Universität Siegen

Production scheduling in small and medium sized companies with additional setup operator constraints

Daniel Schnitzler
December 2016

Prologue

This work, which was accepted by the “Fakultät für Wirtschaftswissenschaften, Wirtschaftsinformatik und Wirtschaftsrecht der Universität Siegen” as my dissertation, was developed while I was general manager of the “Siegener Mittelstandsinstitut der Universität Siegen”. It is therefore closely related to the projects of the “Siegener Mittelstandsinstitut”.

I would like to express my utmost gratitude to Prof. Dr Marcus Schweitzer, the primary reviewer of this work, for providing me with many useful ideas. In particular, I am thankful to him for encouraging me to think outside the box.

I am also thankful to Prof. Dr Gabor Erdelyi, the second reviewer, who supported and encouraged me to complete the work.

Prof. Dr Petra Moog helped me with her knowledge and insights to identify and align the work to the needs and problems of small and medium-sized companies. I am very thankful to her.

I owe special thanks to Christine Dykes and Jonas Weber for helping me correct the style and orthography of this work. Furthermore, I am indebted to Maximilian Göbbels, who checked my code for errors.
This work would have never become a reality without the further support of many of the SMI staff members. Kai Kohlmann and I worked very close on the code for the program. He also ran the test instances. Miriam Hiepler helped a lot with the empirical study, while Karoline Schmidt helped with the empirical study and the interviews with the companies. Dr Liliana Grigoriu worked with me on establishing some models and corrected my language.

I am grateful to my colleagues and friends Monika Belke, Dr. Dominik Kress, Dr. Jenny Nossack, Dr. Alena Otto, Dr. Jan Wolf, Philipp Zeise at the Universität Siegen. Thank you for your support and valuable feedback.

Siegen, October 2016 Daniel Schnitzler

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem

This research was conducted in the region of “Südwestfalen”, Germany. It is the third-largest industrial region in Germany in terms of the number of people employed in the industrial sector. It is also home to many small and medium-sized enterprises (SMEs) [cf. Bähr and Steier, 2013]. On the whole, these SMEs have achieved great success, which is also evident from the number of world market leaders located in the region [cf. Frye *et al.*, 2015].
Although many of these companies have a focus strategy, with respect to Porter [1980, pp. 35–40], their product offerings have become diverse. At the same time, product complexity has increased [cf. Schuh, 2005, p.13]. Consequently, the supply chain has become more complex, meaning that the complexity of production planning and production scheduling has also intensified. By production scheduling, we mean the allocation and sequencing of tasks, e.g. production orders to resources such as machines. In many cases, at the SMEs we visited, production scheduling was done without any special software or system support, but manually with the help of Microsoft Excel. The people responsible for production planning spent much of their working hours creating the daily production plan. Many decisions regarding the production plan were not based on facts and data, but on the experience of the production planners, often rendering these plans unreproducible. Throughout the observations, it showed that:

– the capacity of the machines,

– the time needed to setup a machine to produce a job (setup time),

– the structures of dependencies between the materials, semi-finished and finished goods in production (multi-level bills of materials),

– the availability of persons or robots for setting up the jobs on the machines (setup operators),

are not considered in the making of a plan. Usually plans are repaired after the creation to fix the biggest deficiencies. However, in the production itself, disruptions occur due to unexpected latencies. In the best case, expensive external production capacity can be rented. In the worst case, stock-outs arise or jobs are delivered late. Apart from that, there are high intermediate stock levels.
Production is often run in SMEs as a job shop. That means there are several different machines such as a milling machine, a drilling machine, etc., on which products in different orders must be manufactured. The main research area for this type of production scheduling does not focus, however, on providing solutions for holistic real-world problems. It seems that most of the effort is put into providing computational complexity results for production scheduling problems. Thus, it is often assumed in the literature that a job can be set up if it is available and the corresponding machine is empty. Yet, this can only hold if the setup operator, the person, or robot which changes the attachments of the machine, is not a limiting factor. Therefore, the consideration of setup operators is, in many cases, mandatory.

1.2 Aim

The general aim of this work is to link methods from the area of operations research to real world problems in SME, as this has not been done yet, to the best of our knowledge. Therefore, we investigate if it is possible to apply standard algorithms from the area of operations research, which have been shown to provide good solutions in production planning to real-life cases from SMEs. As the problem structure differs and the size of input varies, we compare and analyse the performance of the algorithms to make a more general statement about further problem contexts.

1.3 Method

As we investigate real-world problems in a holistic approach, the tools used in this work differ from quantitative inquiry to qualitative investigation in the companies to formal approaches from the area of operations research.
With the help of an empirical study, critical fields of action for small and medium-sized companies in the area of production planning were worked out with the purpose of reasoning the approach. The study is restricted to a problem motivation. Thereafter, representing the main focus of the work, three SMEs were used to analyse and explore the production planning problem in detail. Each problem was solved in a two-step approach:

> For each of the three production planning problems, a mixed-integer programme (MIP) was set up together with people from the respective operations scheduling department. Owing to the complexity of the MIPs, standard solvers were only able to solve minor instances of the problems.

Then metaheuristic algorithms were used to solve the MIPs with the aim of creating sound solutions in short computer running time.

1.4 Structure

As the aim of the work is to link different research areas, the work needs also to provide a basic understanding of the underlying research areas. Therefore, Chapter 2 describes state-of-the-art approaches to production planning and their use in small and medium-sized companies.[1] Therefore, fundamentals of general production planning procedures are shown in Section 2.1. The basic characteristics of small and medium sized companies and their differences from large companies are described in Section 2.2. Section 2.3 brings together both parts and indicates, with the help of an empirical study, which was conducted during the research, the peculiarities of the production planning in the area of small and medium sized companies.
In Chapter 3, an overview of recent literature, on production planning problems which take setup operator availability into account is presented from an operations research perspective. This chapter provides fundamentals for the later chosen algorithms.
Chapter 4 shows the first real case. As in Chapters 5 and 6, the information was assessed through data gathering in the companies themselves. It focuses on scheduling of parallel dedicated machines subject to setup constraints, as seen at a German factory. Section 4.1 introduces the problem, while Section 4.2 presents the problem definition. A standard model formulation is shown in Subsection 4.2.1. Extensions to the model can be found in Subsection 4.2.2. The metaheuristic algorithms which are used for the solution of the problem are outlined in Section 4.3. Genetic algorithms (GAs) (Subsection 4.3.1) tabu searches (TSs) (Subsection 4.3.2) and a simulated annealing (SA) (Subsection 4.3.3), as well as a variable neighbourhood search (VNS), combining the methods, are shown, which can solve problems up to 100 machines and 1,000 tasks. Computational results are presented in Section 4.4. Different variants of the metaheuristics are tested on the one hand with the help of random instances (Subsection 4.4.1), whereby a lower bound is calculated to evaluate the results. On the other hand, we design a class of test instances from which the optimum solution is known so as to determine the optimality gap accurately (Subsection 4.4.2). In Subsection 4.4.3 results from tests with real company data are presented. A brief conclusion about the chapter is then given in Section 4.5.
Chapter 5 shows the second real case. It focuses on unrelated parallel machine scheduling subject to setup and assignment constraints, as seen at another German factory. Section 5.1 introduces the problem before the problem is defined and the corresponding MIP is shown and described in Section 5.2. The methods

[1]The goal of this chapter is not to provide a detailed analysis of the status of production planning systems nor shall it provide an overview of all SME characteristics. The chapter should provide a basic understanding of SMEs and production planning systems so that the reasons for and the focus of the main research from Chapters 4, 5, and 6 are clear.

used (GAs 5.3.1, TSs 5.3.2, and VNS) are shown in Section 5.3. Computational results are then presented in 5.4 whereby different variants of the metaheuristics are tested, on the one hand, with the help of random instances (5.4.1). On the other hand, we design a class of test instances from which the optimum solution is known (Subsection 5.4.2) to determine the optimality gap accurately. In Subsection 5.4.3 real data from the company is used to test the metaheuristics. A conclusion is given in Section 5.5.
Chapter 6 shows the third real case. It focuses on the assignment and scheduling of jobs in a job shop with parallel machines subject to setup operator constraints as seen at the third analyzed company. Section 6.1 introduces the problem. In Section 6.2. the problem definition is given. The corresponding MIP is shown in Subsection 6.2.1, while in Subsection 6.2.2. extensions to the MIP are introduced. The focus in Section 6.3 is laid upon metaheuristics. GAs (6.3.1) and TSs (Subsection 6.3.2) as well as a VNS (6.3.3) are developed. Different variants of the methods are then tested in Section 6.4 with the help of random instances (Subsection 6.4.1), instances from which the optimal solution is known (Subsection 6.4.2) and industry data from the company on which this research is based on (6.4.3). A conclusion for the chapter is given in Section 6.5. A final conclusion is given in Chapter 7.

Chapter 2

State-of-the-art approaches in production planning and their use in small and medium-sized companies

2.1 Fundamentals of production planning

Production planning or production management can be defined and explained in a number of ways. For our purposes, to explain production planning and control (PPS) for non-mass producers, which most SMEs are, the Aachener PPS model from Schuh and Gierth [2006] seems most appropriate as it provides a broad perspective, without losing important aspects, on the area of PPS. In general, it can be stated that other works such as [Kurbel, 2011] could also be used as an introduction to PPS, as different works provide more or less the same view on PPS.

Production management plans, controls, and steers all operational resources with the aim of producing or delivering the output/product to the highest quality and quantity at the set time and at the lowest cost [cf. Schuh and Schmidt, 2014, p. 1]. That means the purpose of the PPS is to ensure that the goal of production, for example, to supply the output in the right manner and at the right time, is ensured at various levels as described in the following.

It has to be noted, though, that PPS is a highly complex matter. Many processes and functions are interdependent—starting with the customer order to production scheduling to the replenishment of materials for production. These interdependencies are shown with data and decision correlation. Data correlation means that data for some tasks are dependent on the data of another task. Decision correlation means that a decision of a certain task influences the decision of another task. Moreover, the sheer size of the data is another complexity

issue. In many cases, there are five- or six-digit numbers for products, work plans, etc.[cf. Scheer, 1991]
As a result, the PPS is a hierarchical approach, meaning that distinct levels of planning and control are run in stages while the output of a higher level stage is an prescribed input for the lower level planning. While the planning complexity can be reduced through this approach [cf. Scheer, 1991], it might not be possible to find the optimal solution for the production.

The Aachener PPS model was first introduced in 1993 but is continuously being developed [cf. Schuh and Gierth, 2006, p. 5]. It offers a theoretical background for operational practice. The model uses a holistic approach [cf. Schuh and Gierth, 2006, p. 11].

The priority of the Aachener PPS model is the description of PPS systems from different perspectives to reach the goals:

– selection and introduction of PPS systems,

– reorganization of PPS systems,

– development of PPS concepts,

– development of PPS systems,

– alignment of PPS processes,

in the best possible manner [cf. Luczak and Eversheim, 2001] [cf. Schuh and Gierth, 2006, p. 12]. For our purposes, to provide a basic understanding of the different planning levels in the field of production planning, the reference view of the so-called tasks from the Aachener PPS system seems most appropriate.

The task reference view, which the following paragraphs are based on, is divided into network tasks, core tasks, cross-sectional tasks, and data administration.[cf. Schuh and Roesgen, 2006]

Network tasks deal with inter- and intra-organization networks at a strategic level. It is possible that some tasks have an equivalent in the core tasks but are less detailed due to their inter-organizational character. Within the network tasks, again, three areas are sub-divided [cf. Schuh *et al.*, 2011a, p. 32]:

> **Network configuration**: One part of the network configuration is *product programme planning.* This is the decision of which parts to produce (type, variants, quality, if applicable quantity) and the choice of the corresponding procurement and distribution channels. The other part of the network configuration is the *network dimensioning.* The question of which parts or services, as chosen in the product programme planning, should be provided by the company itself (make-or-buy decision) must be answered in the network dimensioning. Related to this is the evaluation and choice

Network tasks	Core tasks		Cross-selection tasks		
Network configuration	Production program planning		Order management	Inventory management	Controlling
Network sales planning	Production demand planning				
Network requirement planning	Production planning and control	Outside supply planning and control			
Data management					

Table 2.1: Aachener PPS Model based on Schuh and Gierth [2006, p. 21]

of network partners. Finally, in intra-organizational networks, location planning is also part of the network dimensioning. [cf. Schuh and Roesgen, 2006, pp. 32-33]

Network sales planning: *Collaborative demand planning*, forecasting, and *demand consolidation* between corporations are included in network sales planning, which focuses on the distribution of final products. The result of sales planning is an estimate of which and how many products will be ordered. There is an equivalent task called production programme planning in the core tasks. [cf. Schuh *et al.*, 2011a, pp. 34-35]

Network requirement planning: Based on the results of the network configuration and network sales planning, it is first checked roughly whether the estimated demand can be matched by the given capacity in the planning periods (*network capacity planning*). If the estimated demand can be matched, the approval for the network production programme can be given. Otherwise, alternatives (outsourcing for example) must be considered.
Primary production requirements are then, in the so-called *network demand allocation*, allocated to the different production locations in the network from which, at a later stage, the local production plan can be made.
In the *network demand planning*: secondary production requirements are derived from all network locations, and if possible combined. They are then split between products to be produced within the network (insourcing) and products to be produced externally (outsourcing). [cf. Schuh *et al.*, 2011a, pp. 35-38]

Network tasks	
Network configuration	Product program planning Network dimensioning
Network sales planning	Collaborative demand planning Demand consolidation
Network requirement planning	Network capacity planning Network demand allocation Network demand planning
Core tasks	
Production program planning	Sales plan Primary requirement planning Rough resource planning
Production demand planning	Gross secondary requirement determination Net secondary requirement determination Type of supply assignment Lead time scheduling Capacity requirement identification Capacity balancing
Production planning and control	Lot sizing Detailed scheduling Resource detailed scheduling Sequencing Availability check Production order release
Outside supply planning and control	Order calculation Procurement offer obtaining Supplier selection Order release
Cross-sectional tasks	
Order management	Offer processing Order processing Order coordination
Inventory management	Inventory planning Warehouse administration Inventory analysis Inventory control Batch administration
Controlling	Information preparation Measure deduction
Data management	Master data Dynamic data

Table 2.2: The task reference view based on Schuh and Roesgen [2006, p. 30]

Cross-sectional tasks are used for the optimization and integration of the PPS system. They allow the integration of different network tasks, core tasks or core and network tasks [cf. Schuh and Roesgen, 2006, pp. 58-59]:

Order management: To provide an overarching order procedure, the order management starts with the *offer processing*. Here, all relevant information for the order is determined and recorded in the system. Also considered is the broad calculation at what point in time the order would be ready for delivery. If the offer is accepted by the customer, technical details are determined, and necessary external parts are acquired in the so-called *order processing*. Moreover, a rough production scheduling, including upstream activities such as construction, is done in such a way that basic production dates are given. Finally, a rough check of the resources (materials, workers, etc.) and the production capacity is done. The *order coordination* starts with the acceptance of the order and goes until the shipment of the order. It controls all order-related activities such as the progress of production, capacity of the resources, availability of materials, etc. [cf. Kurbel, 2011, pp. 247-251][cf. Schuh and Roesgen, 2006, pp. 59-63]

Inventory management: The purpose of the inventory management is to allow the company to buffer stochastic fluctuations, such as for example fluctuations in customer demand [cf. Whybark and Williams, 1976]. *Inventory planning* is the task to select sufficient disposition strategies and adjust the prolonging parameters to those strategies (minimum stock level, maximum stock level, order quantity, etc.) so that costs are minimized for the given targets.[cf. Corsten and Gössinger, 2009, pp. 475–488] *Warehouse administration* deals with the locations of stock in a warehouse and also the allocation of storage areas such as stock entry and exit or customs area. *Inventory analysis* tries to maximize the efficiency of a warehouse by analysing factors like the stock turnover rate. *Inventory control* records all stock rotations. Apart from the physical registration stocks are also recorded on a value basis.
A batch is a group of materials or products that is produced or handled under the same circumstances (same time, materials, etc.). *Batch administration* manages the information about a batch and provides this information to other departments. [cf. Schuh *et al.*, 2011a, pp. 65–70]

Accounting: The role of financial governance of PPS systems is to control and to provide transparent and expressive information [cf. Much and Nicolai, 1995, p. 157] [cf. Eversheim, 1995, p. 197]. Thus, the focus, compared with order management, is not on individual orders but on the whole PPS system. *Information preparation* is needed for the decision-making process. Key figures or key performance indicators (KPIs) are used to illustrate the PPS processes in a less complex manner. A target actual comparison of KPIs allows the identification of recent or upcoming problems in the specific processes. If relationships between processes and

KPIs are known, *measures can be deducted.* This can either be done manually, or if possible automatically, as in the case of raising minimum stock levels when KPIs, regarding the ability to supply goods, reveal problems. [cf. Schuh and Roesgen, 2006, pp. 69-71]

Data management: The purpose of data management is to store and maintain all relevant data for the enterprise resource planning and/or the PPS system. *Master data* is used permanently in the planning process. Therefore, it needs persistence and high maintenance effort. Material-resource master data, bill of materials, work plans, as well as customer and supplier master data are distinguished. [cf. Kurbel, 2011, pp. 58–92] The persistence of *dynamic data* is more limited than master data. It is, on the one hand, always related to a point in time (i.e. the amount of stock at time t). On the other hand, it gets flagged with certain statuses such as 'finished' for a production order. Dynamic data is related to master data (i.e. which customer owns the order). Dynamic data includes data related to stock, production, and operations. [cf. Kurbel, 2011, pp. 124-127, 171-173, 210-211]

Core tasks involve the key jobs of the PPS system for a single company [cf. Schuh and Roesgen, 2006, pp. 52]:

Production programme planning: The result of the production program planning is which primary production requirements in which amount at which points in time shall be produced [cf. Hackstein, 1988, p. 295]. In order to do so, a *sales plan* must be created first. It is usually done by the sales department. If the company is part of a production network, data from the network sales planning can be used. Otherwise the sales are forecast with the help of statistical data or with the help of turnover targets.[2] Based on the sales forecast, the primary internal requirements, and the customer orders at hand, the gross primary requirement is compared to the actual stock levels. The results of this *primary requirement planning* are the net primary requirements. [cf. Lödding, 2015, p. 108] These requirements are then checked for validity regarding machine and worker capacity, operating inventories, and materials in the *rough resource planning.* If the primary requirements cannot be covered with the given resources, the plan must be changed (i.e. produce products later or add additional workers). [cf. Kurbel, 2011, pp. 144-152]

Production demand planning: The production demand planning‘s task is to plan the resources, outgoing from the results of the production programme planning, in a medium term (i.e. weekly) and provide sufficient production supplies. Thus, a *gross secondary requirement determination* is done. The primary, secondary, and tertiary requirements are

[2]For more information on forecasts and a literature overview, see Tempelmeier [2006, pp. 31–52].

identified without taking stock levels into consideration. Either deterministic approaches, in which production times and quantities are determined with the help of lead times and the bill of materials, or stochastic forecasting approaches are used. The identification of the open secondary requirements is done with a comparison between the amount of stock, reservations, orders, and requirements, either for specific points in time or time periods in the *net secondary requirement determination*. The make or buy decision for the net requirements is done in the *type of supply assignment* step. It is possible that the decision of which parts to procure is permanent. Otherwise is it decided on a case by case basis. Procurement and production orders are then combined periodically. In terms of *lead time scheduling*, the production and procurement orders are planned with the help of processing- and lead times between tasks on the resources or resource groups in a more detailed way than in the rough resource planning, but less detailed than in the production planning and control. Forward-, backward-, and midpoint-scheduling are differentiated. Either simultaneously with lead time scheduling or after it, the amount of necessary capacity for each resource or resource group is calculated for each period in the *capacity requirement identification*. Finally, a *capacity balancing* is done if necessary, by either adding additional resources in periods or by moving production orders to different periods. [cf. Schuh *et al.*, 2011a, pp. 65-70]

Internal production planning and control: Within production planning and control, the input from the production demand planning is used. The first step is *lot sizing*, which means in series production systems, the combination of orders for the same product type and in make-to-order production systems the combination of orders with similar usage of resources. Here, the amount of setup time needed is reduced and the average stock level increased. Lots can be created in many ways. In the *detailed scheduling* the start and end dates for each job on each resource are determined. This can either be made with backward, forward, midpoint, or bottleneck scheduling. It is possible, that the results show problems (i.e., the latest possible start date is in the past or the earliest possible finish time is too late). Thereafter, counter measures such as lot splitting are undertaken. The detailed scheduling does not take the available capacity into consideration. Therefore, the *resource detailed scheduling* compares the available capacity with the required capacity. If needed, jobs are rescheduled, or capacity is added through, for example, additional work shifts. An alternative to the sequential approach is the resource availability planning which simultaneously plans production dates and respects capacities. For production jobs, which are assigned to the same resource or resource group at the same time, a sequence must be established. The decision of the sequence can be made with the help of selection criteria such as priority rules, or with cumulative criteria such as the minimization of setup times. It is also possible, that the decision about the *job sequencing* is left open to the workers in front of the machines (decentralization). After all planning has been undertaken for each production job, the availability of the resources and material is once again checked *(availability check)*. If problems arise, the production plan must be reviewed in the detailed scheduling step. During control and steering, the *production orders are released*. If a production order is released, the preparation of the necessary resources begins and all relevant documentation for the production order is created at the same time. [cf. Herrmann and Manitz, 2015, pp. 14-21][cf. Nicolai *et al.*, 1999, pp. 43-51]

Outside supply planning and control: The number and types of products to be procured at which time are based on the type of supply assignment. After that, lot sizes for the procurement goods are calculated in the *order calculation*. Here, procurement requirements are combined for certain periods. If goods are to be procured for the first time, offers from suppliers must be obtained (*procurement offer obtaining*). After this step, these offers are evaluated in the *supplier selection*. Criteria can be quality, delivery date fidelity, price, and so on. Master contracts with the best suppliers for longer time periods are negotiated thereafter. The *order release* is then done finally regarding the results of the previously described planning steps. [cf. Schuh and Roesgen, 2006, pp. 56-58]

2.2 Differences of small and medium-sized companies in contrast to large companies

There is no single definition of SME. Instead, quantitative and qualitative characteristics are used to identify this type of companies.

2.2.1 Quantitative criteria

Curran and Blackburn [2001, pp.9-10] describe that quantitative SME definitions are useful and popular among researchers and policy-makers as they are simple, objective, and transparent. The quantitative definition of SME is used for all companies in each sector if it is within the limits of the threshold. Thus, these limits are the same for production, trading businesses, and so on. [cf. Clements *et al.*, 1997]
Apart from the limitations of the inferior covering of production problems this might create, there are many variations within the quantitative definitions. [cf. Berisha and Pula, 2015] Different indicators to define the size of the company are used. Popular indicators include capital, earnings, number of employees, turnover, sales volume, etc. [cf. Haake, 1987, p.15], [cf. Theile, 1996, p.16]
As we are only dealing with German companies in the later version of this work, two quantitative definitions which are widely known and used in Germany are shown exemplary.

The ‘Institut für Mittelstandsforschung Bonn’ (IfM) uses two characteristics to decide whether a company is considered as small or medium. The number of employees must be below 500 and the annual turnover below 50.000.000 €. If one of the two characteristics does not apply, the company would be characterized as large. [cf. IfM, 2002]

Enterprise category	Number employees	and	Annual turnover
Small	< 10		$<$ 1 million €
Medium-sized	< 500		$<$ 50 million €
SME	< 500		$<$ 50 million €

Table 2.3: SME-definition of the IfM Bonn based on IfM [2002]

The EU uses more restrictive characteristics. To be considered as a small or medium-sized company, two of the three categories must hold. 1. The number of employees must be below 250 persons. 2. The annual revenue must be below 50.000.000€. 3. The balance sheet total must be below 43.000.000 €. [cf. European-Commision, 2015, p.10]

Enterprise category	Number employees	and	Annual turnover	Annual balance sheet total
Micro	< 10		$<$ 2 million €	$<$ 2 million €
Small	< 50		$<$ 10 million €	$<$ 10 million €
Medium	< 250		$<$ 50 million €	$<$ 43 million €

Table 2.4: SME-definition of the European Commission based on European-Commision [2015, p.12]

However, it has to be acknowledged that, to be considered small or independent, financial interlocking must be taken into account. If more than 25% of the company in question is owned by another company, the values (number of employees, etc.) of the so-called non-independent company are added to the values of the owner company. The combined values are then evaluated.

