learning*, Alabama: Addison-Wesley Publishing Company, Inc., 1989.
- [42] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, Amsterdam, Bristol: Springer-Verlag, 2003.
- [43] J. Miltenburg, "U-shaped production lines: A review of theory and practice," *International Journal of Production Economics*, vol. 70, p. 201}214, 2001.
- [44] . S. B. Vardeman, "The Impact of Dr. Shigeo Shingo on Modern Manufacturing Practices," [Online]. Available: <http://www.public.iastate.edu/~vardeman/IE361/f02mini/bumblauskas.pdf>. [Accessed 22 03 2016].

- [45] S. Aguado, R. Alvarez and R. Domingo, "Model of efficient and sustainable improvements in a lean production system through processes of environmental innovation," *Journal of Cleaner Production*, vol. 47, p. 141–148, 2013.
- [46] J. R. X. Alvesa and J. M. Alvesa, "Production management model integrating the principles of lean manufacturing and sustainability supported by the cultural transformation of a company," *International Journal of Production Research*, vol. 53, no. 17, pp. 5320-5333, 2015.
- [47] J. Nicholas, *Lean production for competitive advantage*, New York: Taylor & Francis Group, 2011.
- [48] S. C. B. Brecht, *MODELING OPTIMAL TRADEOFFS FOR LEAN AND SUSTAINABLE MANUFACTURING*, Kingstown: University of Rhode Island, 2012.
- [49] J. Miltenburg, "The effect of breakdowns on U-shaped production lines," *International Journal of Production Research*, vol. 38, no. 2, p. 353± 364, 2010.
- [50] T. Kahan, Y. Bukchin, R. Menassa and I. Ben-Gal, "Backup strategy for robots' failures in an automotive assembly system," *INTERNATIONAL JOURNAL OF PRODUCTION ECONOMICS*, 2009.
- [51] F. Shin, B. Ram, A. Gupta, X. Yu and R. Menassa, "A DECISION TOOL FOR ASSEMBLY LINE BREAKDOWN ACTION," in *Winter Simulation Conference*, 2004.

- [52] E. Sancı, REBALANCING OF ASSEMBLY LINES, Temmuz: MIDDLE EAST TECHNICAL UNIVERSITY, 2015.
- [53] A. N. Shirazi, M. Steinhaus and M. Sodhi, Multi-Objective Genetic Algorithm Solution for Backup Station in Fault-Tolerant Flow Line design, Kingston: Springer-Verlag, 2011.
- [54] Dürr, "Factory Assembly Systems," 01 09 2006. [Online]. Available: http://www.durr.com/fileadmin/user_upload/fas/en/pdf/Catalogue/01_FAS_conveyor_e06.pdf. [Accessed 2016 05 07].
- [55] S. Yan, "CNN Money," CNNMoney, 09 04 2014. [Online]. Available: <http://money.cnn.com/2014/04/09/autos/toyota-recall/>. [Accessed 30 07 2016].
- [56] J. A. Tompkins, J. A. White, Y. A. Bozer and J. M. Tanchoco, Facility Planning, Danvers: John Wiley & Sons, Inc., 2010, p. 32.