2.2.2 Qualitative criteria

As stated in the last subsection, there may be some problems with any exclusive definition of an SME based on quantitative data. Leite and Ferreira [2011] note that it was shown that, while there may be inconsistencies in quantitative classifications, SMEs around the world share characteristics as the organizational form or strategic issues. Stokes and Wilson [2010, p. 5] add that it might be difficult to define an SME accurately, but they are recognized in the daily operations. In the following, distinguishing features of SMEs in contrast to large companies are presented. Generally speaking, Ackermann and Blumenstock [1993] state that there is a close tie between the owner and the SME in contrast to the owner or owners of large enterprises. Mugler [2006, pp. 25–27] also describe that the company is characterized by the personality of the owner, who is often also the manager.

Management
An older work from Bolton [1971] states that an SME is characterized through the management by its owner in a personalized manner. Other authors such as Theile [1996, pp. 16–17] talk also about a personal principle. The personal principle suggests that the manager or owner takes a leading role in the decision-making process. The company is for him a lifelong duty. Theile [1996, pp. 16–17] also speaks about the unity of leadership and capital. Hence, the manager is at the same time the owner, so he also bears some or all of the financial risk. He has a high personal risk with all company decisions, which is why he is likely to have an enormous influence on all strategic decisions in the company. Pfohl [2006] provides some more characteristics regarding the management of SMEs. The term 'owner-managed company' can often be used for SMEs in contrast to the 'externally managed companies', which are in many cases bigger enterprises. A paternalistic leadership structure is often a result of this, while in big companies the leadership follows, in general, principles and not the ideas of single individuals. Group decisions are thus uncommon in SMEs. Finally, it is through the owner-managed leadership structure, the management team is not easily interchangeable compared to external management in large companies.
Managers in SMEs often have a technical background but lack management knowledge. In large companies, on the other hand, the management team possess management knowledge, while the technical expertise lies with specific departments.
Also, information processing is not widely developed in SMEs. Related to the lack of information processing is the increased importance of improvisation and intuition, which also results from insufficient managerial planning. This might lead to the fact that, in contrast to large companies, limited options exist when wrong decisions are taken.
If the management of large companies can be said to be far away from the daily business, the management of SMEs is often directly involved in the daily business. As a result, there is a chance of accumulation of responsibility in the management level of SMEs, which might lead to an overload on the management. If there is a division of work at the management level, the division is often related to personal reasons and not related to the tasks themselves.

Organization
Theile [1996, pp. 16-17] relates the organization of an SME to the personal principle. The manager is in direct contact with his employees, customers, and suppliers. Therefore, the manager takes part in all technical, organizational, and administrative processes in the company. This is also described by Mank

[1991, p. 52]. Mugler [2006, pp. 25-27] describes the managers personal network. The manager is directly connected to suppliers, customers, and other key stakeholders. He also describes the organization of an SME as low formalized. Pfohl [2006] again gives some additional characteristics. The characteristics of the leadership at a large company as well as an SME also influence the organization. While in an SME the organization is built around the manager, with other people having little detailed knowledge about the system, the organization of large companies does not depend on persons but on tasks. SMEs often have low departmentalization. Moreover, it is typical for an SME to have low delegation and formalization levels.
The positive results are as follows: There is a short and direct information flow; delegations and control occur directly in personal contacts; there are rarely coordination problems; there are strong personal bonds between the team of managers and the employees; and finally there is a high level of flexibility which allows the company to adjust quickly to changing environmental circumstances. The quick adaption to environmental changes is also described by Mugler [2006, pp. 25–27].
On the negative side, as Pfohl [2006] notes, there is a low division of labour, which leads to a duplication of tasks.

Procurement
Pfohl [2006] states that SMEs usually have a weak position in the procurement market due to their low transaction volume. Furthermore, they often order materials as per customer demand. In contrast, large companies have a strong position in the procurement market. Material procurement is done independently of the production jobs but through long-term basic agreements with suppliers.

Production
Also in production, Pfohl [2006] says that the division of labour is less in SMEs than in large companies. In many cases, SMEs use universal machines, which are labour-intensive. Hence, the cost digression per unit with an increasing output is less than the cost digression in large companies with their special-purpose machines.

Sales
Regarding sales, Bolton [1971], Pfohl [2006], and Mugler [2006, pp. 25-27] describe one characteristic of an SME, which is to have only a small market share. Theile [1996, pp. 16–17] speaks about the personal principle which in sales means that the manager is in direct contact with the companys customers. Mugler [2006, pp. 25–27] describes that SMEs provide goods based on individual customer requests.

Logistics Pfohl [2006] provides some specific information about logistics for SMEs. SMEs rarely have a concept of logistics. Therefore, in many cases there is no dedicated logistics department. Furthermore, the companys logistics focus on daily activities in contrast to large companies, which focus on longer-term operational and strategic issues.

Finance
From Bolton [1971] and Mugler [2006, pp. 25–27] come the most important point concerning finance criteria for SMEs, which are similar to quantitative criteria. An SME must be independent. It is not part of a larger group or enterprise. Pfohl [2006] also describes that, due to their limited size, they rarely have access to the financial market and thus have limited financing options at their disposal. In a crisis, they get only limited governmental support. On the contrary, the ownership of large companies is dispersed. They also have access to the financial markets and receive governmental support in crisis situations.

Research and Development
Research and development (R&D) in SMEs is different from the formal R&D in large enterprises Vossen [1998]. Bessant [1999]; Vossen [1998]; Lee *et al.* [2010] as well as van de Vrande *et al.* [2009] state that, while SMEs are more flexible than large organizations and less formalized, they are limited in terms of internal R&D due to their financial resources. Acs and Audretsch [1987] describe them as important for innovation. According to Baum *et al.* [2000]; Ceci and Iubatti [2012]; Edwards *et al.* [2006], SMEs have an external focus on R&D, which is linked to social and personal ties. Baum *et al.* [2000]; Lee *et al.* [2010] describe that they get through these relationships or networks to have access to resources and new technological competences. Hence, Brunswicker and Vanhaverbeke [2015] conclude that SMEs prefer non-monetary activities to transaction-based ones. This type of R&D is known in the literature as open innovation. A general introduction to the topic is given by Chesbrough *et al.* [2006].
While Pfohl [2006] also sees no formal R&D departments in SME, he comes to different R&D characteristics in SME. SME research focuses on short-term, demand-oriented product and procedure developments with low interest in fundamental research. This also implies that there is a brief temporal gap between the invention and its economic use. By contrast, large companies often have dedicated R&D departments which work on long-term product and procedure developments with a strong connection to fundamental research. Therefore, they have a larger time gap between the invention and the economical use for the company.

Personal
As per definition shown, SMEs have fewer employees than their large counterparts. Pfohl [2006] also states that SMEs employ fewer people with high academic qualifications than large companies. Therefore, large companies have a tendency towards employee specialization, while SMEs tend to have people with a broad knowledge base. Mugler [2006, pp. 25–27] states that there is a close relationship between the manager and his employees.

In conclusion, it can be seen that it is not an easy task to differentiate SMEs from large companies. While it might be easy to rely solely on quantitative criteria to identify SMEs, in the case to provide insights into the production scheduling of SMEs this would be by no means accurate enough. Therefore, throughout this work, the word ‘SME refers to companies which show some or all of these characteristics.

2.3 Production planning in small and medium-sized companies

When we bring Sections 2.1. and 2.2. together, it is assumed that SMEs do not follow the described procedure from Section 2.1. Instead, we expect that in production planning SMEs follow the behaviour described by Pfohl [2006] and Mugler [2006, pp. 25–27].

In order to identify first fields of actions and to back up our assumptions for small and medium-sized companies in production planning, a quantitative assessment was conducted. The survey has been published beforehand [cf. Hiepler *et al.*, 2016]. The companies who were contacted for the survey are on the one hand SMEs from North-Rhine-Westphalia which run their own production, known to the University of Siegen. Besides, the van Dijk Electronic Publishing GmbH [2014] database was used to find more participants. Hereby, it was selected first that companies must be from North-Rhine-Westphalia. Then the companies were limited to the manufacturing sectors for plastics, glass, metal, electronics, machines, automotive, etc. Other sectors like bakeries, which are also listed as manufacturing companies, are excluded. As the number of employees and the total sum of the balance sheet were not listed, we separated the companies based on the annual turnover. Every company with an annual turnover of less than 50 million € was included. Moreover, companies from which no annual turnover was known and companies which had a higher annual turnover were checked through the internet. If the companies were managed by the family, it was also included in the questionnaire. More than 1,000 invitations were sent out to take part in the survey online.
The questions of the survey can be found in the appendix. Two existing questionnaires were used as the basis for the creation of the questions in this survey (FIR-RWTH-Aachen [2013]; Spath *et al.* [2010]). As the focus of these questionnaires was different, the results cannot be compared. The general structure and style of the questions from the questionnaires were used to create the questions for our study. They were then adapted especially to the needs for production planning in SME. As described, the goal of the questionnaire is to identify fields of actions for further research in production planning. To focus on the right sector and type of production as well as the goal for production planning in the case studies, these characteristics are first analysed. After that, deficits in production planning are directly asked for. As people may not be aware that they have

deficits in production planning, because they are, for example, buried through too high stock levels which can be used as a buffer for production, questions which check for these characteristics are included as well. Finally, questions regarding the maturity level of production planning are used to identify if the participants have made use of up to date processes and tools for production planning already, or if they have problems in production planning and did not adapt current available solutions. This would then be not a target for research but for consultants.
153 companies took part in the survey of which 116 were usable. Not each company answered every question though. The majority of the companies which took part in the survey are from the metal processing and machine engineering sector (Table 2.5). To validate the findings, the results are compared, wherever possible, to three other recent studies on production planning in SMEs (Heck and Vettiger [2014]; Bley *et al.* [2016]; Boß and Deckert [2017]). If no comparison is made, the information is not provided in the other studies. Bley *et al.* [2016] used a questionnaire and evaluated 244 responses, Boß and Deckert [2017] used a questionnaire and received 60 responses while Heck and Vettiger [2014] conducted interviews. The number of companies interviewed is not known. Heck and Vettiger [2014] describe the interviewed companies as from the industrial sector with own production from the area of Liechtenstein, Switzerland, and the southern part of Germany. Other business areas such as banking or gastronomy were not interviewed. From Bley *et al.* [2016] it is only known that most of the companies contacted were from the industrial sector around Dresden. No more details are given. Boß and Deckert [2017] have most responses from the area of metal processing and machine engineering sector (69%). As they did not include the automotive industry explicitly, and as it is thought that automotive industry is involved in metal processing, the distribution of the industry sectors seems comparable between their study and ours (69% from Boß and Deckert [2017] to 60% in our conducted survey).

It has to be acknowledged though, that the quantitative survey has only limited general significance, as the goal of the survey is, as stated, to identify first fields of actions for deeper research, an overview, and a basic understanding of specific production problems for SME. It is not meant to provide detailed analytical insights. Missing values are not dealt with explicitly. Reasons for missing values could be that some questions aim at internal data, which might not be available so far or which might not be wanted to be shared by the companies. Furthermore, it is possible that the questions were not understood correctly. To increase the validity of the survey and to use more advanced statistical tools, such as a multiple regression analysis, missing data could be handled by interpolation, for example. On the other hand there is no systematics in the missing data to recognize. Thus, no significant loss of meaningfulness is assumed. The introduction of failure explanatory variables exceeds the scope of the study [cf. Eid *et al.*, 2015, p. 49, p. 290]. Moreover, the reliability of the data indicates that only descriptive statements should be derived. Although the total number of participants is high, in some groups, the number of participants is so low

Industry	Frequency
Paper, publisher and print industry	5
Chemical industry	5
Rubber and plastic industry	6
Metal working and production	38
Manufacturing systems engineering	19
Production of office and data processing machines, electrical engineering and lenses	5
Vehicle manufacturing and automotive industry	7
Others	21
Total	106
Missing	10
Total	116

Table 2.5: Industry classification

that also non-parametric methods provide significant propositions only hardly. Finally, as only companies from North-Rhine-Westphalia were interviewed, it is not possible to generalize the answers taken, for all SMEs. Nevertheless, to identify fields for deeper research with the help of company data in the next chapters, the results are sufficient.

It is, furthermore, of interest to know what type of production the companies answering the questionnaire do, to tighten future research. Most of the SMEs investigated produce make-to-order (Section 2.2) (Table 2.6). The companies which we analysed for the case studies in this paper were thus chosen to be make-to-order producers. Only Boß and Deckert [2017] provide information about the type of order processing. Interestingly, they do not consider series production in their analysis. Nevertheless, most of their companies are also make-to-order producers or engineers to order (86%). This needs to be kept in mind when comparing the answers in the study and recommendations.

Regarding the 'logistical command variable', the 'requested delivery date' is mentioned most often (Table 2.7). This goal was also given in the case studies in Chapters 4, 5, and 6.
Due to formatting reasons, the targets are partly consolidated in (Table 2.7). In "other logistical command variable" are the answers "other logistical command variable", "minimize tardiness for production jobs", "minimize makespan" (minimize the time between the start of the first job and the end of the last job), "maximize output", "minimize inventory", "minimize operating cost", "minimize production cost", and "produce in lots" summarized.

Table 2.8 shows the companies satisfaction with their due date punctuality

	Make-to-order	Series production	Variant production	Mass production	Others	Total
Paper, publisher and print industry	3	1	1	0	0	5
Chemical industry	1	0	1	3	0	5
Rubber and plastic industry	3	1	1	1	0	6
Metal working and production	18	12	3	5	0	38
Manufacturing systems engineering	15	2	2	0	0	19
Production of office and data processing machines, electrical engineering and lenses	2	3	0	0	0	5
Vehicle manufacturing and automotive industry	1	3	1	1	0	6
Others	11	3	2	1	1	18
Total	54	25	11	11	1	102

Table 2.6: Production type

	Requested delivery date	No agreed command variable	Other logistical command variable	No statement	Total
Paper, publisher and print industry	3	1	1	0	5
Chemical industry	3	0	1	1	5
Rubber and plastic industry	3	1	2	0	6
Metal working and production	29	4	5	0	38
Manufacturing systems engineering	13	1	3	2	19
Production of office and data processing machines, electrical engineering and lenses	3	0	2	0	5
Vehicle manufacturing and automotive industry	4	0	2	0	6
Others	7	1	7	3	18
Total	65	8	23	6	102

Table 2.7: Logistical command variable and production type

	not at all satisfied	2	3	4	completely satisfied	Total
Paper, publisher and print industry	0	0	0	3	2	5
Chemical industry	0	0	1	3	0	4
Rubber and plastic industry	0	1	2	2	1	6
Metal working and production	1	3	14	13	5	36
Manufacturing systems engineering	1	2	6	6	0	15
Production of office and data processing machines, electrical engineering and lenses	0	1	0	3	1	5
Vehicle manufacturing and automotive industry	0	3	1	1	1	6
Others	1	1	3	9	4	18
Total	3	11	27	40	14	95

Table 2.8: Satisfaction concerning due dates

on a scale of 1–5, where 1 indicates that they are not satisfied at all and 5 indicates that they are completely satisfied. Fourteen companies are completely satisfied; 54 are in the range of 4–5; and 81 are in the range of 3–5.

Table 2.9 shows the inventory management activities of the companies. Nearly 2/3 of the companies use either warehousing or predominantly warehousing. Only little more than 1/3 focuses on just in time delivery or focuses on just-in-time delivery.

Table 2.10 shows the satisfaction of the companies with regard to their stock keeping levels on a scale of 1–5, where 1 means that the stock level is too high and 5 means the stock level is too low. Forty-five of the 95 companies indicated their satisfaction with a 3. This also means that 50 companies are not satisfied with their stock levels. Thirty-two of those 50 companies indicated that they think they have extremely high stock levels.
While one would first assume that make-to-order producers have no stock level, because they produce only when they get an order, there are different reasons for them to keep stock. On the one hand, this can be technically related to lot-sizing. When a lot is created from two different orders with different due dates, the order with the later delivery date will need to be put on stock. On the other hand, apart from the stock with finished products, it is common to put raw material or also semi-finished goods on stock. Then only the last production step or the customization is done when the customer order arrives. Although

	Make-to-order	Series production	Type production	Mass production	Others	Total
Warehousing	16	12	4	4	0	36
Just in time	6	0	0	0	0	6
Predominant warehousing	11	11	5	4	0	31
Predominant just in time	21	3	2	3	1	30
Total	54	26	11	11	1	103

Table 2.9: Inventory management

this question is not specifically asked in the studies from Boß and Deckert [2017] and Heck and Vettiger [2014] both give information about satisfaction with stock levels. Heck and Vettiger [2014] state that 67% of the interviewed companies ignore inventory costs. Therefore, it is clearly not in the focus of the companies. Boß and Deckert [2017] describe that only 20% of the companies have achieved an optimization with the help of PPS for stock. They therefore conclude that this is not in their focus. One reason might be that most of the companies in the questionnaire are make-to-order producers.

Table 2.11 shows the use of IT systems in the companies. Multiple answers could be given by a single company, as it is possible and might make sense to run multiple different IT systems. It should be noted that there is an overlap in the functionality of some systems. It is common for ERP systems also to provide production planning support: For example, it might be possible to create production lots in the ERP system. 57, 7% of the companies interviewed rely on an ERP system. 46, 4% use production data acquisition. 33% run a dedicated PPS system. Other IT-based tools are rarely used. Bley *et al.* [2016] describe that 48% of the companies answered that they use an ERP system and 33% use a system for production planning and control. Boß and Deckert [2017] quote nearly identical numbers. 47% of the companies use an ERP and 36% use a PPS system. However, it must be acknowledged that he offers more answer possibilities such as dedicated systems for logistical and business processes. Such a system can also be described as an ERP system for smaller companies. Then their percentage would go up to 67%. The general ranking of systems used in the studies is however the same. Different to this is the study from Heck and Vettiger [2014]. They quote that 83% of the companies interviewed use an ERP system. No other information is given.

The companies that claimed to be using IT systems were further studied about the type of resource constraints they considered in their planning (Table 2.12). Only little more than 50% of them include resource constraints at all in

	too high	2	3	4	too low	Total
Paper, publisher and print industry	0	1	4	0	0	5
Chemical industry	0	2	1	1	0	4
Rubber and plastic industry	0	2	3	1	0	6
Metal working and production	3	11	14	7	1	36
Manufacturing systems engineering	2	3	7	3	0	15
Production of office machines and data processing machines, electrical engineering and lenses	0	2	3	0	0	5
Vehicle manufacturing and automotive industry	2	1	2	1	0	6
Others	2	1	11	3	1	18
Total	9	23	45	16	2	95

Table 2.10: Satisfaction concerning stock levels

	Frequency	Percentage
Enterprise Resource Planning	56	57,7
Production planning and control system	32	33,0
Manufacturing detailed planning	9	9,3
Supply chain management system	7	7,2
Production data acquisition	45	46,4
Machine data acquisition	16	16,5
Product data management	8	8,2
Additional other IT-Systems	29	29,9
No IT-Systems	13	13,4
No Statement	6	6,2

Table 2.11: Use of IT systems

	Frequency	Percentage
Machine capacity	34	38,6
Human resource capacity	30	34,1
Setup capacity	5	5,7
Sequence-dependent setup times	9	10,2
Machine care	7	8,0
No statement	42	47,7

Table 2.12: Consideration of resource constraints in IT systems

the planning. Among the most considered factors are machine capacities 38, 6% and staff constraints 34, 1%. While this question is not asked specifically in the other studies, some information is given. Boß and Deckert [2017] describe that 50% of the companies answered they estimate the delivery date for an order. Two-thirds of the companies do not take process estimation structures such as methods and time management into consideration. They purely estimate durations based on their experience. Only a few companies use IT tools for planning. Over 70% of the companies decide about the production sequence together in meetings. As production planning is not an easy task, a low consideration of resource constraints in planning is considered. Heck and Vettiger [2014] describe a similar setting. 50% of the companies use pen and paper, or Excel. Again, the consideration of multiple resource constraints in such a planning scenario is difficult. Although the questions in the other studies had a different focus, the answers they provide are in line with the answers given in this study.

Only 35 of the interviewed companies use lot-sizing, which is the combination of orders for the same products in series production or the combination of orders with similar usage of resources in make-to-order production systems, in their production planning (Table 2.13). Nevertheless, we have to take the type of production into account. What is surprising, though, is that most of the make-to-order companies do not use lot-sizing. One explanation for this is that the characteristics of the products to be produced might be too different. Another explanation is provided by Schuh *et al.* [2011b, p. 150]. They indicate that some companies create lots by sequencing similar tasks from different production orders behind each other. As this is different from the common approach that combines the same production orders with a lot, some companies might not know that this can also be considered as lot-sizing and therefore indicated that they did not make use of lot-sizing. Even more astonishing is that some companies with series production indicated that they did not use lot-sizing. Therefore, it appears that the understanding of this question or the definition of lot-sizing might be subject to misunderstanding.

Of those companies which use lot-sizing, only about half of them do so by system support (Table 2.14). While the majority of SMEs in make-to-order production do not make use of lot-sizing (Table 2.13), half of them make use

	Yes	No	Total
Make-to-order	8	40	48
Series production	14	10	24
Variant production	5	6	11
Mass production	8	3	11
Others	0	1	1
Total	35	60	95

Table 2.13: Lot-size planning production type

	Yes	No	Total
Make-to-order	4	4	8
Series production	5	9	14
Variant production	4	1	5
Mass production	4	4	8
Total	17	18	35

Table 2.14: Lot-size planning production type system supported

of system support in lot-sizing. Furthermore, about half the variant producers use lot-sizing. When it comes to system support in lot-sizing, about all of them make use of it. Most of the series producers use lot-sizing but do not use system support. For series producers, an explanation could be that lot-sizing is an important task which can still be done manually. For variant producers, it is also important but cannot be done by hand easily. Therefore, if a variant producer wants to use lot-sizing, it has to do it with the support of IT.

We also investigated those companies that use some kind of an IT system to support the planning (Table 2.15). We believe that these companies should have a more advanced process orientation (at least they paid a decent amount of money to digitalize their processes or some of their processes), and therefore might indicate what others would do. To our surprise, they did not take many factors into account when it came to the size of a lot. Among the most important is machine capacity. However, only 13 of the 33 companies which answered that question took machine capacity into account. The next most mentioned resource constraint is staff capacity which, was only said to be included by seven of the 29 companies which answered that question.

The situation changes when we only investigate those companies that build their lots using an IT tool (Table 2.16). Ten of the 13 companies which answered the questions consider machine capacity explicitly. Only five of the companies under investigation answered that they considered the next most commonly mentioned resource (personnel). Hence, problems with the given capacity might be likely.

	Lot-size planning method		
	Yes	No	Total
Machine capacity	13	20	33
Human resource capacity	7	22	29
Setup capacity	2	2	4
Sequence-dependent setup times	4	5	9
Machine care	3	4	7
No Statement	15	26	41
Total	29	57	86

Table 2.15: Consideration of resources for the lot-size planning if IT systems are existent

	System supported lot-size planning		
	Yes	No	Total
Machine capacity	10	3	13
Human resource capacity	5	2	7
Setup capacity	2	0	2
Sequence-dependent setup times	3	1	4
Machine care	3	0	3
No Statement	5	10	15
Total	15	14	29

Table 2.16: Consideration of resources for the lot-size planning if IT systems are existent and lot-size planning is done system supported

	Frequency	Percentage
Imprecise determination of due dates / make-to-order	23	22,8
Unbalanced degree of capacity utilization	51	50,5
Imprecise sales planning	25	24,8
Insufficient treatment of completion confirmation data	22	21,8
Lacking tracking of progress towards completion	19	18,8
Logical break between planning and controlling	7	6,9
System break between planning and controlling	13	12,9
Static production planning insufficient consideration of external effects)	12	11,9
Planning with average and estimated values	28	27,7
Others	4	4,0
No Deficits	14	13,9

Table 2.17: Deficits

In Table 2.17 we asked the companies about their production deficits. The most frequently mentioned deficit is the uneven capacity utilization, with more than 50% of the SMEs claiming this. The use of average and estimated values in the planning is the second most mentioned deficit $(27,7\%)$. Answers can also be taken from Heck and Vettiger [2014]. They claim that 83% of the interviewed companies saw capacity utilization as a problem. 50% of the companies also complained about time-consuming setups as a problem for production planning. As most of the companies in the study from Heck and Vettiger [2014] use pen and paper, or Excel, these resource constraints are not considered explicitly, and they create a problem in production planning and later in production.

When considering the underlying type of production, the make-to-order and series producers (which we especially focus on, as previously stated) mostly cited uneven capacity utilization as a deficit (Table 2.18). In case of a make-to-order producer, this might be related to the uncertainty of incoming orders. For the series producers, which do their lot-sizing by hand, this manual procedure might be an explanation.

We finally asked the SMEs to suggest what improvements they might make as a solution to deficits (Table 2.19). Owing to illustration reasons, some answers are summarized again. We consolidated the answer possibility system break between planning and controlling, logical break between planning and controlling,

	Make-to-order	Series production	Type production	Mass production	Others	Total
Imprecise determination of due dates / make-to-order	12	6	4	1	0	23
Unbalanced degree of capacity utilization	27	13	4	7	0	51
Imprecise sales planning	7	7	6	5	0	25
Insufficient treatment of completion confirmation data	12	5	3	2	0	22
Lacking tracking of progress towards completion	11	0	4	4	0	19
Planning with average and estimated values	15	6	3	3	1	28
Others	13	10	5	4	1	36
No Deficits	8	3	2	1	0	14
Total	52	26	11	11	1	101

Table 2.18: Deficits production type

static production planning (insufficient consideration of external effects), etc. from the survey to others in the table. The central planning approach and the remonstrance of reaction strategies were suggested by most of them.

We now go on to summarize the most important findings with respect to further research that seeks to aid in SMEs production planning from Tables 2.5–2.19 and their respective counterparts in the other studies. From Table 2.6 it becomes clear that research should focus on make-to-order producers. This was also the answer given by Boß and Deckert [2017] who had a participation of 86% make-to-order producers. This is the largest group of production companies among SMEs. The most important logistical command variable is the requested delivery date (Table 2.9). To our surprise, there are more companies closer to being satisfied with their due date punctuality than there are companies which are not (Table 2.8). An explanation might be given with the help of (Table 2.10) (satisfaction with stock levels). About half of the companies indicated that they were not satisfied with their stock keeping levels. Undoubtedly, there is a connection between stock levels and punctual delivery. By adding additional stock, punctual delivery increases, while problems in production planning might still exist. Many companies indicate that they have deficits in the production (Table 2.17). In Table 2.19, a central planning approach is most often suggested to tackle the deficits. Therefore, we will focus on a central planning approach.

	First method [1]	Second method [2]	Third method [3]	Fourth method [4]	Fifth method [5]	Sixth method [6]	Seventh method [7]
Imprecise determination of due dates / make-to-order	3	13	0	4	8	11	2
Unbalanced degree of capacity utilization	5	28	6	7	16	23	11
Imprecise sales planning	6	14	5	6	8	15	2
Insufficient treatment of completion confirmation data	2	16	1	4	12	7	0
Lacking tracking of progress towards completion	2	14	2	5	10	11	0
Planning with average and estimated values	3	18	2	4	8	13	1
Others	8	25	3	8	16	21	1
No Deficits	1	0	0	0	0	0	13
Total	12	44	8	11	25	32	27

[1] self-dependent planning and controlling of single process steps by the executing department (decentralized approach)
[2] continuous planning and controlling of the process steps by the central planning department (centralized approach)
[3] constant (automated) determination and forwarding of the order progress by the vendor to the customer
[4] constant (automated) determination and forwarding of the actual due date by the customer to the vendor
[5] Treatment of completion confirmation data in real time
[6] Provide practical reaction strategy for significant plan differences
[7] No statement

Table 2.19: Possible methods against deficits

As most of the companies do not incorporate (many) resource constraints in the planning (Table 2.12) the models and algorithms later explicitly incorporate further resource constraints, as this might be a reason for the deficits. Clearly, the requirements to provide enough IT equipment for planning in SMEs need to be fulfilled first. This result is similar in all studies. But this should not be a task for research but for consultants who should implement these systems, as the procedure and tasks related to such an implementation are widely known. With regard to the results, the limitations of the study will be clearly put up again. There is an overall limited significance caused by a lot of uncontrollable factors. Furthermore due to the related focus of regional companies on distinguished industrial sectors, a possible bias exists. Nevertheless, it is a first step in a mixed methods approach. The meaning for the following chapters is that the main results of the findings in the study will be taken as a rough guideline for the detail case investigation of the three analyzed companies. The results will be nevertheless examined carefully.

Chapter 3

Fundamental literature on scheduling with the consideration of setup operators

3.1 Scheduling review

Scheduling is the allocation of temporary tasks to resources to achieve a certain goal, for example, the minimization of tardiness.[3] It is used in a variety of areas such as health care (see for example Cayirli and Veral [2003] for an overview of outpatient scheduling in health care), project scheduling (see for example Brucker *et al.* [1999] for an overview of project scheduling), informatics (see for example Garey and Johnson [1977]), or as in our case machine scheduling.
The problem structures are nevertheless related to each other. Therefore, the widely known classification and depiction from Graham *et al.* [1979] can be applied. To give an overview of the most relevant works in the field of offline scheduling, where we refer to the publications which provide fundamental insights into this area and the most recent publications, the machine environment dimension from Graham *et al.* [1979] is adequate to classify the works. We therefore differentiate the problem by:

- Single machine,

- identical parallel machines,

- uniform parallel machines,

[3]With tardiness in general, we refer to the difference between a point in time t, a due date for the customer for example and the point in time t‘, when the job is finished, good is delivered etc., when t‘ is later than t.

- unrelated parallel machines,
- open shop,
- job shop,
- flow shop.

Further subcategories can be created by mixing these problems, such as flow shop with parallel machines. As there is much more literature on scheduling, the interested reader can find a review of the literature on machine scheduling problems from Abedinnia *et al.* [2017a,b].
Since most works focus on computational complexity, a brief introduction will be given for problems which are classified as NP-hard. There is no known algorithm to solve the problem in polynomial time. It is also unknown if the solution is verifiable in polynomial time. For NP-complete problems, there is also no known algorithm which will solve the problem in polynomial time, but a solution can verified in polynomial time.[4]

Single machine scheduling problem
For the single-machine scheduling problem, i.e. there is one machine on which jobs have to be scheduled, with the goal to minimize total tardiness Lawler [1977] provide pseudopolynomial algorithms. Du and Leung [1990] show that the problem is NP-hard. When the jobs have weights and the goal is to minimize total tardiness, Lawler [1977]; Lenstra *et al.* [1977] show that the problem is NP-hard.
If the goal is to minimize the number of late jobs, Moore [1968]; Maxwell [1970]; Sidney [1973] show that the problem is polynomially solvable. If the goal is to minimize the number of late jobs which have weights, Lawler and Moore [1969]; Karp [1972] demonstrate that the problem is NP-hard.
The single machine scheduling problem with the goal to minimize the weighted sum of completion times is shown to be solvable in polynomial time by Smith [1956].
Over time, many variants have been investigated such as precedence relations [cf. Lawler, 1973], release dates for jobs [cf. Lenstra *et al.*, 1977], or batches [cf. Brucker *et al.*, 1998]. Today there are even many more variants. Google Scholar lists more than 200 publications for single machine scheduling for the first quarter of 2018. Recent works include maintenance- [cf. Nesello *et al.*, 2018; Pacheco *et al.*, 2018] or sequence-dependent setup times [cf. Nesello *et al.*, 2018; Pacheco *et al.*, 2018; Chen, 2018] in the problem setting.

Identical parallel machines
Identical parallel machines indicate that there are jobs that need to be allocated and scheduled on one of the machines. The running time for a job is independent of the machine on which it is allocated. If the goal is to minimize the makespan, the most important results are that the problem is already NP-hard with two

[4]For more information see for example Garey and Johnson [1975].

machines [cf. Lenstra *et al.*, 1977] as well as for an arbitrary number of machines [cf. Garey and Johnson, 1978]. Mnich and Wiese [2015] show that some cases are polynomially solvable when certain variables are restricted. Jansen [2010]; Chen *et al.* [2014] provide approximation results. Recent works from 2018 focus on the development and improvement of algorithms for this problem. These works include for example Schwerdfeger and Walter [2018]; Mrad and Souayah [2018]; Sheremetov *et al.* [2018].

Uniform parallel machines
In contrast to identical parallel machines, in this case machines have a certain speed factor for the processing of jobs such that the processing time for jobs can differ due to the machine on which it is allocated. Only two publications are known to provide complexity results for uniform parallel machines. Jansen *et al.* [2016] provide an approximation algorithm with the help of sparsification techniques. Knop and Koutecky [2017] showed that if the processing time of a job p is limited to k, then there exists an algorithm with polynomial running time. As there are only a few recent publications in this section (Google Scholar lists 392 between 2014 and the end of the first quarter 2018), there is no clear research direction. Among the most recent publications are "Bi objective scheduling on uniform parallel machines" from Zeng *et al.* [2018], "Power of preemption for minimizing total completion time on uniform parallel machines" Epstein *et al.* [2017], or "Uniform parallel machine scheduling for minimizing total resource consumption with a bounded makespan" from Lin and Ying [2017].
,

Unrelated parallel machines
If the processing times of a task differ on the machines and there is no relation between the difference in processing times on the machines, the problem is called unrelated parallel machine scheduling. There are more complex results for this problem than for other parallel machine scheduling problems. When the goal is to minimize the sum of completion times, Horn [1973]; Bruno *et al.* [1974] show that the problem is polynomially solvable. Hoogeveen *et al.* [2001] show that the problem is non-approximable when the goal is to minimize weighted completion times. Knop and Koutecky [2017] show that there exists an algorithm with polynomial running time when the processing times and weights are limited. Further results show approximation algorithms for the goal to minimize makespan [cf. Lenstra *et al.*, 1990] and exceptional cases for the same goal which can be solved in polynomial time [cf. Knop and Koutecky, 2017]. When the most recent literature is reviewed, some focus seems to be on batch respectively lot sizing [cf. Eremeev *et al.*, 2018; Lu *et al.*, 2018; Tan *et al.*, 2018]. Another focus is on the consideration of further resources [cf. Villa *et al.*, 2018; Arbaoui and Yalaoui, 2018].

Open shop
In the open shop, there is a set of jobs. Each job consists of operations which can be scheduled in any order on the machines. If the goal is to minimize makespan, Gonzalez and Sahni [1976] show that the problem is NP-hard for more than two

machines. Williamson *et al.* [1997] also show that there is no polynomial-time approximation algorithm which creates a schedule with strictly less than 5/4 of the optimal schedule length. If the goal is to minimize the sum of completion times, Achugbue and Chin [1982] showed that the problem is NP-hard for two machines, for an arbitrary number of jobs and when the number of jobs is fixed. Hoogeveen *et al.* [2001] show that there exists no polynomial approximation algorithm for this target. Lawler *et al.* [1981] prove that the two machines minimize maximum lateness problem is NP-hard when the number of jobs is limited or unlimited. In the case that the number of tardy jobs shall be limited, Liu and Bulfin [1988] show that the problem is polynomially solvable if all processing times equal one. Recent works provide no clear direction. Some publications focus on providing algorithms for open-shop scheduling problems [cf. Tanimizu *et al.*, 2017; Bai *et al.*, 2017]. Other publications focus on the amendment of existing models and their complexity [cf. Tellache and Boudhar, 2017].

Job shop
As opposed to the open-shop problem, operations in the job shop problem need to go on the machines in a certain order. The order can be different for each job. Among the most important works are from Lenstra and Kan [1979] which prove that the problem with two machines and the target to minimize the makespan is NP-hard, as well as the problem with three machines when all processing times are set to one. Sotskov and Shakhlevich [1995] prove that the problem is NP-hard with three machines and three jobs. When the target is to minimize the sum of completion times, Garey *et al.* [1976] show that the problem is NP-hard with two machines. Three machines and three jobs are also NP-hard [cf. Sotskov and Shakhlevich, 1995]. If there are two machines and the processing time is set to one, the problem is polynomially solvable [cf. Kubiak and Timkovsky, 1996]. If the jobs have, however, weights, the problem becomes NP-hard [cf. Timkovsky, 1998]. The problem is also polynomially solvable with two machines, the goal to minimize the number of tardy jobs when the processing time is set to one [cf. Kravchenko, 1999]. They also show that the problem becomes NP-hard when the goal is to minimize the number of weighted tardy jobs. Timkovsky [1998] shows that this problem (two machines, processing times set to one) is NP-hard when the goal is to minimize maximum weighted tardiness. When the most recent works in job shop scheduling are examined, many of them provide new algorithms [cf. Rameshkumar and Rajendran, 2018; Nouiri *et al.*, 2018; Dao *et al.*, 2018]. Another, but smaller research stream is the extension of the problem [cf. Zhao *et al.*, 2018; Piroozfard *et al.*, 2018; Devassia *et al.*, 2018]

Flow shop
The flow shop problem can be described as a set of jobs with operations which must be produced on a chain of machines. The sequence for the operations of the jobs on the machines has to be the same for each job. Garey *et al.* [1976] show that the problem is NP-hard for three machines and the goal to minimize makespan. Williamson *et al.* [1997] show furthermore that there is no polynomial time approximation algorithm which provides schedules shorter than 5/4

of the optimal schedule. If only two machines are considered, the problem is polynomially solvable [cf. Johnson, 1954]. In the case where the goal is to minimize the sum of completion times, the problem is NP-hard with two machines [cf. Garey *et al.*, 1976]. Hoogeveen *et al.* [2001] show, for an arbitrary number of machines, that there is no polynomial time approximation algorithm. The two-machine flow shop problem with the goal to minimize maximum lateness is also proven to be NP-hard [cf. Lenstra *et al.*, 1977]. When the recent works are considered, there is a clear and heavy focus on hybrid flow shop problems. That is, there is a flow shop but at certain or all stages there are also parallel machines. Recent publications are for example from Dios *et al.* [2018]; Shahvari and Logendran [2018]; Mousavi *et al.* [2018]. Another focus is flow shop scheduling with permutation. In this case, jobs cannot overtake each other on machines. Thus, a permutation of the jobs is created as a solution and this permutation will be the same on all machines. Examples come from Abdel-Basset *et al.* [2018]; Fernandez-Viagas *et al.* [2018].

3.2 Scheduling with setup operators review

While Section 3.1 has provided a general introduction to and status quo of the literature in the field of scheduling, this section will provide more details on the specific area of scheduling under the explicit consideration of operators as a scarce resource, which was mentioned as a specific factor in the conducted survey which led to this focus overall. The structure in this section will follow those of Section 3.1 except for the cases with parallel machines. In the area with the consideration of operators, the literature can better be divided by the degree of freedom for the parallel machine planning than by the processing time variations.

While there exists a review of operator/machine interference problems from Stecke and Aronson [1985], which also tackles the problem of setup operator availability, numerous works have been published since then. In our literature overview (Table 3.1) regarding machine scheduling offline problems with the explicit consideration of operators, we classify by distinguishing between two types of operations, the setup for the job (the time needed to setup a machine to produce a job) and the job's processing (the actual production of the good) itself. In both cases, operator availability might act as a further restriction. In an informal way, by scheduling we refer to the allocation of tasks to resources and the creation of a sequence for these tasks. In our case, we allocate jobs to machines (if the allocation is not predetermined), create a sequence of jobs on machines (if not predetermined) and create a sequence for the setup of the jobs.

Multiple setup and machine operators
The most general case in regard to operator availability constraints in the field of scheduling is, when multiple operators need to setup jobs on machines and another type of operators need to attend the machines during the processing phase. As for all multiple operator cases described in this paper, a job needs at most one operator at any point in time. To the best of our knowledge, the problem is only dealt with once [cf. Chen *et al.*, 2003] and is therefore not mentioned explicitly in Table 3.1. They consider a job shop context with transfer lots. The initial problem is relaxed and decomposed into smaller sub-problems, which are then solved with a dynamic programming approach. Finally, a heuristic is used to build a feasible schedule out of the solutions of the sub-problems.

Single setup and single machine operator
When only a single setup operator and a single machine operator are considered—i.e. only one person can attend a setup and only one other person can attend the job during its processing—no contribution has been made to the best of the authors knowledge.

Multiple setup operators
Also in the multiple setup operator case, there is a group of setup operators and one needs to attend the setup of the job, only limited contributions have been made so far. In the related field of **lot-sizing**, Tempelmeier and Copil [2012] integrate parallel common setup operators in the capacitated lot-sizing problem (CLSP) with linked lot-sizes, multiple machines and sequence-dependent setups. They then solve the problem with a fix and optimize heuristic based on Helber and Sahling [2010].

Werner and Kravchenko [2010] provide complexity results for the **parallel machine** scheduling problem with multiple setup operators. They provide a polynomial algorithm for the makespan minimization case with equal setup and equal processing times and a pseudopolynomial algorithm for the case with unit setup times, m machines and m-1 setup operators. They prove, furthermore, that the problem of minimizing maximum lateness with a fixed number of machines and servers is NP-hard. Finally, they provide a worst-case description of two-list scheduling algorithms for the parallel machines makespan minimization problem. Kerkhove and Vanhoucke [2014] present a hybrid metaheuristic for the parallel machine scheduling problem with due and release dates for jobs, sequence-dependent setups and limited setup operators. Furthermore, machines are arranged in geographically dispersed locations such that due dates and the objective function (minimize weighted total tardiness) are influenced by the machine–job combination. Kerkhove and Vanhoucke solve the problem in two phases. They first use a combination of GA and SA without considering the limited setup operator availability. They then correct the solutions using standard dispatching rules such as first come first served for the setup operators.

Single setup operator
In the associated case, where only a single setup operator needs to attend the setup of the job is considered, publications in different problem contexts are known. Regarding **lot sizing**, Tempelmeier and Buschkühl [2008] propose a model formulation, based on the proportional lot-sizing and scheduling problem [cf. Haase, 1994] and a reformulation based on the simple plant location problem. Tempelmeier and Buschkühl are able to find good solutions with the help of standard solvers within a few minutes of central processing unit (CPU) time. In a subsequent work Tempelmeier and Copil [2016] integrate a single setup operator into a CLSP with parallel machines and sequence-dependent setups.

Cheng and Sriskandarajah [1999] analyse the single setup operator problem in the field of a two-machine **flow shop**, when setup and dismounting times exist, and the operator can only setup respectively dismount jobs in a cyclic pattern. The goal is to reduce makespan. Furthermore, they distinguish between setups and dismounting operations which can be conducted without the actual job on the machine, and those, in which the actual job on the machine is needed. The only decision to be made is the order of jobs on the first machine. They show that both problems are NP-complete. Glass *et al.* [2000] show that the open-shop and flow-shop problem with two machines, a setup operator, and the makespan minimization goal are NP-hard. Additionally, the two-machine, no-wait flow shop problem with a single setup operator is shown to be solvable in polynomial time, while the two-machine open-shop problem with a single setup operator is NP-hard. Furthermore, they present some exceptional cases of the parallel dedicated machines problem[5] which are solvable in polynomial time such as when only two machines are considered, or the case when all setup and processing times equal a constant c. Moreover, they show a greedy algorithm for the multiple machine case, which has worst case ratio of two and an improved heuristic for the two-machine case with a worst-case ratio of 3/2. Brucker *et al.* [2005] provide complexity results for a flow shop with a single setup operator when setup times are separable from the jobs. They analyse problems with variations in the number of machines (two, arbitrary), setup times (constant, unit), processing times (constant, unit) and the objective function (makespan, total completion times, total tardiness, maximum lateness, number of tardy jobs, with and without weights). They show that the problem with a setup operator is at least as difficult as the corresponding classical flow shop problem without a setup operator. Su and Lee [2008] investigate a two-machine no wait, separate setup flow shop with a single setup operator with the goal to minimize the total completion times. They show optimal solutions for some restricted cases and properties for the general case, from which a heuristic and a branch and bound is established. Then, they use their methods on the special case of the problem, without the setup operator and compare it to the methods proposed by Aldowaisan [2001]. Although their heuristic is faster, the

[5]Jobs are already dedicated to the machines in advance. Therefore, no assignment of jobs needs to be made. The only decision to be made is the sequence of jobs on the machines.

branch and bound algorithm from Aldowaisan requires smaller execution times, as the number of jobs increases.

Several works have been published in the area of a single setup operator and two or more **parallel machines**. Koulomas [1996] show that the problem with two machines, arbitrary setup and processing times and the objective to minimize idle time is NP-hard by reducing it from the three-partition problem, which is shown to be NP-hard by Garey and Johnson [1975]. Moreover, they provide a beam search heuristic, which works efficiently with two machines but can also be adapted to a context with more than two parallel machines. Kravchenko and Werner [1997] present a pseudo-polynomial algorithm for the two-machine case when all setup times are one and the goal is to minimize makespan. They also prove that in the more general case with an arbitrary number of machines, the problem is NP-hard. Finally, complexity results for different variants when the objective function is not the minimization of the makespan, but the idle time, are shown. Blazewicz *et al.* [1999] investigate parallel machines with a single setup operator problem with setup and dismounting times. They prove that the problem is already NP-hard for the two-machine case. It is, however, polynomially solvable when all jobs are identical. Furthermore, it is shown that any list scheduling algorithm has a worst-case approximation ratio of three times the optimal value for the makespan minimization goal. Hall and Sriskandarajah [2000] provide additional complexity results for the single setup operator and two or more parallel machines. They distinguish between the number of machines (two, arbitrary), setup times (constant, unit, arbitrary), processing times (unit, arbitrary) and the objective function (makespan, total completion times, total tardiness, maximum lateness, number of tardy jobs, with and without weights). Wang and Cheng [2001] provide an approximation algorithm for this problem when the goal is to minimize the weighted sum of the completion times. They show that their algorithm builds solutions such that the completion time of a job is not more than $(5 - \frac{1}{m})$ times away from the completion time in an optimal solution. They also provide further results for special cases of their problem. Kravchenko and Werner [2001] introduce an algorithm for the parallel machines one setup operator problem with the goal to minimize the sum of completion times, when all setup times equal one. They show that their algorithm has a worst case bound of $n^{'}(m-2)$ times the difference to the sum of the completion times in the optimal schedule. $n^{'}$ is the number of jobs that have a processing time less or equal $m - 1$. Brucker *et al.* [2002] present additional complexity results for this problem setting. They differentiate by the number of machines (two, arbitrary), release dates (unit, constant, arbitrary), setup times (unit, constant, arbitrary), processing times (unit, constant, arbitrary), and objective functions (makespan, total completion times, weighted total completion times, total tardiness, weighted total tardiness, maximum lateness, number of tardy jobs, weighted number of tardy jobs). Abdekhodaee and Wirth [2002] prove that the two-machine, one-setup-operator makespan minimization problem is NP-hard. They also provide polynomial time algorithms for the two special cases when each processing time of a job is not larger than each setup time or when jobs are of equal length and the number of jobs is even. In a subsequent paper, Abdekhodaee *et al.* [2004] investigate two special cases for the two parallel machines one setup operator setting when jobs have identical processing times and each setup is less or equal to the processing times and when jobs have equal setup times and each processing time is longer than the setup time. Both cases are NP-complete. In 2006, they use various heuristics, such as a GA on the two machines one setup operator, makespan minimization problem. They can achieve solutions which are at maximum 5% higher than the lower bound and at an average 2% higher. Gan *et al.* [2012] created for this problem context an MIP model and two branch-and-price variants based on the models. They test the methods together with two greedy heuristics established by Abdekhodaee *et al.* [2006] and the 'XPressMP' solver based on the formulated MIP model with the help of randomly generated data. For small cases, the MIP solver shows superior performance, whereas for larger instances, the branch-and-price variants show superior performance. Hasani *et al.* [2014d] developed a SA and a GA for the two parallel machines, single-server, makespan minimization problem, which is tested on instances with up to 1000 jobs. On the whole, their heuristics show a reliable performance and is close to lower bounds. Hasani *et al.* [2014a] furthermore use the concept to decompose a schedule into blocks to provide a MIP model for the same problem. Their model clearly outperforms all existing models in the literature for this problem setting. When the goal is to reduce forced idle time and jobs have to be scheduled alternatively on two machines, Hasani *et al.* [2014b] describe a MIP model and present a hybrid heuristic consisting of a constructive algorithm and a tabu search. They tested it on problems with up to 100.000 jobs. In another study, Hasani *et al.* [2014c] consider the problem of minimizing total weighted job completion time for an arbitrary number of parallel machines with a single server. They propose an approximation algorithm with a worst-case ration of $(3 - \frac{1}{m})$. Therefore, they improved the existing approximation algorithm from Wang and Cheng [2001] with a worst-case ratio of $(5 - \frac{1}{m})$. Kim and Lee [2012] formulate two MIP models for the case of arbitrary parallel machines, a single server and the makespan minimization goal. They furthermore created a hybrid heuristic, combining SA and TS. The heuristics are tested with respect to an optimal value or best-found value by 'Cplex in 3600 seconds. The average gap is $2,06\%$ to the values calculated by 'Cplex. Hasani *et al.* [2016] present two constructive algorithms for the two-parallel machine, one single setup operator, minimizing the makespan problem. These work well in large problem instances with up to 10.000 jobs but are not superior to the approaches presented from Hasani *et al.* [2014d] and Hasani *et al.* [2014a] in small and medium-sized instances.

Blazewicz *et al.* [1999] deal with the **parallel machine problem when jobs are dedicated to the machines**. They show that any list scheduling algorithm has a worst-case scenario of two times the optimal value. Huang *et al.* [2010] investigated this problem with sequence-dependent setup times. They propose a mixed integer formulation and a hybrid GA using characteristics of GA and branch-and-bound methods. To test their algorithm, Huang *et al.* propose lower bounds. They can find solutions close to the lower bound for up to 10 machines and 100 jobs. Apart from the hybrid GA, Huang *et al.* show a special case of the problem which is solvable in polynomial time. Xie *et al.* [2012] investigate a problem where there are not only setups for the jobs, but also dismounting times. They present an approximation algorithm with a worst-case performance ratio of two.

Some papers consider a setting where jobs are not only assigned to machines, but the sequence of jobs is predetermined for each machine (**parallel dedicated machines with job chains**). Then, the only decision to be made is the setup sequence. Wikum *et al.* [1994] deal with this problem. They describe chains of jobs, which have to be scheduled for a single machine with minimum time lags between these jobs on a chain. Transferred to our context, the single machine corresponds to the setup operator, while the jobs are setups. The minimum time lags can be regarded as processing times. They prove for instance that the problem is solvable in O(k log k) time, where k is the number of chains, and each chain consists only of one job but becomes NP-complete as soon as one chain has two jobs. Munier and Sourd [2003] show three cases which are solvable in polynomial time. (1st) Finding the optimal schedule in the makespan minimization problem, when all setups have the same duration o and all processing times have the same duration d. (2nd) Finding the optimal schedule in the makespan minimization problem, when each processing time is less than the shortest setup time. (3rd) Finding the optimal schedule when the goal is to reduce the flowtime where all processing times are smaller than a constant setup time. Brucker *et al.* [2006] extend these results by showing that when all jobs have the same duration o and all delays have the same duration d and the goal is flowtime or makespan minimization, the problem is solvable in polynomial time. Schauer and Schwarz [2013] study a so-called body shop scheduling problem (BSSP) which corresponds to a special type of the job shop problem. In addition to the classical job shop, machines are connected to a so-called laser source. A machine can only process a job if no other machine is processing a job. Moreover, the machines need to move to the jobs. While moving takes zero time, machines cannot bypass each other, where they are aligned on a straight line. The single-machine scheduling problem with job chains can be seen as a restricted subproblem of the BSSP. (Note that the single setup operator and the single machine operator with parallel dedicated machines with job chains correspond both to the single machine scheduling problem with job chains). They show that the problem is polynomially solvable with two chains but becomes NP-hard with three. Thus, this means that our problem is polynomially solvable with two machines, but becomes NP-hard with more than two machines. A general overview and complexity results for other types of single machine problems with precedence delays can be found in Brucker and Knust [1999].

Multiple machine operator case
Different works have been published for the multiple machine operator case.

That is there is a group of machine operators and each job needs to be attended during its processing phase by one operator. Agnetis *et al.* [2011] investigate this problem in the **job shop context**. They show that the problem is already NP-hard with three jobs, three machines and two operators or with n jobs, which have only one task and three machines with two operators. Note that the latter problem could be regarded as three parallel dedicated machines. They also present a dynamic programming algorithm that runs in pseudopolynomial time if the number of jobs is fixed, a fully polynomial time approximation scheme, a branch and bound approach and two heuristics.

Bourland and Carl [1994] investigated two cases of the so-called fractional operator problem. There are several **parallel machines** which have to process several jobs. Each job needs a certain proportion of an operator during its processing. Hence, an operator can attend several machines at once if he has ample capacity. Bourland and Carl [1994] distinguish between two consecutive problems (planning and control). While the goal in the planning stage is to find a cost minimal cyclic schedule for a constant demand rate, in which each product can only be set up once, the goal in the control stage is to find a short term schedule, based on the outcome of the planning stage, which suits known, but unstable demand, whereby products can be set up more than once. Furthermore, they present a dynamic programming example for the control stage.

Kellerer and Strusevich [2004] provide complexity results for different variants of the multiple machine operator problem with **parallel dedicated machines**. They differentiate between the number of machines which must be served (two, three, arbitrary), the number of operators available (one, two, three, arbitrary), how much capacity a job needs (one, arbitrary), the capacity of the resources (one, two, arbitrary), family setup or removal times, and job pre-emption. In all cases, the objective is makespan minimization.

Single machine operator
A special case of the multiple machine operator is when only a single machine operator is present in such a way that only one machine can/must be attended when a job is being processed. Espelage and Wanke [2000] analysed a **flow shop** with buffers in front of each machine and only one worker, who can supervise one machine at a time. A further restriction is that he can only change his position if there are no more jobs waiting at the current machine. The goal is to minimize the movement of the worker, where a movement is considered as the change of a worker between two machines. They show that when the job order is fixed, and the buffer size is restricted to two, a polynomial algorithm is found. In the case when a job order has to be found and the least number of buffers has to be two, the problem is NP-complete. They also show in 2003 that the problem, when there are no buffers in front of the machines, is NP-complete, and provide a linear time approximation scheme with a ratio two for that case. Bako and Vickson [2004] investigate a single-machine operator setting in the open and flow shop case with two machines. Each time the

operator starts working on a different machine, he must set it up first. Thus, jobs are processed in batches on a machine. They show that the problem of minimizing the number of tardy jobs is NP-hard. Furthermore, they provide pseudopolynomial dynamic programming algorithms for both cases.

Kellerer and Strusevich [2003] provide complexity results for the m **parallel dedicated machines** makespan minimization problem, where only some jobs need an operator. They show that the problem is solvable in polynomial time in the two-machine case if pre-emption is not allowed, or in the multiple machine case, where the pre-emption of jobs which do not need an operator is permitted. Additionally, it is shown that the problem with three machines without pre-emption is NP-hard in the ordinary sense and problems where the number of machines is part of the input are NP-complete even if pre-emption of resource jobs is allowed. Finally, heuristic algorithms based on the group technology approach are presented for the specific contexts and their worst-case performance is investigated.

From a scientific perspective, the focus so far is more on providing complexity results for the different problem types rather than on the development/implementation of algorithms to tackle real world problems around setup operator scheduling. With a view to filling this research gap, we first focus on the most restricted case—the single setup operator parallel dedicated machines problem (Chapter 4). We then approach the more general case of parallel machines with multiple setup operators, in which only two contributions have been made so far (Chapter 5). Finally, we create a new problem context by adding parallel machines to a job shop (Chapter 6). Although the problem complexity increases with that, we do not know why this type of problem context has not been regarded in the literature so far, as it has high practical significance.

	multiple setup operators	single setup operator	multiple machine operators	single machine operator
open shop		Glass *et al.* [2000]		Bako and Vickson [2004]
job shop			Agnetis *et al.* [2011]	
flow shop		Cheng and Sriskandarajah [1999]; Glass *et al.* [2000]; Brucker *et al.* [2005]; Su and Lee [2008]		Espelage and Wanke [2000, 2003]; Bako and Vickson [2004]
parallel machines	Werner and Kravchenko [2010]; Kerkhove and Vanhoucke [2014]	Koulomas [1996]; Kravchenko and Werner [1997]; Blazewicz *et al.* [1999]; Hall and Sriskandarajah [2000]; Wang and Cheng [2001]; Kravchenko and Werner [2001]; Brucker *et al.* [2002]; Abdekhodaee and Wirth [2002]; Abdekhodaee *et al.* [2004, 2006]; Gan *et al.* [2012]; Kim and Lee [2012]; Hasani *et al.* [2014d,c,a,b, 2016]	Bourland and Carl [1994]	
parallel dedicated machines		Koulamas and Smith [1988]; Blazewicz *et al.* [1999]; Glass *et al.* [2000]; Huang *et al.* [2010]; Xie *et al.* [2012]	Kellerer and Strusevich [2004]; Agnetis *et al.* [2011]	Kellerer and Strusevich [2003]
parallel dedicated machines with job chains		Wikum *et al.* [1994]; Munier and Sourd [2003]; Brucker *et al.* [2006]; Schauer and Schwarz [2013]		

Table 3.1: Classification of relevant literature

Chapter 4

Parallel dedicated machines subject to setup constraints

4.1 Introduction

In Chapters 4, 5, and 6, we analyse the problems indicated in Chapter 2 by the survey.[6] The three companies chosen for detailed investigation fit perfectly within most of the companies interviewed from the survey. On the one hand, the companies use a make-to-order production strategy. Some products of the three companies analysed in detail are also created in batches. Thus, we also take care of the second most mentioned production type, make-to-stock. The arrangement of the material flow and the machines which they possess furthermore underline that the make-to-order strategy for these three companies seems correct. The companies use universal machines for production so that they can easily produce various parts with the same machines. Moreover, the material flow also follows this strategy. There is no single flow of material through the machines. Instead, the type and sequence a job must go depends on the job themselves. Therefore, we talk about a job shop.
This part of the research is based on a medium-sized company from Germany from the automotive sector with the focus on the production of small metal parts. The number of people employed as well as the revenue are well in the quantitative definition of the IfM [2002]. The company is owned and run by a family. The CEO of the company is directly involved in sales. According to Theile [1996], this is also an indicator for an SME.
Prototypical, we analysed one of their factories with 13 machines on which several thousand pieces are produced every day. When the results from Section

[6]The description and connection of the companies to the empirical study has to be done carefully. While the companies are in line with the study, this means they have the same disadvantages. In the contracts which were made for the case study, anonymosity was assured to them. Hence it is a balancing actto give enough information to the reader and provide anonymosity.

2.3 are considered, it can be said that while the company has an ERP system, the scheduling is done manually. There is a planning department which creates production plans using Excel. Jobs are dedicated to machines regarding machine capabilities but also the running costs. Each machine has ten to twenty jobs to fulfil. Problems due to the creation of unfeasible plans are dealt with directly on the shop floor. It can therefore be said that the company shows the same characteristics as most of the companies in the empirical study. Hence, it is expected that results from this section can be transferred to majority of SME with production. As the company uses universal machines, before a job can be produced on a machine, the machine has to be set up for the job. The setup itself involves the change of the tooling, the adjustment of the apparatus for the pieces which are to be produced, as well as the programming of the computer numerical control (CNC) system with the job specific parameters.

The disadvantage of universal machines is that they tend to be labor intensive [cf. Pfohl, 2006]. The setup needs to be done by a setup operator. In this case, a setup operator is a dedicated person who has the necessary skills and tools for setting up machines. No one else at the company is allowed or able to perform the changeovers. Setup operators work in two shifts. The early shift starts at 6 a.m. and continues until 2 p.m., with a break between 10 and 10:30 am. The second shift starts at 2 p.m. and continues until 10 p.m., with a break from 6 p.m. to 6:30 p.m. In any shift, there is only one setup operator, meaning that at any point in time, only one setup is possible. On the contrary, the machines are operated in three shifts, seven days a week, meaning they are always available to process jobs.

Each time, the setup operator switches between machines, a traveling time occurs. The traveling time consists partly of the distance between the two machines, which the setup operator must cover but also the time, the setup operator needs to return tools no longer needed and pick up new ones.

The main production goal is to minimize the total tardiness. New production jobs are added once a day regarding their due date at the end of the existing production plan. By that rule, new production jobs can be easily integrated in the existing plan while the production goal is still considered. There is no planned schedule for the setup operator himself. Instead, the planning department sets up the next available job. If several jobs can be chosen, it sets up the job, which was available first.

When using this due date heuristic on the data we received from the company, some jobs are still late. At the same time, it can be seen that setup operators and machines have idle times. Hence, computerized scheduling, taking setup operators into account might provide better results than the due date heuristic currently applied. Our aim is to provide a schedule for the setup operator which minimizes total tardiness. Based on the given production plan by the company, we apply our methods as described in Section 4.3.

For this reason, a modified approach is proposed, which explicitly incorporates setup operator availability. Furthermore, we include time periods in which the machines cannot be set up and travel times between machines for the operator.

4.2 Problem definition

In what follows, we consider a set of machines $1, ..., M$. For each machine m there are preassigned jobs $j = (1, m), \ldots, (n_m, m)$ which have to be processed in a given sequence. A job is specified by a pair (j,m) consisting of the job index and a machine number. For notational convenience, we assume that jobs are numbered accordingly. There are N jobs in total. Each job (j, m) requires processing time $p_{j,m}$ and is preceded by a setup of duration $s_{j,m}$. Each machine can process one job at a time. Pre-emption, the interruption of a job, to produce a different job, or to make a break, is not allowed either for jobs or for setups. For each job and each setup, the start time (and, hence, the completion time) is to be decided. For a schedule to be feasible we require setups to be non-overlapping. The goal is to minimize total tardiness. Note that we can assume that each job follows its preceding setup without any idle time in between.
As described before, the problem can also be seen as a single machine scheduling problem with job chains and non-negative minimum time lags between consecutive jobs in a chain with the goal to minimize the sum of tardiness. The setup operator in the parallel machine case can be regarded as the single machine in the single machine scheduling problem. Setups in the parallel machine case are jobs in the single machine scheduling problem. Processing times in the parallel machine case correspond to minimum time lags in the single machine scheduling problem. Using the three-field notation introduced by Graham *et al.* [1979] our problem, can be represented as $(1|chains(l_{ij} >= 0)|\sum_{j=1}^{J} T_j)$, whereby T_j is the tardiness of a job j.

We additionally consider two generalizations of the problem setting described above. First, we incorporate breaks $a = 1, \ldots, A$. A break is a time interval where the setup operator cannot work and, therefore, no setup can overlap with a break. Second, we consider travel times between machines for the setup operator. We denote the travel time needed between the end of a setup on machine l and the start of a setup on machine m by $h_{l,m}$. In a formal setting, our model is defined by:

Indices	
m	Machine index, m = 1, …, M
j	Job index, j = 1, …, n_m
a	Break index, a = 1, …, A
Parameters	
$p_{j,m}$	Processing time of job (j,m)
$s_{j,m}$	Setup time of job (j,m)
bs_a	Beginning of break a
be_a	End of break a
K	Adequate large number
$h_{l,m}$	Travel time from machine l to machine m
N	Number of jobs in total
$L_{j,m}$	Delivery date of job (j,m)
Variables	
$t_{j,m}$	Tardiness of job (j,m)
C_{max}	Makespan
$\gamma_{i,l}^{j,m}$	Binary variable which equals 1 if the start of the setup for job (j,m) follows start of the setup for job (i,l) directly
$\beta_{j,m,a}$	Binary variable which equals 1 if job (j,m) starts after break a
$S_{j,m}$	Starting time of the setup for job (j,m)

Table 4.1: Notation parallel dedicated machines

4.2.1 Standard model formulation

$$min \quad \sum_{m=1}^{M}\sum_{j=1}^{n_m} t_{j,m} \tag{4.1}$$

$$S_{j+1,m} - p_{j,m} - s_{j,m} - S_{j,m} \geq 0 \qquad \forall \quad j = 1,\dots,n_m - 1, m = 1,\dots,M \tag{4.2}$$

$$S_{j,m} - S_{i,l} + (1 - \gamma_{i,l}^{j,m}) \cdot K \geq s_{i,l}$$

$$\forall \quad j = 1,\dots,n_m, i = 1,\dots,n_l, m = 1\dots,M, l = 1,\dots,M, m \neq l \tag{4.3}$$

$$\sum_{m=1}^{M} \sum_{\substack{j=1 \\ (j,m)\neq(i,l)}}^{n_m} \gamma_{i,l}^{j,m} \leq 1 \qquad \forall \quad i = 1,\dots,n_l, l = 1,\dots,M \tag{4.4}$$

$$\sum_{l=1}^{M} \sum_{\substack{i=1 \\ (j,m)\neq(i,l)}}^{n_l} \gamma_{i,l}^{j,m} \leq 1 \qquad \forall \quad j=1,\dots,n_m, m=1,\dots,M \quad (4.5)$$

$$\sum_{m=1}^{M} \sum_{j=1}^{n_m} \sum_{l=1}^{M} \sum_{\substack{i=1 \\ (j,m)\neq(i,l)}}^{n_l} \gamma_{i,l}^{j,m} = N-1 \quad (4.6)$$

$$(S_{j,m} + p_{j,m} + s_{j,m}) - L_{j,m} \leq t_{j,m}$$

$$\forall \quad j=1,\dots,n_m, m=1,\dots,M \quad (4.7)$$

$$t_{j,m}, S_{j,m} \geq 0 \qquad \forall \quad j=1,\dots,n_m, m=1,\dots,M \quad (4.8)$$

$$\gamma_{i,l}^{j,m} \in \{0,1\}$$

$$\forall \quad j=1,\dots,n_m, i=1,\dots,n_l, m=1,\dots,M, l=1\dots,M, (j,m)\neq(i,l) \quad (4.9)$$

$$\gamma_{i,m}^{j,m} = 0. \qquad \forall \quad i=1,\dots,j-1, j=2,\dots,n_m, m=1,\dots,M \quad (4.10)$$

The objective function (4.1), minimize the sum of tardiness $t_{j,m}$ for all jobs $m = 1,\dots,M, j = 1,\dots,n_m$, together with constraint (4.7), the tardiness for a job j,m must be greater or equal the starting time for the job $S_{j,m}$ + its processing time $p_{j,m}$ + its setup time $s_{j,m}$ - its due date $L_{j,m}$, represent the goal to minimize total tardiness. Constraint (4.2) ensures the non-overlapping of jobs on a machine. The difference between the starting time of a successor job $S_{j+1,m}$ and the sum of the starting time for the predecessor job on the same machine $S_{j,m}$, the processing $p_{j,m}$ and setup time $s_{j,m}$ of the predecessor job must be positive. Constraint (4.3) ensures the non-overlapping of setups on different machines. If a job j on machine m is the direct successor of job i on machine l ($\gamma_{i,m}^{j,m}$), the starting time of job j,m - the starting time of job i,l must be greater or equal the setup time of i,l $s_{i,l}$. Constraint (4.4) rules that each job i,l has maximum on successor, while constraint (4.5) rules that each job j,m has maximum one predecessor. Constraint (4.4) ensures that there is a strict sequence of successor–predecessor relationships and no cycles. There is number of jobs - 1 relationship overall. Therefore, cycles are not possible as in this case, there would be more relationships as number of jobs - 1. Constraints (4.8), (4.9) and (4.10) define the variables domains. That means they describe the limitations of the variables.

4.2.2 Extensions

The model can be further enhanced by incorporating breaks and travel times between machines for the setup operator as described in Section 4.1. They are not part of our core problem, but of course have a high practical relevance.

$$S_{j,m} + s_{j,m} \leq bs_a + \beta_{j,m,a} \cdot K$$

$$\forall \quad j = 1, \ldots, n_m, m = 1, \ldots, M, a = 1, \ldots, A \tag{4.11}$$

$$S_{j,m} \geq be_a \cdot \beta_{j,m,a} \qquad \forall \quad j = 1, \ldots, n_m, m = 1, \ldots, M, a = 1, \ldots, A \tag{4.12}$$

$$S_{j,m} - S_{i,l} + (1 - \gamma_{i,l}^{j,m}) \cdot K \geq s_{i,l} + h_{l,m}$$

$$\forall \quad j = 1, \ldots, n_m, i = 1, \ldots, n_l, m = 1, \ldots, M, l = 1, \ldots, M, m \neq l \tag{4.13}$$

$$\beta_{j,m,a} \in \{0, 1\} \qquad \forall \quad j = 1, \ldots, n_m, m = 1, \ldots, M, a = 1, \ldots, A \tag{4.14}$$

$$min \quad C_{max} \tag{4.15}$$

$$S_{n_m,m} + p_{n_m,m} + s_{n_m,m} \leq C_{max} \qquad \forall \quad m = 1, \ldots, M \tag{4.16}$$

Constraint (4.11) and (4.12) rule that the starting time for a job $S_{j,m}$ and its setup $s_{j,m}$ must either be finished before a break a starts bs_a or the starting time must be greater or equal to the end of break a be_a. They are connected through binary variable $\beta_{j,m,a}$ and ensures that one of the constraints is met and the other one has not to be fulfilled. Constraint (4.13) extends constraint (4.3) by incorporating travel times for the operator between the machines $h_{l,m}$. Constraint (4.14) defines the domain of the binary variable. Constraint (4.15) and (4.16) replace constraint (4.1) and (4.7) when the goal is to reduce makespan.

4.3 Methods

When we tried to find the optimal solution by dynamic programming. We thought that even while we might not be able to find the optimal solution when all constraints are included, we might be able to solve the problem with less constraints. While searching for ideas how to do it, we found a work which showed that if one chain has more than one job the problem is NP-complete [Wikum *et al.*, 1994]. Hence, we focus on metaheuristics to provide good solutions in a short time frame for our much larger real-world problem sizes. Metaheuristics are a type of heuristics which can be applied to a variety of problem settings. Although they do not guarantee that the optimal solution will be found, they are generally able to provide good solutions with reduced computational time in contrast to algorithms which will guarantee optimal solutions [Blum and Roli, 2003]. There are different ways to classify metaheuristics. As the focus is on different search strategies, we differentiate them by local vs global search. We define a local search method as an algorithm which explores a neighbourhood deeply and continues only slowly to other neighbourhoods. A global search on the other hand explores a single neighbourhood only roughly and moves very quickly to other areas, such that a wider proportion of the overall solution space is explored. We use the TS for the local search algorithms and the GA in the field of global search algorithms. Both can be said to be the most common metaheuristics for the local, respectively the global search. This might also be shown by the number of publications found in the EBSCO host database (1.887 for TS 10.358 for GA) [EBSCO, 2017]. SA was furthermore included as it combines the advantages of the local search with those of the global search. In the beginning, the algorithm moves fast and broad through the solution space while it focuses after some running time more intensively on the area it is in. The idea is akin to the idea of the structure of our also used VNS: Search in the beginning broadly to identify promising neighbourhoods which shall then be explored in more detail.

In total, six different variants of TS, GA, and SA were used to solve the previously described problem. The methods are either used separately or combined in a VNS. In the VNS, the six different variants are used in ascending order regarding the duration they need to search the solution space. This order was experimentally determined to be:

1. GA 2-point Crossover repair after each exchange (V1),
2. GA 2-point Crossover repair at end (V2),
3. GA 1-point Crossover,
4. TS Swap,
5. TS Insert,
6. SA.

The VNS algorithm can run for a maximum predetermined running time q. Every method has a maximum of $q/6$ minutes time before we switch to the next method, except when it does not find a better solution within half the allowed time. Then the method is also switched and the remaining time is added to the maximum running time of the following method. In our experiments, the duration for the switching was found out with the help of test runs.
We use heuristics to find starting solutions for the methods. For the TSs and the SA, only the best solution is taken. For the GAs, all solutions are used for creating the starting population:

1. Of all jobs allowed to be processed with regard to the chains, set up the job with the earliest due date. Repeat until all jobs are processed. The motivation for the method is minimizing the sum of tardiness.

2. Of all jobs allowed to be processed with regard to the chains, set up the job with the least slack time. Repeat until all jobs are processed. The motivation for the method is minimizing the sum of tardiness.

3. Of all jobs allowed to be processed with regard to the chains, set up the job with the shortest setup time. Repeat until all jobs are processed. The motivation is, to get as many machines running as possible, such that the sum of the idle times is minimized.

4. Of all jobs allowed to be processed with regard to the chains, set up the job with the longest processing time. Repeat until all jobs are processed. The motivation is that the operator creates periods of time in which he has to set up fewer machines since more machines are still processing jobs and such the idle time for the waiting machines is reduced.

5. Of all jobs allowed to be processed with regard to the chains, set up the job from the chain with the largest sum of remaining setup and processing times. The motivation is to balance the finish times of the last jobs on the chains and such minimize the makespan.

We use a machine representation as a solution in contrast to the usual form of a job representation in the field of job scheduling. The sequence in the machine representation indicates to which machine the setup operator should move next. Given the sequence ‘3,2,2,1,4,5,1,2,3,1, this would mean first set up machine 3, then machine 2, and so on. Alternatively, the sequence ‘3,2,1,5,4,6,7,8,9,10 in a job representation would stand for setting up the machine which job three is dedicated to, then the machine which job two is dedicated to and so on. The length of the sequence is equal to the number of setups and jobs, respectively.
Our problem structure allows that each representation can be easily transferred to the other. Furthermore, the search procedures are also identical. The algorithm is, however, more efficient in the machine representation. That is because of the fixed order of jobs on the machines. In contrast to the machine representation, when the job representation is used, the algorithm must check for validity or repair the found solution, respectively.

Each sequence to be evaluated is subject to a post processing: Set up the jobs according to the sequence. If a setup operator has idle time and can only start the next job at time t, search in ascending order regarding the goal (i.e. next tardy job) for another job that can be set up before time t. If such a job is found, do the setup. Repeat until all jobs are processed.

4.3.1 Tabu Search

Tabu search was created by Glover [1986, 1989, 1990]. It is based on the methods of classical local search. In contrast, TS can overcome local optima, because it allows worse solutions than the current solution as the starting point for the next iteration.

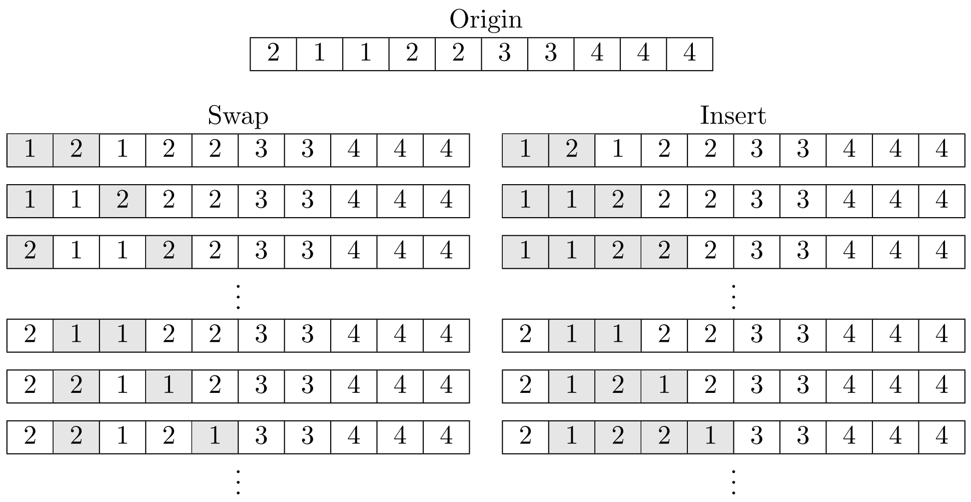
Outgoing from a solution x, a neighbourhood N(x) is created. This is done with the help of a classical neighbourhood search methods such as the Swap exchange. Each solution in N(x) is evaluated using a fitness function. The best feasible solution is taken as the starting solution for the next iteration. To prevent the algorithm from running in circles, the previous solution, part of the previous solution, or the transformation from the previous to the current solution is prohibited for a certain period (aspiration criteria) and put on the tabu list. The search is stopped when the stopping criteria is met (search time, number of iterations and so on).

Either the Swap exchange or the Insert method is used to create a new neighbourhood. The Swap method directly exchanges two numbers in a solution sequence, if they do not lead to the same schedule (e.g., exchanging three and three), while the Insert method puts a number a to a later position after a number b and moves all numbers that were previously between these two forward by one position. In both cases, the search space for a position in which to insert or into which to swap an element is limited to the next 15 positions due to time constraints. The tabu list consists of the 200 last used entire sequences. The type of the tabu list as well as the size were determined through test instances. Furthermore, a shake is used to randomly generate a new origin every 200 iterations. In the shake, two randomly chosen positions are exchanged with each other until 50% of all positions have been moved. The problem structure allows for the consideration of interim values of the fitness function. For all machines, the completion times are recorded for each position in the origin string. The newly built strings use the partial results while calculating their fitness value. With fitness value we refer to the value the created schedule gets receives regarding the objective function. In our case, this is always the sum of tardiness or the makespan. In a broad outline, the algorithm runs as follows:

```
while (gap > y) do {
  origin string = best string of the past iteration
  if (iteration == one of the predefined values to shake the origin string) do
  {
      shake origin string
  }
  for (i = 0 ; i < N ; i++) do {
      if (i + 15 ≤ N) do {
          j = i +15
      }
      else do {
          j = N
      }
      for ( x = i to j) do {
          use chosen exchange method on the origin string and create new
          string with parameters i and x
          post processing
          if (new string not in tabu list) do {
              calculate fitness function
              if the new solution is better than the best solution found in
              this iteration, and/or the best global solution, update the
              best solutions correspondingly
          }
      }
  }
}
```

To provide a better understanding, an example of the Swap and Insert exchange is given below:

Figure 4.1: TS Swap and Insert



For the given schedule "2,1,1,2,2,3,3,4,4,4", the Swap method exchanges the positions of the numbers directly. Starting with numbers 2 and 1, then 2 and 1 again, followed by 2 and 2 (which are not exchanged since both numbers are the same), the positions are exchanged, until the last position or the 15th position from the anterior exchange point in the schedule (number four) is reached. The anterior exchange point is then set forward to the second position, number one, and exchanges them with its neighbours. This is continued until the anterior exchange point has also moved forward to the end of the schedule. Hence, maximum 45 new schedules are derived for the Swap method in the example given.
The first exchange for the Insert is equal to the one for the Swap method. The second differs, as it can be seen. Number 2 is moved to the third position in the row, while both number 1s are pulled one position closer to the beginning. In the third new schedule, number 2 is put into the fourth position while number 1s and number 2 are pulled one position closer the front of the schedule. These exchanges are then continued until the last position or the fifteenth position from the anterior exchange point in the schedule is reached. Like the Swap exchange, the anterior exchange point is then set forward by one position to number 1 and the exchange is started again outgoing from the new anterior exchange point. The whole procedure is terminated when the anterior exchange point reaches the end of the schedule.

4.3.2 Genetic Algorithm

The Genetic Algorithm is a biased stochastic search [cf. Goldberg, 1989, pp. 10–11]. It was first proposed by Holland [1975]. It is related to the evolutionary search procedures. There might be slight differences in the notation [cf. Jiao *et al.*, 2007] or level of depth [cf. Blau *et al.*, 2004] regarding the different problem settings.

There are chromosome strings (or chromosomes) which represent a solution to a problem. Each part of the chromosome string is called a gene and represents a part of the solution. The characteristics of genes are shown by alleles. All chromosome strings together at a stage are in a so-called population.

An example of the mentioned notation can be derived from 4.3.2. In the first line the chromosome strings are shown in a so-called population. The chromosome strings have 10 genes at the most. The first chromosome string consists of genes 5, 4, 3, 1, etc. From the second chromosome string we take gene 2 as an example. Gene 2 consists of the following alleles: 5, 2, 4, 1, 3. Allele 2 is described as 1,0,1,1,0,0. The example stops here. But it is also possible that the characteristics of the alleles can be distinguished further, and so deeper levels exist. Thus, the more detailed the genetic algorithm is, the more levels arise. However, our example already has $10^{10} + 5^5 + 6^6 = 10000049781$ different probabilities of how a chromosome strings might look.

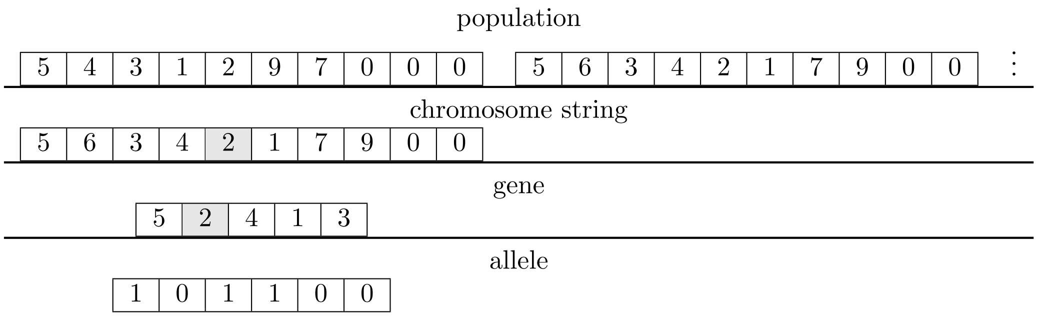


Figure 4.2: GA Structure

The basic procedure of a genetic algorithm is as follows: The first step is to form a population. Hence, different solutions get decoded into chromosome strings. This starting solution is also different from other methods. The GA makes use of multiple points and not a single one. [cf. Goldberg, 1989, pp. 7 - 9]. Each chromosome string is evaluated using a so-called fitness function.
In the next step, a new population has to be found. This is named the reproduction phase. A popular approach for reproduction is the roulette wheel strategy [cf. Goldberg, 1989, p. 11], which is also known as the Monte Carlo simulation. Each sequence gets an area of the roulette wheel assigned in relation to its fitness function. Thus, the better sequences have a higher chance of being selected. The problem with this approach is that convergence might apply quickly [cf. Baker, 1985]. If a chromosome string has a vast area in the roulette wheel, there will probably not be a diverse new population. Therefore, Baker

[1985] introduced the so-called ranking. All sequences are ranked according to their function. The highest rank gets the biggest area. The space gets assigned by a linear function, which means the space assigned does not increase or decrease between the sequences. It does not make a difference if one chromosome string makes up a high percentage of the whole sum of the fitness functions. Empirical tests by Baker [1985] have shown that although the ranking method is slower, the results are better because it does not lose genes. In the next step, the simulation is run until there are the same numbers of chromosome strings in the new population as there were in the old.
The stage of crossover follows. Two chromosome strings are randomly selected and the crossover is undertaken. There exist different variants of crossovers. Two, respectively three of the most common crossovers are explained in more detail at the end of this section. The crossover is repeated with the chromosome strings of the reproduced population until the new population is filled.
Finally, the new chromosome strings have to run through a mutation in order to make sure that the search does not end in a local optimum. Each gene is mutated with a specific probability, the mutation rate. The rate has to be very small otherwise the genetic algorithm becomes a pure random search [cf. Holland, 1992, p. 110]. When selected, the gene is altered at random. The mutations goal is to prevent the genetic algorithm being judged to quickly and therefore remaining in a local optimum. In the end, the chromosome strings are evaluated by the fitness function and the complete process starts again. This process is repeated until the stopping criterion is met. A common constraint is, for example, a pre-specified number of iterations [cf. Jiao *et al.*, 2007]. A disadvantage of this criterion is that an optimal solution might have already been found while the computer is still calculating. A moving average rule [cf. Balakrsihnan and Jacob, 1996] is a good fix in this case. The moving average rule stops the process when, for a certain number of iterations, the fitness function remains the same. The moving average rule, also known as the conversion rate, should be calculated for each problem individually using the sensitivity analysis in a trial example run. A tight conversion rate would let the computer run infinitely, whilst a very loose convergence rate would stop the process although a better solution could be found easily.

Three GAs are applied in our case. Our chromosome strings are the sequences, which represent solutions. Although there are a variety of crossover methods (see for example Soni and Kumar [2014] for an overview), we use the two most traditional variants, which are well known and have been extensively tested in other problem settings. We use two 2-point crossovers, with different repair and a 1-point crossover. The 1-point crossover was first introduced by Holland [1975]. The N-point crossover, which is a generalization of the 1-point crossover was introduced by Jong [1975]. The most basic form of the N-point crossover is the 2-point crossover.

The **2-point crossovers** select uniform randomly two crossover points on two arbitrary taken chromosome strings from the old population, where strings with a better fitness value have a higher probability of being selected and exchange the genes between these points with each other. The following formula regarding Baker is used to assign the selection probability to strings:

> Rank the strings according to their fitness value beginning with the worst string
>
> selection probability of string i $= \frac{\text{position in ranking for i}}{\sum\limits_{m=1}^{\text{number of strings}} m} \cdot 100.$

The new chromosome is created, from the first parent chromosome string until the first crossover point, the second parent chromosome string from the first crossover point until the second crossover point, and the first chromosome string from the second crossover point to its end. We create a second chromosome by exchanging the orders in which the parents are considered.
In the first case, the exchange of genes is done sequentially and each gene exchange is followed by a repair if necessary. In case a machine received in an exchange has no more jobs to be processed, then the dispensed machine has too few occurrences in the resulting string. A repair is necessary. Starting from the left, the resulting string is searched for a gene (machine) which has the same characteristic as the received gene. This gene is then changed to the characteristic of the dispensed gene.
In the second case, the repair is done after all genes between the two crossover points have been exchanged. Again, it is likely that in the resulting strings, some machines have too many occurrences, while others have too few. A repair is necessary. For each machine, it is controlled if the number of occurrences in the new chromosome string is equal to that in the parent string. Beginning with the first machine, the repair is done as follows: If machine q is underrepresented in the chromosome string, the first gene with the characteristic of an overrepresented machine beginning left in the new chromosome string, is changed to the characteristic of the underrepresented machine. If the machine is overrepresented in the chromosome string, the leftmost gene with the characteristic q, is changed to the characteristic of the next underrepresented machine. This is done until all machines are represented equally in the old and new chromosome strings. It should be noted that the two presented crossover methods are likely to have different outcomes due to their different repair mechanisms.
In the case of the **1-point crossover**, a crossover point is randomly picked, for two arbitrary chosen chromosome strings. Whereby, strings with a better fitness value have a higher probability of being selected from the old population. The same probability selection formula as in the 2-point crossover methods is also used here. All genes in the strings before the crossover point are then swapped with each other. The repair follows the exchange of all genes before the crossover point and is undertaken in such a way that the relative precedence of genes from the parent chromosome strings is also reflected in the new chromosome strings. Each new child string c of the parent p with the same postfix is considered. From left to right, each gene in the parent p is considered, and it is checked whether the machine q it represents is underrepresented in the child c. If it is,

the first gene of the child after the crossover point that has not been replaced during the repair, it is set to the characteristic of q.

To provide a better understanding an example is given below. See Table 2.1 as an example for the 2-point crossover variants.

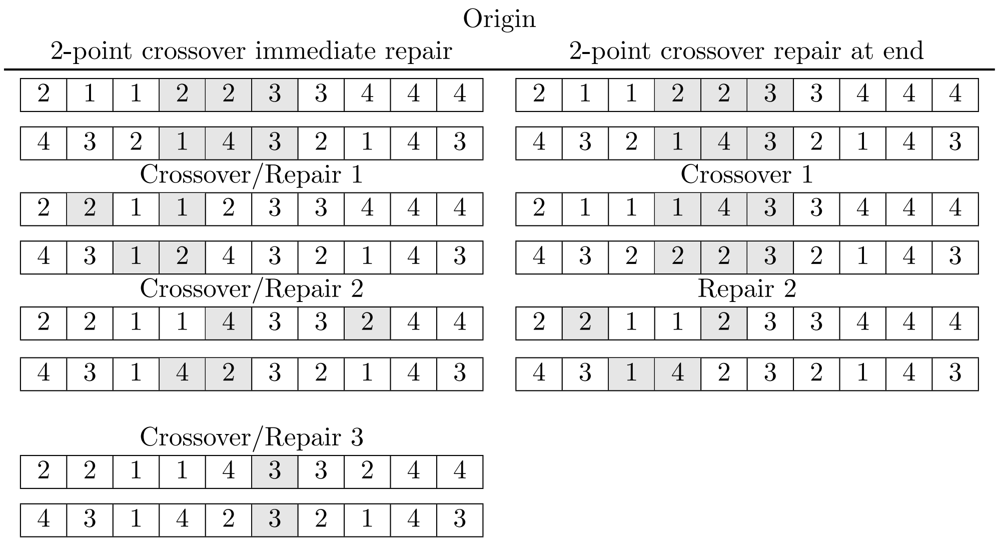


Figure 4.3: GA 2-point crossover

For the given schedule “2,1,1,2,2,3,3,4,4,4” and “4,3,2,1,4,3,2,1,4,3” the 2-point crossover immediate repair switches the positions between the crossover points (3–6) consecutively. Starting with the switching between 2 and 1 from position 3, the first position which has the same characteristic of the newly gained gene (1 and 2 respectively) is switched to the characteristic of the dispensed (2 and 1 respectively). This is continued until the second crossover point (3, position 6) is reached. In the 2-point crossover, all genes are first exchanged and then the repair follows. In the example, less repair is necessary. The repair follows the same rule as in the 2-point crossover immediate repair case. In the first string, two characteristics of 2 are missing, while one characteristic of 1 and 4 is too much. Therefore, the first genes with the characteristic 1 and another with 4 is changed to 2. In the second string, the problem is analogue except that the first genes with 2 are changed to 1 and 4.
In the 1-point crossover, the relative precedence of the parent strings is retained. In the example, all genes from the beginning until and including position 3 are exchanged. (2,1,1 and 1,4,4). Starting with the first position in the origin string the number of characteristics in the new string are checked. Gene 2 is used three times in the origin 1 string but only once in the new string. Therefore, the first position of the crossover point (position 4) is changed to the characteristic of 2. Position 2 in origin string 1 has characteristic 1. There are two genes with characteristic 1 in origin 1, but only one gene with 1 in new string 1. Therefore,

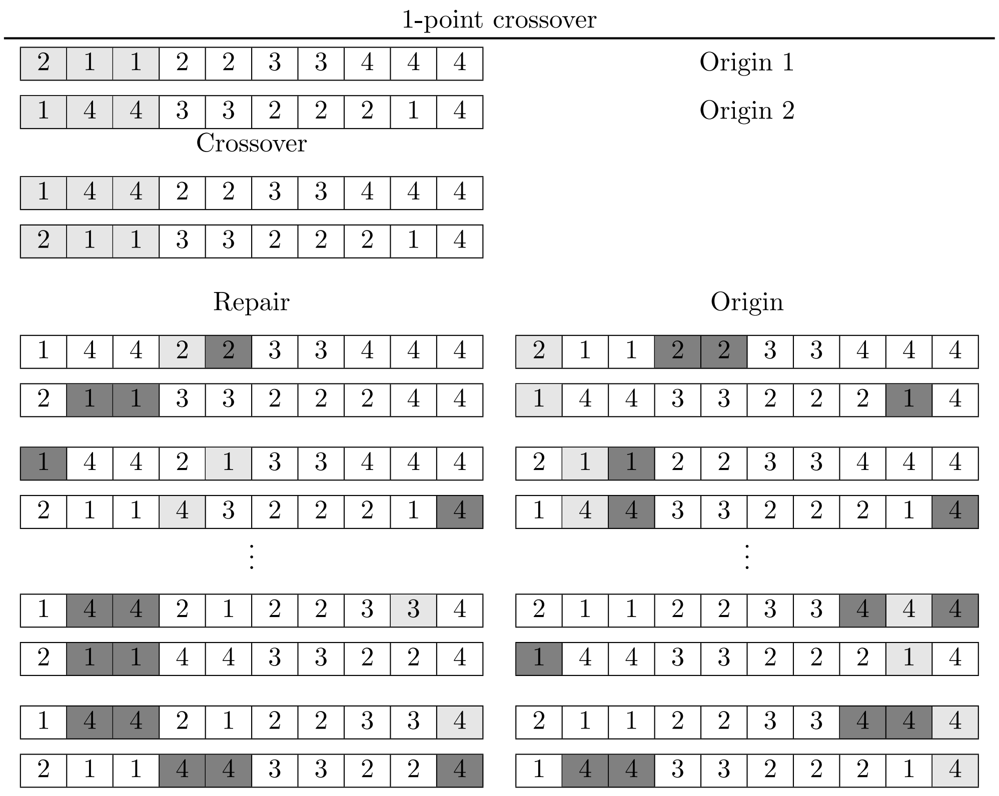


Figure 4.4: GA 1-point crossover

position 5 gets changed to value one. This procedure is followed until the last position (10) in the origin string when characteristic 4 is reached. The new string 2 is controlled and repaired analogously.

All three crossover variants share, however, the same type of mutation and elitism, as well as the same population size. The population size is fixed to 100, which was found to be a sufficient size regarding convergence and solution quality. In addition to the crossovers, each chromosome string in the new population is selected with a probability of one percent for mutation. If a string is selected, two genes in the string are chosen randomly and switched. Regarding elitism, the two best chromosome strings are automatically transferred from the old to the new population. Furthermore, all variants have in common that a new population is first created completely, before the new chromosome strings are subject to post processing and rated with the help of the fitness function. Therefore, there are no ‘intermediate populations containing strings partly from the old and the new population. The procedure can be described briefly as:

```
while (gap > y) do {
    while (size of new population < total population size - 2 ) do {
        pick 2 strings from the old population regarding their fitness value
        crossover and repair
        mutate each of the new chromosome strings with a predefined probability
        add the 2 new strings to the new population
    }
    add the 2 best chromosome strings of the last population to the new population
    post processing
    calculate fitness function
    sort the strings according to their fitness value
    if (best string of new population < so far found best global string)
    so far found best global string = best string of new population
    old population = new population
    clear new population
}
```

4.3.3 Simulated Annealing

Simulated annealing is an approximation algorithm developed by Kirkpatrick *et al.* [1983], used in mathematical optimization. The basic idea is to simulate the annealing process from metallurgy. After heating up metal, the slow annealing of the metal allows the atoms to sort themselves and create a stable structure, close to the optimum. Regarding the algorithm, the temperature is a probability of a worse solution to be accepted.

Similar to the TS, starting from a solution x a new neighbourhood N(x) is created. Either the Swap exchange or the Insert method is used to create a new neighbourhood. Since both methods have been extensively described in Subsection 4.3.1 no further explanation will be given here. In contrast to the TS, it is however one solution of the neighbourhood picked randomly and checked with the help of the probability function whether it will be accepted. The probability of a solution x' to be accepted p(x') can be described with the following formula: $(e^{-\frac{f(x)-f(x')}{t}}) \cdot 100$. $f(x)$ and $f(x')$ are thus the fitness values of the last and the recently calculated solution. T is a parameter that regulates the probability of the general acceptance of worse solutions in the SA. Each time a new solution is found, it gets updated by u, the so-called annealing factor by $t = t \cdot u$. The parameters u and t were obtained with the help of tests. While t is set to 1000 in our case, u is selected in such a way that after a 360-second running time the probability to select a worse solution is close to 0.

In general the algorithm runs as follows:

```
while (gap > y) do {
  create new neighbourhood outgoing from the last solution x
      randomly select a string $x'$ out of the neighbourhood of x
      post processing
      calculate fitness value of $x'$
      calculate $p(x')$
      randomly choose a number z between 0 and 100
      if ($z \leq p(x')$)
          $x = x'$
          $t = t \cdot u$
}
```

4.4 Computational results

The algorithms described in Section 4.4 are implemented in Java and tested on an Intel i7 Quadcore 3.4 GHz, 16 GB memory computer with the help of randomly generated instances (5.1), instances from which the best solution is known (5.2) and real-life data received from the company (5.3).

4.4.1 Random instances

The instances are created in such a way that they on the one hand reflect the production situation of the company which is in the focus in this chapter. The minimum and maximum setup times or the proportion between setup and production times are derived from this situation, for example. On the other side, the instances shall also show the performance of the algorithms when the size of the instances are a multiple of the instances with real data from the company (more machines or jobs for example). For each test case, 25 instances are solved with randomly generated numbers with regard to:

1. Setup times, which vary from 0 to 300 time units,
2. processing times, which vary from 0 to 3000 time units,

3. due dates for the jobs. The due dates are calculated as follows: Sum up all production and setup times of the jobs to a variable s. Multiply s with a positive factor greater than one, so that there is a chance that a schedule with zero tardiness is possible, to s*.[7] For each job, assign uniform randomly a due date between 0 and s*. Sort the jobs according to their due dates on the affiliated machines.

Furthermore, some cases consider:

4. Breaks, or more precisely setup operator availability times. Setup operators are available from Monday to Friday between 12:00 a.m. and 4 p.m. They have breaks from 4:00 to 4:30 a.m. and from 12:00 to 12:30 p.m.,

5. a distance between machines which the setup operator must cover. Machines are given randomly an x and y coordinate between 0 and 1000. The distance and the travel time for the setup operator between two machines $m1$ and $m2$ is respectively calculated by: $|x_{m1} - x_{m2}| + |y_{m1} - y_{m2}|$.

The computing time is limited to 360 seconds per instance and method. From 10 to 100 machines and 10 to 40 jobs on each machine (OEM) are considered in an instance (Tables 4.2, 4.5). A lower bound, calculated by an MIP relaxation, and a random search are used to compare the results of the methods described in Section 4.3. Cases 1–16 have the objective to minimize the makespan. The makespan is defined as the time between the start of the setup for the first job and the end of production for the last job including the time for breaks, if applicable. Cases 1–8 consider only the simplified problem, without breaks and distances between machines. Cases 9–16 consider all extensions. Cases 17–32 have the objective to minimize total tardiness. Cases 17–24 consider only the simplified problem, without breaks and distances between machines. Cases 25–32 consider all extensions as well.

Cases 1–16
What can be seen from cases 1–16 (Tables 4.3, 4.4) is that there is already a huge improvement in comparison to the starting solution (greedy algorithms) for all metaheuristic methods. The random search also shows some improvement in contrast to the starting solutions but gets clearly beaten by any metaheuristic method.

From Figures 4.5 (Diagram Case 2) to 4.8 (Diagram Case 14) it can be seen that the biggest improvement gains are already achieved before 60 seconds running time, depending upon the number of machines. Only in the biggest case in the diagrams, Figure 4.8 (Diagram Case 14), larger gains can be achieved after 60 seconds. One explanation could be that already with minor changes in the schedule, large improvements can be achieved. It has to be considered, however, that the TSs, as well as the SA, both of which will investigate a neighbourhood

[7] In our case we choose 1,1 as a factor.

Case	Machines	Jobs	Setup time	Processing time	Breaks	Machine distance
1	10	10 OEM	U(0,300)	U(0,3000)	/	/
2	10	20 OEM	U(0,300)	U(0,3000)	/	/
3	10	40 OEM	U(0,300)	U(0,3000)	/	/
4	25	10 OEM	U(0,300)	U(0,3000)	/	/
5	25	20 OEM	U(0,300)	U(0,3000)	/	/
6	50	10 OEM	U(0,300)	U(0,3000)	/	/
7	50	20 OEM	U(0,300)	U(0,3000)	/	/
8	100	10 OEM	U(0,300)	U(0,3000)	/	/
9	10	10 OEM	U(0,300)	U(0,3000)	yes	100·100
10	10	20 OEM	U(0,300)	U(0,3000)	yes	100·100
11	10	40 OEM	U(0,300)	U(0,3000)	yes	100·100
12	25	10 OEM	U(0,300)	U(0,3000)	yes	100·100
13	25	20 OEM	U(0,300)	U(0,3000)	yes	100·100
14	50	10 OEM	U(0,300)	U(0,3000)	yes	100·100
15	50	20 OEM	U(0,300)	U(0,3000)	yes	100·100
16	100	10 OEM	U(0,300)	U(0,3000)	yes	100·100

Table 4.2: Random instances makespan 1–16

very closely by their structure, are beaten in all but one case by the GAs, which investigates a neighbourhood in a broader sense. Besides, as can be seen in Figure 4.8 (Diagram Case 14), the GAs get their first gains much faster than the TSs and the SA. Therefore, this explanation does not hold in general. Another explanation might be that the neighbourhood structure in case 4.8 (Diagram Case 14) challenges the TSs and SA harder than the other cases from the diagrams in such a way that it needs to enter a new neighbourhood to achieve the gains. This would mean that there is already a different level of challenges for the algorithms within this problem setting.

The VNS shows as expected the best overall results of all metaheuristics, thereby indicating that it can take advantage of the benefits of each metaheuristic. In six of the eight cases, it is better than the individual methods alone when no breaks and travel times are incorporated. When breaks and travel times are incorporated in four out of the six valid cases, the VNS performs best.

While there is, in the case of 10 machines, only a minor gap to the lower bound, in all other cases, there is still a huge gap, especially when breaks and travel times are incorporated. In two cases (11 and 15), no lower bounds were found within acceptable time (360 seconds). Therefore, it is difficult to validate the overall performance of all the metaheuristics, although they bring huge improvements in contrast to the greedy algorithms.

	Starting solution	GA 2 point V1	GA 2 point V2	GA 1 point
1	149,83%	1,48%	0,95%	0,92%
2	173,19%	2,55%	2,15%	2,17%
3	190,20%	4,70%	4,27%	4,39%
4	315,80%	72,80%	72,76%	72,92%
5	349,30%	85,42%	85,38%	85,97%
6	517,76%	232,81%	233,158%	234,76%
7	569,45%	257,81%	257,76%	258,22%
8	883,93%	550,06%	549,65%	550,68%
9	235,79%	102,32%	101,24%	100,23%
10	287,93%	141,91%	139,79%	140,44%
11	XXX%	XXX%	XXX%	XXX%
12	553,73%	390,93%	390,40%	377,91%
13	676,19%	500,79%	498,93%	492,12%
14	1046,88%	867,44%	863,93%	864,22%
15	XXX%	XXX%	XXX%	XXX%
16	2034,10%	1836,45%	1832,76%	1831,21%

Table 4.3: Gap to the lower bound for cases 1–16, methods 1–3 and starting solution

	TS Swap	TS Insert	SA	VNS	Random Search
1	1,64%	1,96%	3,93%	0,60%	68,86%
2	4,36%	4,92%	4,84%	1,89%	106,81%
3	7,41%	7,34%	5,60%	3,36%	141,31%
4	82,48%	84,59%	81,88%	72,72%	202,77%
5	92,48%	92,77%	93,46%	88,12%	405,08%
6	235,30%	235,39%	237,20%	229,85%	391,08%
7	259,46%	260,86%	265,18%	261,82%	479,14%
8	551,90%	553,50%	545,89%	540,91%	733,64%
9	102,66%	106,93%	115,34%	98,07%	145,22%
10	145,87%	148,38%	154,94%	141,98%	211,95%
11	XXX%	XXX%	XXX%	XXX%	XXX%
12	409,16%	410,50%	411,31%	382,65%	447,54%
13	515,07%	521,77%	522,22%	495,92%	593,54%
14	901,53%	905,69%	917,79%	872,03%	938,87%
15	XXX%	XXX%	XXX%	XXX%	XXX%
16	1883,96%	1884,98%	1874,84%	1830,20%	1904,66%

Table 4.4: Gap to the lower bound for cases 1–16, methods 4–8

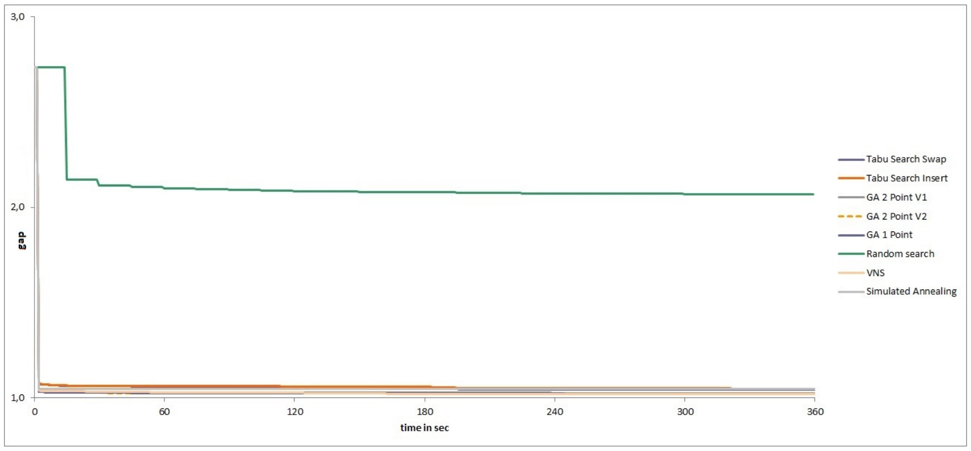


Figure 4.5: Diagram Case 2

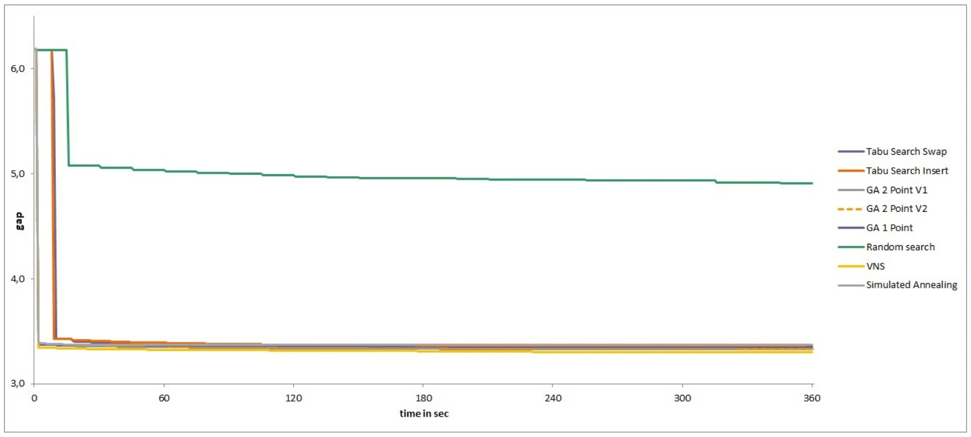


Figure 4.6: Diagram Case 6

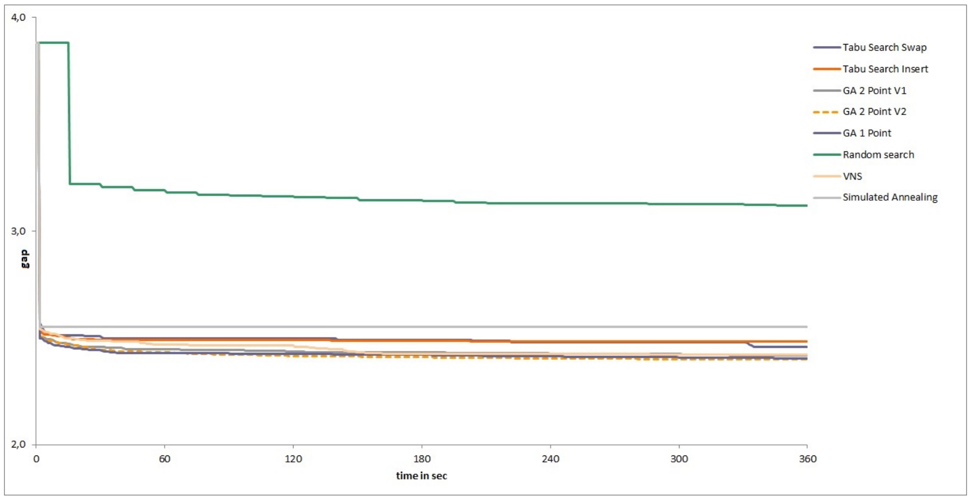


Figure 4.7: Diagram Case 10

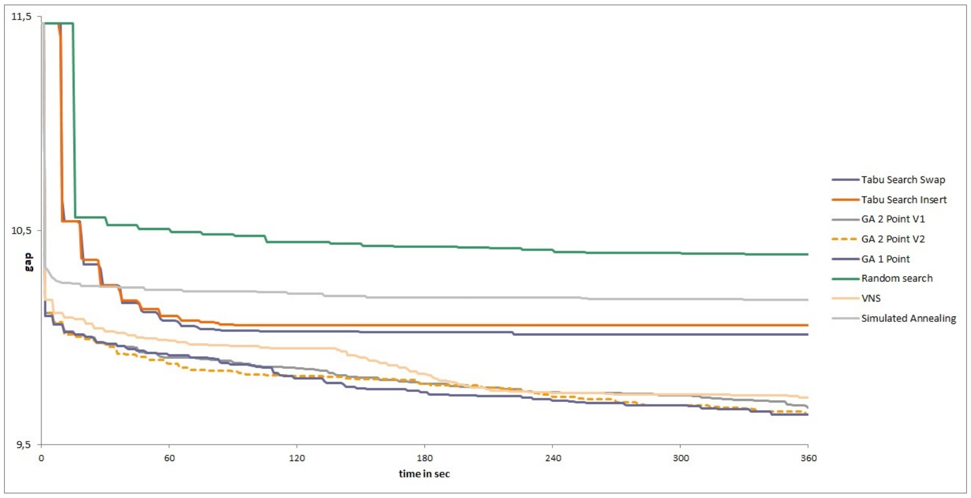


Figure 4.8: Diagram Case 14

Case	Machines	Jobs	Setup time	Processing time	Breaks	Machine dis-tance
17	10	10 OEM	U(0,300)	U(0,3000)	/	/
18	10	20 OEM	U(0,300)	U(0,3000)	/	/
19	10	40 OEM	U(0,300)	U(0,3000)	/	/
20	25	10 OEM	U(0,300)	U(0,3000)	/	/
21	25	20 OEM	U(0,300)	U(0,3000)	/	/
22	50	10 OEM	U(0,300)	U(0,3000)	/	/
23	50	20 OEM	U(0,300)	U(0,3000)	/	/
24	100	10 OEM	U(0,300)	U(0,3000)	/	/
25	10	10 OEM	U(0,300)	U(0,3000)	yes	$100 \cdot 100$
26	10	20 OEM	U(0,300)	U(0,3000)	yes	$100 \cdot 100$
27	10	40 OEM	U(0,300)	U(0,3000)	yes	$100 \cdot 100$
28	25	10 OEM	U(0,300)	U(0,3000)	yes	$100 \cdot 100$
29	25	20 OEM	U(0,300)	U(0,3000)	yes	$100 \cdot 100$
30	50	10 OEM	U(0,300)	U(0,3000)	yes	$100 \cdot 100$
31	50	20 OEM	U(0,300)	U(0,3000)	yes	$100 \cdot 100$
32	100	10 OEM	U(0,300)	U(0,3000)	yes	$100 \cdot 100$

Table 4.5: Random instances total tardiness 17–32

Cases 17 - 32
In contrast to cases 1–16, where the goal was to minimize the makespan, in cases 17–32 (tables 4.6, 4.7) with the aim to minimize total tardiness, the greedy heuristics for the starting solutions performed much better. Overall, in 135 of the 400 calculated instances, they found the best solution immediately.

With respect to the GAs, the 1-point crossover dominates the 2-point crossovers. It is in no case worse but in eight cases better than the best solution of the 2-point crossover methods. The 1-point crossover is further dominated by the TS in all cases. This might indicate that the starting solution is already in a good neighbourhood for further improvements. When comparing the two TSs, especially the Swap method showed promising results. It performed only worse in one case than the Insert method, but in 14 cases it was better. The SA provides only average results. While it is better in the larger instances than the GAs, it performs worse in the smaller instances. This is especially noteworthy, as the SA also uses the Swap exchange, which provided the best results in the TS. Therefore, the explanation about the starting solution in a good neighbourhood can be questioned.

From Figures 4.9 (Diagram Case 18) to 4.12 (Diagram Case 30) it can be seen, that all metaheuristics find better solutions very fast. The first gains for the TSs are overall bigger than for the other methods, which is in line with the theory that the starting solution is in a good neighbourhood. On the other hand, it can clearly be seen that the SA has less gains in the beginning than the GAs.

	Starting solution	1,0	GA 2 point V1	GA 2 point V2	GA 1 point
17	402,08%	7	140,63%	108,75%	94,18%
18	3059,45%	9	372,89%	312,09%	254,15%
19	896,50%	7	451,15%	399,86%	389,17%
20	71,44%	6	60,22%	60,22%	60,22%
21	91,14%	6	65,84%	65,84%	65,84%
22	116,44%	6	78,06%	78,06%	78,06%
23	49,66%	7	36,31%	36,31%	36,31%
24	215,63%	8	129,00%	129,00%	129,00%
25	422,65%	7	261,63%	239,94%	224,55%
26	506,35%	7	254,73%	220,90%	179,76%
27	367,19%	7	315,16%	313,55%	295,07%
28	103,19%	10	103,19%	103,19%	102,16%
29	85,18%	13	85,18%	85,18%	85,04%
30	37,42%	10	37,42%	37,42%	37,42%
31	496,35%	11	496,35%	496,35%	496,35%
32	32,86%	14	32,86%	32,86%	32,86%

Table 4.6: Gap to the lower bound for cases 17–32, methods 1–3 and starting solution

	TS Swap	TS Insert	SA	VNS	Random Search
17	21,60%	32,90%	245,83%	28,68%	217,29%
18	83,29%	83,62%	407,06%	96,98%	1820,72%
19	30,62%	37,48%	527,39%	29,40%	896,50%
20	19,65%	23,18%	60,22%	19,59%	71,44%
21	22,96%	25,17%	65,57%	22,96%	91,14%
22	25,04%	25,91%	78,06%	25,07%	116,44%
23	23,20%	23,57%	36,31%	23,20%	49,66%
24	71,92%	74,11%	126,55%	78,05%	215,63%
25	91,11%	96,86%	267,10%	33,68%	422,65%
26	25,80%	48,05%	254,71%	28,15%	506,35%
27	32,89%	40,00%	158,58%	22,82%	367,19%
28	43,40%	43,40%	61,87%	21,94%	103,19%
29	23,23%	22,43%	57,93%	8,16%	85,18%
30	17,32%	18,32%	21,55%	12,25%	37,42%
31	58,14%	90,43%	496,35%	24,23%	496,35%
32	15,92%	20,42%	31,55%	15,26%	32,86%

Table 4.7: Gap to the lower bound for cases 17–32, methods 4–8

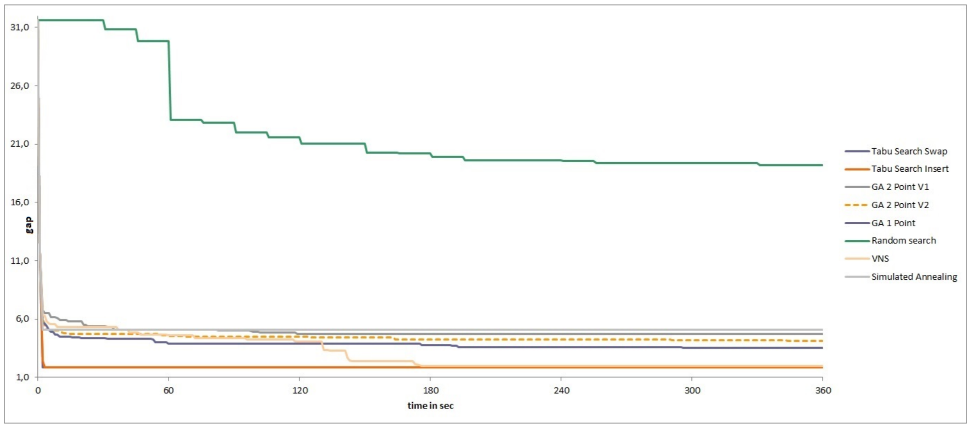


Figure 4.9: Diagram Case 18

So, for the moment, there is no conclusion regarding the explanation of different performances of the metaheuristics.

The VNS showed again an exceptionally good performance. Overall it performed as the best method in nine cases. In the other cases it was close to the best-found valid solution. The path of the VNS from Figures 4.9 (Diagram Case 18) to 4.12 (Diagram Case 30) is interesting though. It first follows the performance of the GAs and then switches for further improvements to the TSs. Before the switch, the performance is worse than the performance of the TSs. The switch allows the VNS, however, to get overall a better performance than the TSs. Therefore, it indicates that a neighbourhood exists which was found by the GAs and which allows an overall better performance when analysed in detail than the starting neighbourhood, which is more intensively explored by the TSs.

It has to be acknowledged though, that in 25 of the 400 test cases the lower bound indicated that no tardiness exists, while the best-found starting solution showed a tardiness. It remains unclear how these cases can be compared. They are so far not included in Table 4.6 and Table 4.7.

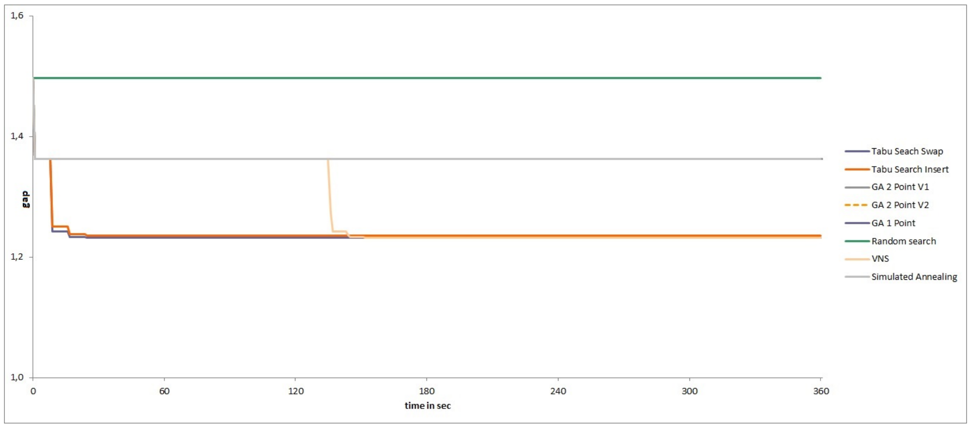


Figure 4.10: Diagram Case 22

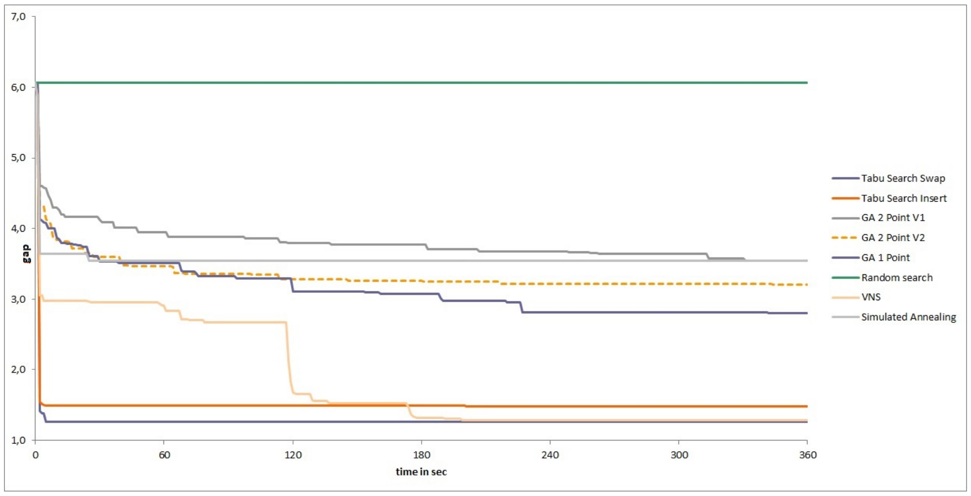


Figure 4.11: Diagram Case 26

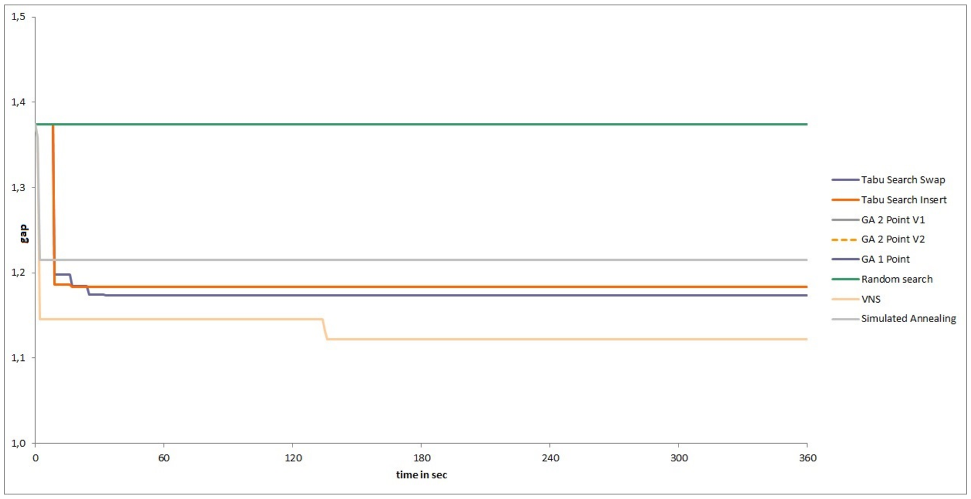


Figure 4.12: Diagram Case 30

4.4.2 Instances with known best solution

We tried different variants to create lower bounds, when we were not able to solve the MIPs because we ran out of memory. On the one hand, we started by relaxing some integer and binary variables and/or eliminating constraints. "CPLEX" and "Gurobi" were, however, still not able to solve these relaxed problems. Only when all integers were relaxed, the problem was solvable by standard solvers. But we started to calculate lower bounds manually, by summing up the setup and break times, and adding the smallest production times of the jobs to the machines. These lower bounds were worse than relaxing all integers in the MIP and using the solution of these as the lower bounds. Nevertheless, the results for these lower bounds were rather weak as could be seen in 4.4.1. Therefore, we also created instances in such a way that the best solution is known beforehand. It is clear that there are different ways to create a schedule from which the best solution is known beforehand. The most obvious way is that all machines and the setup operator have a workload of 100% until a certain point in time t. There is no other schedule that finishes before t. Nevertheless, this is not a realistic schedule regarding the data we have received from the company. Therefore, we tried to make the schedule as realistic as possible, while still maintaining the knowledge at which point in time the optimal schedule ends. For us, this means that the setup operator as well as the machines both have idle time in the optimal schedule.

1. For the time period 0 to t a sequence for the setup of jobs is randomly chosen. This sequence also implies the assignment of jobs to machines. Two consecutive setups on the same machine are not allowed. The setups of the jobs are selected in such a way that the setup operator has

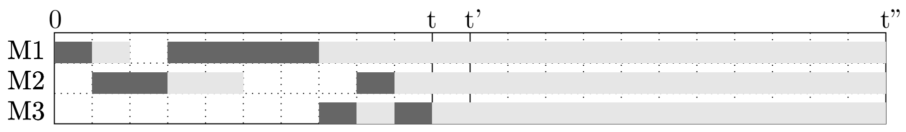


Figure 4.13: Generation of instances with best solution known parallel dedicated machines: Step 1

a workload of 100%. Therefore, the earliest possible point in time, when the setup operator is finished, is t. (Figure 4.13)

2. On each machine, after the final setup, a processing time amounting to the difference between t' and the end of the final setup on the machine is inserted. A lower bound for the last completion time for all jobs is the point when the last setup is finished (t) plus the minimum processing time over all final jobs on the machines. This happens at time t'. Therefore, our proposed schedule which finishes also at t' is optimal. (Figure 4.13)

3. Between two jobs on a machine, a randomly chosen processing time with the maximum duration of the distance between the two setups is introduced. Until point t, the setup operator still has a workload of 100%. Hence, the argumentation from point 2 still holds. Therefore, the schedule is still optimal. (Figure 4.14)

4. The final processing times on the machines is increased from t' to t". The argumentation from point 2 can be used analogously. Therefore, the schedule is still optimal. (Figure 4.14)

5. The processing times between t' and t" are interrupted through the insertion of setup times, which therefore split the last jobs into smaller jobs. The setup times are chosen in such a way that the machines still have a 100% workload between the last setup before t on each machine and t". The setup operator still has a full workload between 0 and t. If the machines have a full workload after their last setup before t, a lower bound is the point in time when the last setup is finished before t, plus the minimum amount of work a machine has to do after the final setup before t. This point is reached at t". The proposed schedule ends at t". Hence, it is optimal. (Figure 4.14)

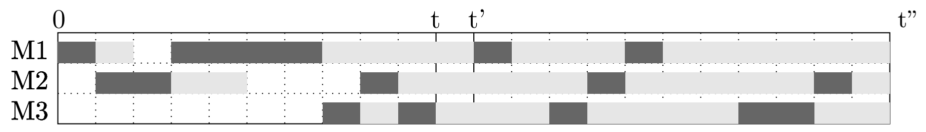


Figure 4.14: Generation of instances with best solution known parallel dedicated machines: Step 2

For each case, 25 instances are solved.

Case	Machines	Jobs
33	10	10 OEM
34	10	20 OEM
35	10	40 OEM
36	25	10 OEM
37	25	20 OEM
38	50	10 OEM
39	50	20 OEM
40	100	10 OEM

Table 4.8: Instances from which the best solution is known 33–40

Cases 33–40
What can be seen clearly from tables 4.9 and 4.10 is the huge gap to the lower bound for the greedy heuristics. Even in the smallest cases, the gap is still on

	Starting solution	GA 2 point V1	GA 2 point V2	GA 1 point
33	188,50%	0,03%	0,00%	0,00%
34	193,70%	0,02%	0,01%	0,03%
35	195,46%	0,03%	0,03%	0,05%
36	380,83%	0,20%	0,11%	0,13%
37	356,78%	0,09%	0,08%	0,09%
38	536,59%	0,67%	0,59%	0,59%
39	564,16%	0,67%	0,66%	0,70%
40	805,75%	2,02%	1,98%	2,01%

Table 4.9: Gap to the optimum for cases 33–40, methods 1–3 and starting solution

	TS Swap	TS Insert	SA	VNS	Random Search
33	0,08%	1,09%	1,64%	0,00%	75,05%
34	1,13%	1,25%	1,25%	0,02%	107,96%
35	0,99%	0,98%	0,80%	0,03%	140,08%
36	1,68%	2,07%	2,14%	0,23%	206,09%
37	1,47%	1,61%	1,51%	0,28%	258,86%
38	2,43%	3,45%	3,19%	1,32%	371,39%
39	2,24%	2,34%	2,12%	0,87%	450,65%
40	3,90%	4,10%	3,66%	2,31%	621,52%

Table 4.10: Gap to the optimum for cases 33–40, methods 4–8

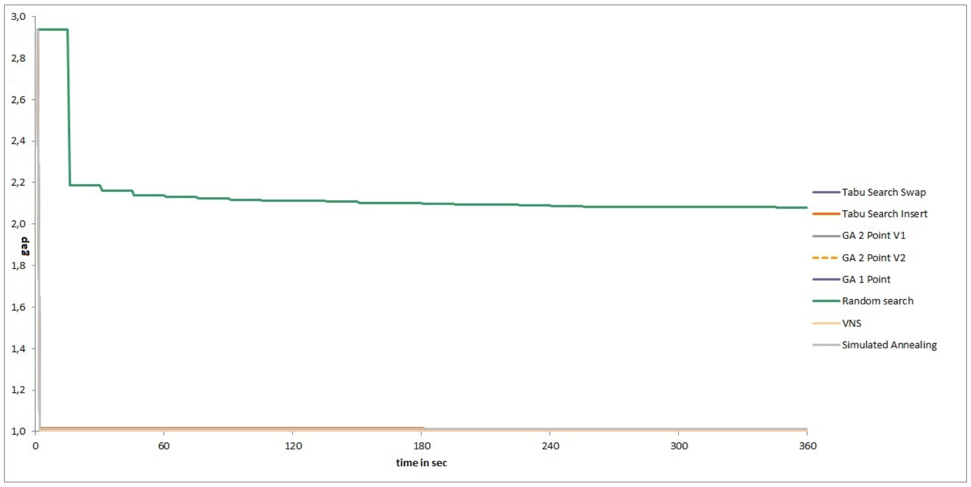


Figure 4.15: Diagram Case 34

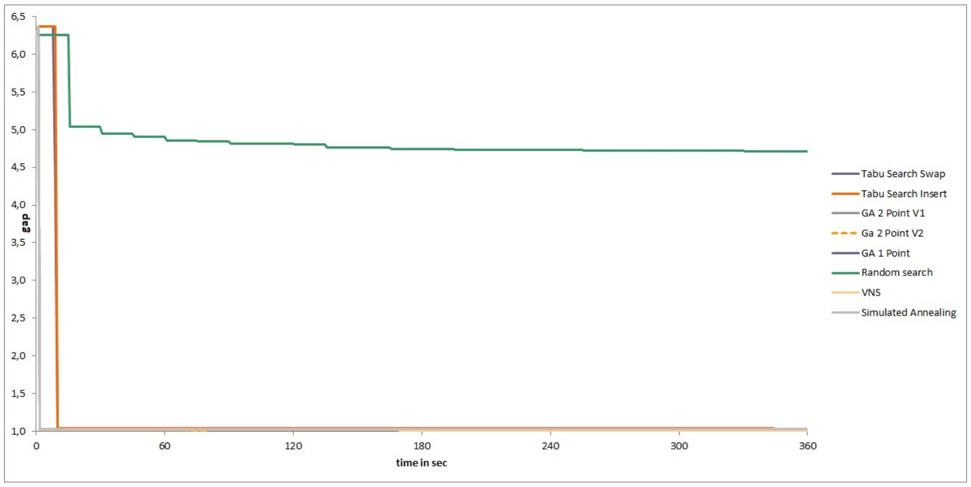


Figure 4.16: Diagram Case 38

average of 188, 5%. In the largest instances, the gap can get as high as 805, 75% on average. The metaheuristics are on the other side very close to the optimum after 360 seconds. No metaheuristic is more than 4, 1% away from it. Even more, Figure 4.15 (Diagram Case 34) and 4.16 (Diagram Case 38) show in an exemplary manner that all metaheuristics come very close to the optimum in a very short time frame. This is an interesting result, as the cases were made in such a way that they were thought to be hard to solve, as they are very dense.

Although the neighbourhood search methods (TSs, and SA) show a satisfactory performance, they are always further away from the optimum than the GAs. The SA is still the weakest metaheuristic. The TSs are in line with results 1–16 and the GAs are best. They are in all but one case less than one per cent away from the optimum on average. Therefore, it seems clear that the choice of goal for optimization has an influence on the performance of the metaheuristic.

The performance of the VNS is, however, worse than the performance of the GAs. Although it is closer to the results of the GAs than to the other methods, or sometimes gets the same results as the GAs, it is worse than the GAs on average. Therefore, the stage of TS or SA, with a detailed view on the neighbourhood it takes from the GAs, does not bring up sufficient results, as in cases 1–16. This could either be that the GA search is not finished yet and thus the other methods have to start in a not-so-good neighbourhood, or the structure in the neighbourhood brought up by the GAs, cannot be efficiently investigated by the TSs and SA. Hence, while the objective function, number of jobs, machines is comparable from cases 1–16 to cases 33–40, the underlying problem structure must be different. Further research must be done.

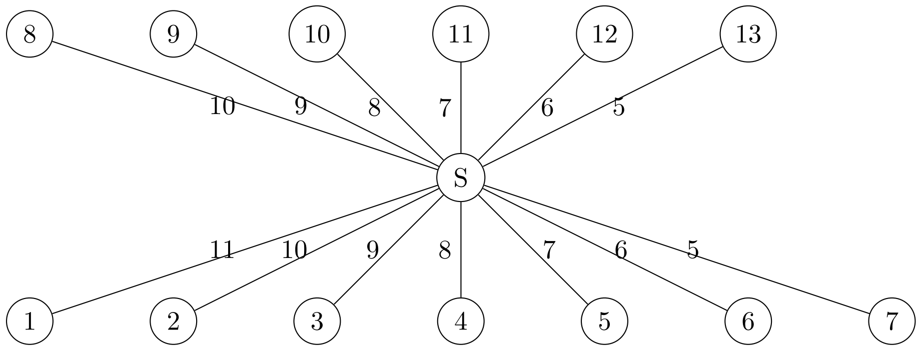


Figure 4.17: Travel distances at the company

4.4.3 Case study

As the problem has already been described in the introduction, the information provided below focuses on detailed production data from the company. For our calculations we use ten old, real production instances, which are referred to as case 41. The planning period covers the next four weeks in each instance, where, on average, 144 jobs need to be produced on 13 machines. The minimum number of jobs to be scheduled is 114 while the maximum is 196. The objective is to minimize total tardiness. Since the company has a large product portfolio and only the customer related quantity is produced, there are significant differences in the production times for the jobs. In the data we received from the company, the shortest production time for a job is 62 minutes, the largest production time is 4167 minutes while the average production time is 2103 minutes and the average median of all production times is at 2088 minutes.
The shortest setup time for a job, in the data provided by the company, is 60 minutes, the longest 180 minutes, while the average setup time is 120 minutes and the median of all setup times is also 120 minutes.
The travel time between two machines, Figure 4.17, consists of picking up the tooling at one machine, bringing it to the tooling keeping location, picking up new tooling and then going to the new machines. Although machines are arranged in two rows, the position of the tooling keeping location at the end of one row increases overall travel time. Therefore, the travel times are between 10 and 22 minutes. Furthermore, setup operators work in shifts with breaks and the goal is to minimize total tardiness, as described in Section 4.1.

Since the goal in this subchapter is to identify the possible production improvements for the company if more advanced heuristics would be utilized, we use their due date heuristic to find a starting solution and compare it to the VNS, which shows the best and most steady performance. The first comparison is the average makespan. In the VNS the makespan is on average 56656 minutes. In the due date heuristic, the makespan is on average 339934 minutes.[8] The total production output, i.e. the number of finished jobs per day is therefore also higher for the VNS (3,66 in the VNS to 0,61 jobs in the due date heuristic). The higher production output is due to reduced waiting time for the setup operators and for the machines. In the due date schedule setup operators wait 87,74% and machines 93,27% of their time while in the VNS schedule the waiting times are only 34,39% for setup operators and 61,68% for the machines. Hence, it can be said that the company can achieve significant efficiency gains by using metaheuristics. Although we did not test the performance of the other algorithms, in this context, which is close to the random cases with the goal of minimizing total tardiness, we would expect that all metaheuristics increase the solution quality in contrast to the due date heuristic. Nevertheless, in line with Subsection 4.4.1 we expect the TSs to perform best, even slightly better than the VNS, which is used. Especially the SWAP exchange is sought to provide additional solution quality. The GAs as well as the SA are expected to provide worse solutions than the VNS in this case.

4.5 Conclusion

The aim of this chapter is to extend the parallel dedicated machine scheduling problem with the scarce resource setup operators. With this goal in mind, a formal problem definition in Section 4.2. was given. Here the operation time of a job was split into setup and production time. The setup can only be undertaken if the setup operator has ample capacity. Moreover, in the model, travel times between the machines and break times, and shift times for the setup operator were also included.

Seven different methods were developed to tackle the problem (three GAs, two TSs, one SA, and one VNS) in 4.3. Section 4.4. then brings up a broad computational study in which the methods were tested. Different instances with the objective either to minimize total tardiness or makespan were used here.

Randomly generated instances with up to 100 machines and 1000 jobs were tested with and without break and travel times between the machines in 4.4.1.

If the goal was to minimize makespan with random instances, the VNS performed best. In eight of the 14 cases, precisely in six out eight cases without breaks and travel time, and two of six cases with breaks and travel time, the VNS performed best. In the other cases, the VNS was also close to the best-found solution. After that, the GAs also showed a satisfactory performance and delivered the best-found solution in some cases. TSs and SA performed relatively weak and left a bigger gap to the best-found solution.

Also in the minimize total tardiness problem with random instances the VNS

[8]Note that the makespan is so large due to the break and travel times.

performed very well. It delivered in nine out of the sixteen cases the best solution after 360 seconds. The TSs also provided good results. To our surprise, the GAs still had a huge gap at the lower bound compared with the TSs and the VNS.

Since we provided rather weak lower bounds, we created in 4.4.2 instances from which the optimal solution was known. Due to the structure of the instances, only the makespan minimization goal was pursued. In contrast to cases 1–16, the GAs outperformed all other methods. The VNS, by contrast, was able to find the same solution quality as the GAs in two cases and is close in the other six cases. The TSs as well as the SA still had a larger gap from the optimal solution.

Finally, we tested the VNS, as it provided the steadiest performance of the metaheuristics so far, in Subsection 4.4.3 on real data provided by the company. It could clearly be shown that the use of the VNS would deliver huge savings to the company.

Section	Type	Target	Density	Performance
Parallel dedicated machines	Random instances	Makespan no breaks	low	GAs best, TS good
		Tardiness no breaks	low	TSs better than rest
		Makespan breaks	low	bigger gap with breaks rest same
		Tardiness breaks	low	GAs have problems
	Best solution known	Makespan	high	all metaheuristics have strong performance

Table 4.11: Results 1

Comparing the test scenarios so far, it can be said clearly that the performance of the methods is dependent upon the problem structure. Nevertheless, any metaheuristic was much better than the currently used sorting heuristics in this area to generate a plan. The VNS search was in all of the scenarios close to the best solution and performed especially well in the random cases 1–16. As the work/performance of the VNS is dependent on the work/performance of the underlying metaheuristics, it is not included in Table 4.11, as the target is to identify influencing factors on the performance of the different algorithms. With column 5 density, we refer to whether the structure of the optimal solution is thought to be very dense. Although the setup operator as well as the machines have idle time in the best solution known case the machines have maximum workload in the first part of the schedule and in the second part the schedule, the setup operator has maximum workload. Therefore, this is seen to be a dense

schedule. We also expect a schedule to be denser with a higher number of setup operators, as they are clearly a limiting factor. Hence, the higher number of setup operators in a schedule, the less the amount of overall idle time.
The most obvious result is in the best solution known case (makespan). All metaheuristics show a robust performance. There is no difference in the number of jobs. Therefore, it is estimated that density has a positive influence on the performance of all metaheuristics. Also, in the random instances makespan cases, the results are good for all metaheuristics. Although they are closer to the lower bound in the smaller instances, the gain they can achieve is bigger in the larger instances. Therefore, there is no advantage for smaller instances. The GAs are slightly better than the TSs. In the tardiness random instances cases, the TSs are better than the GAs. Hence, a first expectation would be that the TSs work better with the goal to minimize total tardiness. When breaks are included, the gap to the lower bound overall increases for all metaheuristics. The GAs seem especially to have problems with breaks and the goal total tardiness. So, this combination seems to have a negative influence on the GAs. No interesting facts were found about the SA.

Overall, the chapter provides first insights into the so far unknown relevance of setup operators in the area of production planning. From a practical point of view, the results are of special importance to production companies and have huge cost-saving potential. As shown in the results, huge savings can be gained by the use of more advanced algorithms. On the other side, as seen in Section 2.3., many SMEs lack basic software infrastructure to use these advanced algorithms. Therefore, an introduction is only the second step for them. In these cases, the primary recommendation is to increase awareness that setup operators are a limiting factor and to ensure that a sufficient number of setup operators is always available. This can be done through the advanced training of operators for example such that they are able to do both jobs. This chapter also contributes to the sparse literature on setup operator scheduling. Nevertheless, while old questions have been answered, new questions arose. What remains unclear is the different performance of the metaheuristics regarding the problem structure. Some focus should therefore be clearly put on identifying better lower bounds. On the other hand, the influence of the different constraints, targets could be checked in a computational study where all combinations of targets and constraints are tested. Moreover, we have only tested these types of algorithms on the very distinct case of parallel dedicated machines with one setup operator. Their performance in more general problems remains unclear so far.

Chapter 5

Parallel machines subject to setup constraints

5.1 Introduction

The problem we consider in this chapter is based on the production situation for synthetic material at a medium sized company. While the company analysed in this chapter is above the limits of the quantitative definition of the IfM [2002] (it has around 700 employees) it is still family owned in second generation. Different members of the family are responsible for different tasks in the company such as sales, production or human resources. The company also uses universal machines which can produce a variety of products. Therefore, they also have the problem of labour intensity Pfohl [2006]. There is an ERP system, but it does not support PPS functionality. The planning is done completely manually with Excel. Stock levels are so high that sometimes also the parking spaces for employees are used to store parts. Hence, the company is perfectly in line with the answers from our study, and thus the results are expected to be applicable to most other SMEs in the field of production. The mathematical problem can be seen as the proceeding of the parallel dedicated machines problem with a single setup operator from Chapter 4. The difference now is that there is no job-chain on the machines, jobs are not preassigned to machines and instead of one setup operator, there are multiple setup operators. We focus on their plant at the headquarter with 10 machines.
Before a job can be produced on a machine, the machine has to be set up for the job. A setup time occurs in the following cases:

1. A **change of dimension** is needed when the succeeding product has a diameter that is different from its predecessors. The time for the change is sequence-dependent. For the products with the largest diameters, the time to change the dimension can take up to 20 hours.

2. If a successor product follows immediately on the same machine and has

a different colour from that of the predecessor product, a **colour change** setup must be done. The machine must be cleaned of the leftovers of the old base material. The base material for the successor product then gets filled into the machine. Depending on the sequence of the products, the colour change can take up to 36 hours.

3. If the successor does not follow immediately, the machine must still be cleaned and shut down afterwards. We will refer to this as **dismounting** as was done by Cheng and Sriskandarajah [1999]. Dismounting can take up to eight hours.

4. If a job has no direct predecessor, a **start-up time** occurs. This is the time for heating up the machine and setting it up with the correct tooling. The start-up time is machine-dependent and can take up to 20 hours.

Setup and dismounting operations, as well as machine start, must be done by a setup operator. This dedicated person has the necessary skills and tools for setting up and dismounting the machines. Setup operators work in three shifts, seven days a week. At each point in time, three setup operators are available, meaning that three setup operations can be done in parallel.
The company uses the following scheduling strategy: New production jobs are added every Monday to the plan. They are added in such a way that the start and end times of the jobs in the existing plan do not get affected. New jobs are sorted by their delivery date and planned behind the already scheduled jobs as early as possible in the plan. The plan covers the next four weeks. There is no planned schedule for the setup operator himself. Instead, he sets up the next available job. If several jobs can be chosen, he sets up the job which was available first.
For the company, the most important goal is the punctual delivery of customer orders. Although, as of today, their punctuality is close to 100%, future requirements of their customers will affect their production schedule. On one hand, the minimum order level will diminish. More jobs will have to be produced and scheduled. The proportion of setup time as opposed to production time will increase. Besides, the company has a growth of 10% annually, which is why the number of jobs to be scheduled and produced also increases. Therefore, manual scheduling is likely to reach its limitations. Computerized scheduling, taking the above-mentioned restrictions into account, might provide better results, than the strategy currently applied.

5.2 Problem definition

In a more formal way, the problem can be described as follows: We consider a set of machines $m = 1, \dots, M$. There are jobs $j = 1, \dots, J$. Each job j requires processing time $p_{j,m}$, which depends on the machine the job is assigned to, and is preceded by sequence-dependent setups of duration $s_{l,j}$. If no job follows in d time units on the same machine after the direct preceding job has been finished, a job and machine-dependent dismounting $b_{j,m}$ occur. After the dismounting of a machine, a start-up time c_m for this machine is needed. Pre-emption is allowed neither for jobs nor for setups. Each job j can be produced on a set of machines, but may only be assigned to exactly one machine. The binary parameter $\theta_{j,m}$ indicates whether job j can be done on machine (m). At any point in time $t = 1, \dots, T$, there can be at most Q_t overlaps of setups, dismounting, and start-ups.

For each job and each setup, the assignment and start time (hence the completion time) is to be decided. The goal is to minimize total tardiness. Note that we can assume that each job follows its preceding setup without any idle time in between.

5.2.1 Standard model formulation

$$min \quad \sum_{j=1}^{J} ta_j \tag{5.1}$$

$$L_j - (S_j + p_j + s_j) \leq ta_j$$

$$\forall \quad j = 1, \dots, J \tag{5.2}$$

$$s_j \geq \sum_{m=1}^{M} \sum_{i=1}^{J} s_{i,j} \cdot \phi_{i,j}^{m} \qquad \forall \quad j = 1, \dots, J \tag{5.3}$$

$$s_j \geq \sum_{m=1}^{M} \gamma_{j,m} \cdot c_m \qquad \forall \quad j = 1, \dots, J \tag{5.4}$$

$$p_j = \sum_{m=1}^{M} p_{j,m} \cdot \gamma_{j,m} \qquad \forall \quad j = 1, \dots, J \tag{5.5}$$

$$\sum_{i=1}^{J} \sum_{j=1, j \neq i}^{J} \phi_{i,j}^{m} \geq \sum_{j=1}^{J} \gamma_{j,m} - 1 \qquad \forall \quad m = 1, \dots, M \tag{5.6}$$

$$\sum_{m=1}^{M} \sum_{i=1}^{J} \phi_{i,j}^{m} \leq 1 \qquad \forall \quad j = 1, \dots, J \tag{5.7}$$

Indices	
m	Machine index m = 1, . . . , M
j	Job index, j = 1, . . . , J
t	Time index, t = 1, . . . , T
Parameters	
$e_{j,m}$	Binary parameter which is 1, if job j is producible on machine m
$p_{j,m}$	Processing time of job j on machine m
$s_{l,j}$	Setup time of job j, if j follows job l
$b_{j,m}$	Dismounting time of job j on machine m
c_m	Start-up time of machine m
d	Timeslot for shutting down a machine
t_t	Time at point in time t
K	Adequate large number
Q_t	Number of setup overlaps allowed at time t
L_j	Delivery date of job j
Variables	
S_j	Starting time of job j
p_j	Processing time of job j as defined in the model
s_j	Setup time of job j as defined in the model
ta_j	Tardiness of job (j)
C_{max}	Makespan
θ_j	Binary variable which turns 1 if job j is too late
$\gamma_{j,m}$	Binary variable which turns 1 if job j is produced on machine m
$\iota_{j,m,t}$	Binary variable which turns 1 if job j is produced on machine m at time t
$\alpha_{i,j}$	Binary variable which turns 1 if setup of job j is finished after setup of job i
$\beta_{i,j}$	Binary variable which turns 1 if job j is finished after job i
$\zeta_t^{i,j}$	Binary variable which turns 1 if setup of job i and setup of job j overlap at time t
$\sigma_{i,j}$	Binary variable which turns 1 if job j is produced after job i
$\phi_{i,j}^m$	Binary variable which turns 1 if job j is produced next after job i on machine m
$\eta_{i,j}$	Binary variable which turns 1 if job j is produced immediate after job i
$\lambda_{i,j,t}$	Binary variable which turns 1 if t is before the starting time of job i
$\omega_{i,j,t}$	Binary variable which turns 1 if t is after the ending of the setup time of job i or job j
$\delta_{i,j,t}$	Binary variable which turns 1 if t is after the ending of the processing time of job i or job j

Table 5.1: Notation parallel machines

$$\sum_{m=1}^{M}\sum_{j=1}^{J}\phi_{i,j}^{m} \leq 1 \qquad \forall \quad i = 1, \ldots, J \quad (5.8)$$

$$\gamma_{j,m} \geq \phi_{i,j}^{m} \qquad \forall \quad i = 1, \dots, J, j = 1, \dots, J, i \neq j, m = 1, \dots, M \quad (5.9)$$

$$\gamma_{i,m} \geq \phi_{i,j}^{m} \qquad \forall \quad i = 1, \dots, J, j = 1, \dots, J, i \neq j, m = 1, \dots, M \quad (5.10)$$

$$\sum_{m=1}^{M} \gamma_{j,m} = 1 \qquad \forall \quad j = 1, \dots, J \quad (5.11)$$

$$\gamma_{j,m} \leq e_{j,m} \qquad \forall \quad j = 1, \dots, J, m = 1, \dots, M \quad (5.12)$$

$$(1 - \phi_{i,j}^{m}) \cdot K + S_j \geq S_i + s_i + p_i$$

$$\forall \quad i = 1, \dots, J, j = 1, \dots, J, i \neq j, m = 1, \dots, M \quad (5.13)$$

$$S_i + \sigma_{i,j} \cdot K \geq ((S_j + s_j) - K \cdot \zeta_t^{i,j}) - K \cdot \lambda_{i,j,t} - K \cdot \omega_{i,j,t}$$

$$\forall \quad t = 1, \dots, T, i = 1, \dots, J, j = 1, \dots, J, i \neq j \quad (5.14)$$

$$\sum_{j=1}^{J} \zeta_t^{i,j} \leq Q_t \qquad \forall \quad t = 1, \dots, T, i = 1, \dots, J \quad (5.15)$$

$$(\sigma_{i,j} - 1) \cdot K + S_i \leq S_j \qquad \forall \quad i = 1, \dots, J, j = 1, \dots, J, i \neq j \quad (5.16)$$

$$\sigma_{j,i} \geq 1 - \sigma_{i,j} \qquad \forall \quad i = 1, \dots, J, j = 1, \dots, J, i \neq j \quad (5.17)$$

$$\sigma_{j,i} + \sigma_{i,j} = 1 \qquad \forall \quad i = 1, \dots, J, j = 1, \dots, J, i \neq j \quad (5.18)$$

$$t_t \geq S_i - K \cdot \lambda_{i,j,t} \qquad \forall \quad i = 1, \dots, J, j = 1, \dots, J, i \neq j, t = 1, \dots, T \quad (5.19)$$

$$t_t < S_i + K \cdot (1 - \lambda_{i,j,t})$$

$$\forall \quad i = 1, \dots, J, j = 1, \dots, J, i \neq j, t = 1, \dots, T \tag{5.20}$$

$$t_t \leq S_i + s_i + K \cdot \omega_{i,j,t} + K \cdot (1 - \alpha_{i,j})$$

$$\forall \quad i = 1, \dots, J, j = 1, \dots, J, i \neq j, t = 1, \dots, T \tag{5.21}$$

$$t_t \geq S_i + s_i - K \cdot (1 - \omega_{i,j,t}) - K \cdot (1 - \alpha_{i,j})$$

$$\forall \quad i = 1, \dots, J, j = 1, \dots, J, i \neq j, t = 1, \dots, T \tag{5.22}$$

$$t_t \leq S_j + s_j + K \cdot \omega_{i,j,t} + K \cdot \alpha_{i,j}$$

$$\forall \quad i = 1, \dots, J, j = 1, \dots, J, i \neq j, t = 1, \dots, T \tag{5.23}$$

$$t_t \geq S_j + s_j - K \cdot (1 - \omega_{i,j,t}) - K \cdot \alpha_{i,j}$$

$$\forall \quad i = 1, \dots, J, j = 1, \dots, J, i \neq j, t = 1, \dots, T \tag{5.24}$$

$$\omega_{i,j,t} = \omega_{j,i,t} \qquad \forall \quad i = 1, \dots, J, j = 1, \dots, J, i \neq j, t = 1, \dots, T \tag{5.25}$$

$$S_i + s_i \geq S_j + s_j - K \cdot \alpha_{i,j}$$

$$\forall \quad i = 1, \dots, J, j = 1, \dots, J, i \neq j \tag{5.26}$$

$$t_t \leq S_i + s_i + p_i + K \cdot \delta_{i,j,t} + K \cdot (1 - \beta_{i,j})$$

$$\forall \quad i = 1, \ldots, J, j = 1, \ldots, J, i \neq j, t = 1, \ldots, T \tag{5.27}$$

$$t_t \geq S_i + s_i + p_i - K \cdot (1 - \delta_{i,j,t}) - K \cdot (1 - \beta_{i,j})$$

$$\forall \quad i = 1, \ldots, J, j = 1, \ldots, J, i \neq j, t = 1, \ldots, T \tag{5.28}$$

$$t_t \leq S_j + s_j + p_j + K \cdot \delta_{i,j,t} + K \cdot \beta_{i,j}$$

$$\forall \quad i = 1, \ldots, J, j = 1, \ldots, J, i \neq j, t = 1, \ldots, T \tag{5.29}$$

$$t_t \geq S_j + s_j + p_j - K \cdot (1 - \delta_{i,j,t}) - K \cdot \beta_{i,j}$$

$$\forall \quad i = 1, \ldots, J, j = 1, \ldots, J, i \neq j, t = 1, \ldots, T \tag{5.30}$$

$$\delta_{i,j,t} = \delta_{j,i,t} \qquad \forall \quad i = 1, \ldots, J, j = 1, \ldots, J, i \neq j, t = 1, \ldots, T \tag{5.31}$$

$$S_i + s_i + p_i \geq S_j + s_j + p_j - K \cdot \beta_{i,j}$$

$$\forall \quad i = 1, \ldots, J, j = 1, \ldots, J, i \neq j \tag{5.32}$$

$$(\phi_{i,j}^m - 1) \cdot K + S_i + p_i + s_i + d \geq S_j - (1 - \eta_{i,j}) \cdot K$$

$$\forall \quad i = 1, \ldots, J, j = 1, \ldots, J, i \neq j, m = 1, \ldots, M \tag{5.33}$$

$$(\phi_{i,j}^{m} - 1) \cdot K + S_i + p_i + s_i + b_{i,m} + c_m \leq S_j + \eta_{i,j} \cdot K$$

$$\forall \quad i = 1, \ldots, J, j = 1, \ldots, J, i \neq j, m = 1, \ldots, M \tag{5.34}$$

$$S_j, t_{j,m} \geq 0 \qquad \forall \quad j = 1, \ldots, J \tag{5.35}$$

$$e_{j,m} \leq 0. \qquad \forall \quad j = 1, \ldots, J \tag{5.36}$$

$$\theta_j, \gamma_{j,m}, \iota_{j,m,t}, \alpha_{i,j}, \alpha_{i,j}, \beta_{i,j}, \zeta_t^{i,j}, \sigma_{i,j}, \phi_{i,j}^{m}, \eta_{i,j}, \lambda_{i,j,t}, \omega_{i,j,t}, \delta_{i,j,t} \in \{0, 1\}$$

$$\forall \quad i = 1, \ldots, J, j = 1, \ldots, J, i \neq j, m = 1, \ldots, M, t = 1, \ldots, T \tag{5.37}$$

The objective function (5.1), minimize the sum of tardiness for all jobs ta_j, together with constraint (5.2), the tardiness for a job j must be greater than or equal the starting time for the job S_j + its processing time p_j + its setup time s_j - its due date L_j, represent the goal to minimize total tardiness. Constraint (5.3) defines the setup time for a job s_j. Jobs have sequence-dependent setup times depending upon the predecessor i. To reduce the usage of sequence-dependent setup times in all calculations, the setup of a job s_j is connected to the sequence-dependent setup time $s_{i,j}$ which is actually used ($\phi_{i,j}^{m}$). Constraint (5.4) ensures that the setup time of a job s_j is greater or equal to the start-up time of the machine c_m if the job is on the machine $\gamma_{j,m} = 1$. Constraint (5.5) sets the production time of a job j p_j equal to the machine-dependent production time $p_{j,m}$ on which the job runs in the solution $\gamma_{j,m} = 1$. Constraints (5.6), (5.7), (5.8), (5.9), (5.10), (5.11), ensure a strict sequence of jobs on machines. Constraint (5.11) ensures that each job is connected to exactly one machine. $\phi_{i,j}^{m}$ is the direct successor j, predecessor i relationship on machine m. Each job j can have no more than one predecessor on a machine m (5.7). Each job i can have no more than one direct successor on a machine m (5.8). There can only be a successor predecessor relationship on a machine m $\phi_{i,j}^{m}$, if the successor (5.9) and the predecessor (5.10) are assigned to the machine m $\gamma_{j,m} = 1$. The number of relationships on a machine must be greater than or equal to the number of jobs assigned to that machine - 1 (5.6). Moreover, considering (5.9) and (5.10), this prohibits sub-cycles as the number of direct relationships is limited. (5.12) ensures that a job j can only be processed on machine m $\gamma_{j,m} = 1$, if the machine is able to process job j $e_{j,m} = 1$. If job j is the successor of job i on machine m

$\phi^m_{i,j} = 1$, this makes the first part of (5.13) $(1 - \phi^m_{i,j}) \cdot K + S_j \geq S_i + s_i + p_i = 0$ such that the start time of job j S_j must be greater than or equal to the start time of i S_i + setup of i s_i and the processing time of i p_i.
Constraints (5.14)–(5.32) regulate the number of setups at any point t. As the explanation of these constraints would be more difficult to understand, only the basic functioning of them will be described if expressed in the linear model. For each point in time t, there is only a limited number of setups allowed Q_t. Therefore, for each t, it must be checked whether t is between the start of a job S_j and the end of the setup for the job $S_j + s_j$. $\sigma_{i,j}$ is a binary variable which is one if job j is produced within time frame d after job i on the machine (5.33). If this does not hold, the start-up time of job j must also incorporate the dismounting of the predecessor $b_{i,m}$ and the start-up time of the machine c_m on top of the finish time of the predecessor $(S_i + s_i + p_i)$ (5.34), (5.35), (5.36) and (5.37) define the variables domains.

5.2.2 Extension

In case the goal is to minimize makespan, constraints (5.38) and (5.39) replace constraints (5.1) and (5.2).

$$min \quad C_{max} \tag{5.38}$$

$$S_j + p_j + s_j \leq C_{max} \qquad \forall \quad j = 1, \ldots, J \tag{5.39}$$

5.3 Methods

Two TSs and two GAs were used to solve problems outlined before. The methods which showed the worst performance in Chapter 4 were not considered. The functionality and elements of the methods presented here, are derived from Section 4.3, so that the performance of the metaheuristics can be compared better. The methods are either used separately or combined in a VNS, as in Section 4.3.

The four algorithms are sorted in the VNS in view of their exploration of the neighbourhood. We start with the algorithms which search a neighbourhood more broadly and then continue with the TSs which investigate a neighbourhood more closely. This order is:

1. GA 2 point crossover,
2. GA 1 point crossover,
3. TS Swap Exchange,
4. TS Insert.

The VNS is stopped after q minutes. As we have only four methods here, each method has a maximum of $q/4$ minutes time before we switch to the next method in the VNS. When a method does not find a better solution within half of its allowed time, it is switched to the next method and the remaining time is added to the next method. The duration for the switching was found experimentally.
We use a two-stage combination of different heuristics to find a starting solution for the metaheuristics. In the first stage, we assign jobs to machines and decide the processing order (job-chain) on each machine. In the second stage, we find a schedule for the setup operators and thus the times at which the jobs are processed are also determined. For the TSs, only the best solution is taken while for the GAs we use all solutions to form the first population.

First stage heuristics:

1. Schedule jobs in non-descending order by their due date as early as possible. Do this on a machine which has the highest production speed for the job. If two machines have the same production speed, use the machine which can start the job earlier. Repeat until all jobs are processed.

2. Schedule jobs in non-descending order by their due date as early as possible. Do this on a machine which can start the job first. If two machines can start the job at the same time, use the machine with the higher production speed. Repeat until jobs are processed.

3. Schedule jobs in non-descending order by their due date as early as possible. Do this on a machine which can finish the job first. If two machines can finish a job at the same time, use the machine with the higher production speed for the job. Repeat until all jobs are processed.

4. Schedule jobs in non-descending order by their slack time as early as possible. Do this on a machine which can finish the job first. If two machines can finish a job at the same time, use the machine with the higher production speed for the job. Repeat until all jobs are processed.

5. Schedule jobs in non-descending order of the number of machines which they can be served by. Do this on a machine which can handle the least amount of jobs, as early as possible. If multiple jobs can be served by the same number of machines, schedule the jobs in non-descending order of their due dates. Repeat until all jobs are processed.

Second stage heuristics:

Each time a setup operator gets available, the heuristics use one of the following strategies to find his next assignment. These times are considered in non-decreasing order of their occurrences.

1. Of all jobs allowed to be processed with regard to the job-chains, set up the job with the earliest due date.

2. Of all jobs allowed to be processed with regard to the job-chains, set up the job with the least slack time.

3. Of all jobs allowed to be processed with regard to the chains, set up the job with the shortest setup time. Repeat until all jobs are processed.

4. Of all jobs allowed to be processed with regard to the chains, set up the job with the longest processing time. Repeat until all jobs are processed.

5. Of all jobs allowed to be processed with regard to the chains, set up the job from the chain with the largest sum of remaining setup and processing times. Repeat until all jobs are processed.

A solution is represented by a sequence of jobs together with the name of the machine they are assigned to. The sequence also indicates to which machine a setup operator should move next. Given the sequence 'J2 M1, J3 M2, J4 M0, J1 M3, this would mean first setting up machine 1 with job two, then machine 2 with job 3, then machine zero with job 4, etc.

5.3.1 Tabu Search

As the basic functionality of the TSs has been described in Subsection 4.3.1 extensively, we will here only focus on the differences and the main elements of the TSs which are used for this problem setting.

The neighbourhood is created and explored in a three-stage method in regard to the used objective function. Two different search methods (Swap exchange, Insert method) can be chosen. As both methods have been extensively explained in Subsection 4.3.1, they are not described again here. The fitness value must be calculated after each stage. Two different objective functions have been implemented (makespan minimization, minimize total tardiness). After each fitness value calculation, for each machine an objective value (i.e., tardiness) is calculated. There is a predefined ratio that describes which proportion of the machines is set to high or low. Our tests suggested that we set the ratio to 1/3 in smaller instances and up to 1/6 in large instances.

1st stage: For each job from a high machine, it is then moved to each low machine, if possible. The position of the job within the solution representation will stay the same. Only the assigned machine will be changed.

2nd stage: Use the selected exchange method.

1. For each job assigned to a high machine, exchange the position directly with each other job of the high machine, if possible.

2. For each job a assigned to a high machine, put it after each other job b of the high machine, and move all jobs that were previously between these two on the high machine forward by one position.

In both cases, the search space for a position in to which to insert or in which to swap an element is limited to the next 15 positions due to time constraints.

3rd stage: Set up the jobs based on the sequence. If a setup operator has idle time and can only start the next job at time t, search in ascending order of the goal (i.e. due dates) for another job that can be set up before time t. If such a job is found, do the setup. Repeat until all jobs are processed. We call this post-processing.

After a neighbourhood has been created, and their solutions have been rated, the best solution, which is not prohibited by the tabu list or is prohibited but meets the aspiration criterion (the solution is the new best global solution), is taken. Besides the exchange methods, a shake is used, to randomly generate a new origin every 200 iterations. Thus, we choose two jobs randomly which change their position and machine assignment if possible. This is done until 50% of all jobs have been exchanged.

The tabu list consists of the last conducted exchanges and moves on the machines. The list follows a first in, first out principle, which means that the oldest moves and exchanges in the tabu list leave it first. The size of the tabu list is dynamic, but there is a minimum (10) and a maximum (50) size. If the fitness value of the current solution is better than that of the last solution, the size of the tabu list is reduced by one, if the minimum has not been reached. If the fitness value of the current solution is lower than that of the last solution, the size of the tabu list is extended by one, if the maximum has not been reached. The goal of the procedure is that areas which lead to improvements should be explored more intensively than areas which cause deterioration.
The algorithm runs as follows:

```
while (gap > y) {
    origin string = best carried over string of the past iteration
    if (iteration == one of the predefined values to shake the origin string) {
        shake origin string
    }
    for ( i = 0 ; i < last position in the origin string ; i++) {
        for ( m = 0 ; m < number of machines ; m++) {
            if (job in position i is on a high machine) {
                if (machine m is on a low machine) {
                    move the job in position i from the high to the low ma-
                    chine, if possible
                    calculate fitness value of the new string and save string in
                    the temporary neighbourhood
                    }
                }
            }
        }
        for (i = 0 ; i < last position in the origin string ; i++) {
            for (f = i+1 ; f <= last position in the origin ; f++) {
                if (job in position p is on a high machine) {
                    if (job in position f is on the same machine) {
                        use chosen exchange method
                        post-processing
                        calculate fitness value and save string in temporary neigh-
                        bourhood
                    }
                }
            }
        }
        sort the strings regarding their fitness value in the temporary neighbour-
        hood
        make the first string whose move is not prohibited by the tabu list (or is
        prohibited but meets the aspiration criterion= the new origin string
        if (fitness value of the old origin string < fitness value of new origin string)
        {
            if (size of tabu list < maximum size of tabu list ) {
                add the move from the old origin string to the new origin string
                add the move to the tabu list
            }
            else {
                delete the oldest move in the tabu list
                add the move from the old origin string to the new origin string
            add the move to the tabu list
        }
    }
    if (fitness value of the old origin string > fitness value of new origin string)
    {
        if (size of tabu list > minimum size of tabu list ) {
            delete the two oldest moves from the tabu list
            add the move from the old origin string to the new origin string
            add the move to the tabu list
        }
        else {
            delete the oldest move in the tabu list
            add the move from the old origin string to the new origin string
            add the move to the tabu list
            add the move from the old origin string to the new origin string
            add the move to the tabu list
        }
        if ( fitness value of so far found best solution > fitness value of new
        origin string) {
            update so far found best solution
        }
    }
}
```

5.3.2 Genetic Algorithm

In this chapter, we focus only on the main elements and the differences to Subsection 4.3.2, in which the GAs functionality has been described in detail. Two GAs are applied. We use a 2-point crossover and a 1-point crossover. In all variants, the population is limited to 500 chromosome strings.

The same selection method for the **2-point crossover** as in Section 4.3 explained, is used. In contrast to Section 4.3, the following formula is used here and in Chapter 6 to assign the selection probability to strings:

$$\text{Selection probability in \% of string a} = \frac{fitnessvalue_a \cdot 100}{\sum\limits_{i=string1}^{\text{last string}} fitnessvalue_i}.$$

If strings have a higher selection probability of $q\%$, they are added to set C. The number of strings in C is c. Reduce the selection probability of each string in C to $q\%$. The sum of the differences between the higher selection probability and q is distributed among the strings which are not in set C with the help of the following formula:

$$\text{Selection probability of string a} = \text{Selection probability of string a} +$$

$$\frac{\text{the sum of the differences between the higher selection probability and } q}{\text{number of strings in population } -c}.$$

Rank the strings according to their fitness value

$$\text{selection probability of string i} = \frac{\text{position in ranking for i}}{\sum\limits_{m=1}^{\text{number of strings}} m} \cdot 100.$$

After each fitness value calculation, machines are categorized into high or low in regard to the objective function. For each machine, an objective value (i.e. end of last job) is calculated. Machines which have a higher average value are classified as high, machines which have a lower value are classified as low. The new chromosomes are built in a two-stage approach.

1st stage: In the first stage, only the assignment of machines to jobs is considered and not the job sequence itself. The new chromosome is built—from the first parent chromosome string until the first crossover point, the second parent chromosome string from the first crossover point until the second crossover point, and the first chromosome string from the second crossover point to its end. We create a second chromosome by exchanging the orders in which the parents are considered.
The exchange of genes is done sequentially, and each gene exchange is followed by a repair if necessary. To start with, the objective values of each machine gets adjusted. Here the average value from the respective values of the parents is built. The assumption is that machines which were previously below/above average in both parents will also be below/above in the new chromosome string. For machines which were classified differently from the parents the estimation is more unclear. Therefore, they move towards the new average. In case a machine has been assigned to a job on which it cannot be processed, the job gets assigned a non-marked machine with the lowest value on which it can be processed. The assigned machine gets marked now, such that it is not assigned anymore in the repair until all other low machines have been assigned.

2nd stage: In the second stage, only the sequence of jobs is considered. The procedure is the same as in the first stage, except for the repair. The repair is done after all genes between the two crossover points have been exchanged. Again, in the resulting strings, some jobs may not occur, while others have multiple occurrences. A repair is necessary. For each job, it is controlled if the number of occurrences in the new chromosome string is equal to that number in the parent string. Beginning with the first job, the repair is done as follows: If job q is underrepresented in the chromosome string, the first gene with the characteristic of an overrepresented job beginning left in the new chromosome string, is changed to the characteristic of the underrepresented job. If the job is overrepresented in the chromosome string, the leftmost gene with the characteristic q is changed to the characteristic of the next underrepresented job. This is done until all jobs are represented equally in the old and new chromosome strings. Note that the two presented crossover methods are likely to have different outcomes due to their different repair mechanisms.

Postprocessing After a sequence has been initially evaluated. A so called postprocessing is used. Setup the jobs regarding the sequence. If a setup operator has idle time and can only start the next job at time t, search in ascending order of the goal (i.e., due dates) for another job that can be setup before time t. If such a job is found, do the setup. Repeat until all jobs are processed.

In the case of the **1-point crossover**, the selection method is similar to the selection method of the 1-point crossover in Chapter 4. The same formula as for the 2-point crossover versions in this chapter is used to assign the fitness probabilities to the chromosome string. As in the 2-point crossover case, machines are classified into high or low after each fitness evaluation. The chromosomes are then also built in a two-stage approach in which in the first stage only the machine assignment gets changed and in the second stage the job sequence.

1st stage: All genes in the strings before the crossover point are swapped with each other. If a repair is necessary, because a job has been assigned a machine on which it cannot be processed, the repair is done analogous to the 2-point crossover version.

2nd stage: The crossover of the second stage is similar to the one of the first stage, except the repair, which is done differently. The repair follows the exchange of all genes before the crossover point and is undertaken in such a way that the relative precedence of genes from the parent chromosome strings is also reflected in the new chromosome strings. For each new child string c of the parent p with the same postfix is considered. From left to right, each gene in the parent p is considered, and it is checked whether the job q it represents is underrepresented in the child c. If it is, the first gene of the child after the crossover point that has not been replaced during the repair, is set to the characteristic of q. Finally, again the **post-processing** follows.

The two crossover variants share the same type of mutation and elitism. In addition to the crossovers, each chromosome string in the new population is selected with a probability of one percent for mutation. If a string is selected, two genes in the string are chosen randomly. The machines for each of the genes assigned are changed randomly to another machine which can process the job, before the genes are switched. As the results for the algorithms shall be comparable across the chapters, the same procedure for the elitism is used as in Subsection 4.3.2. The new population gets the two best chromosome strings from the old population. There are no ‘intermediate populations which contain partly old and new strings, meaning that a new population is established first completely before further steps such as post-processing and the rating are undertaken. The procedure runs as follows:

```
while (gap > y) do {
  while (size of new population < total population size - 2 ) do {
      pick 2 strings from the old population regarding their fitness value
      crossover and repair
      mutate each of the new chromosome strings with a predefined prob-
      ability
      add the 2 new strings to the new population
  }
  add the 2 best chromosome strings of the last population to the new
  population
  post-processing
  calculate fitness function
  sort the strings according to their fitness value
  if (best string of new population < so far found best global string)
  so far found best global string = best string of new population
  old population = new population
  clear new population
}
```

5.4 Computational results

The tests were performed on the same computer as in Chapter 4. The algorithms presented here were tested on an Intel i7 Quadcore 3.4 GHz, 16 GB memory computer with the help of randomly generated instances (5.4.1), instances from which the best solution is known (5.4.2) and real-life data received from the company (5.4.3). The programming language used was again Java.

5.4.1 Random instances

For each test case (Table 5.2), three instances are solved with randomly generated numbers. Like Subsection 4.4.1, instances are created in such a way that they not only reflect the production situation of the described company but also show the performance of the algorithm in larger instances. The cases are built as follows:

1. For each job–machine combination, randomly pull a number between 0 and 100. If the number is greater than 50, the job can run on the machine. If the job cannot run on any machine after the procedure, assign uniform randomly one machine.

2. For each job–machine combination, the job can run on, uniform randomly assign a processing time between 0 and 3000.

3. For each machine, uniform randomly assign a maximum interim time between 0 and 1000, in which no dismounting is needed.

4. For each job–machine combination, the job can run on, uniform randomly assign a dismounting time between 0 and 3000.

5. For each job–job combination, assign a setup time between the dismounting time (COT) and 3000.

6. For each machine, a start-up time (SUT) and 3000 equal to the smallest setup time for the job.

7. Due dates for the jobs. The due dates are calculated as follows: Sum up all maximum production and setup times of the jobs to a variable s. Multiply s with a positive factor to s*.[9] For each job assign uniform randomly a due date between 0 and s*.

The computing time is limited to 360 seconds per instance and method. From 10 to 25 machines, 100 to 500 jobs, and 1 to 5 setup overlaps are considered in an instance. In the total tardiness, as well as the makespan minimization problem, it was considered to create lower bounds by relaxing some constraints (unlimited number of setup overlap, identical parallel machines) and then use a polynomial algorithm. However, these generalizations of our problems are already NP-hard. Lenstra *et al.* [1977] describe the makespan minimization problem while Du and Leung [1990] show already that the total tardiness single machine scheduling problem, a special case of the parallel machine problem is NP-hard. We relaxed further constraints and tested them with cases which we were able to solve to optimality with the help of "CPLEX". It showed that the lower bounds were even weaker as in Subsection 4.4.1. We therefore compare the performance of the metaheuristics to the best starting heuristic after 360 seconds running time. Cases 42–53 have the objective to minimize the makespan.

[9]In our case we choose 2 as a factor.

Cases 54–65 have the objective to minimize total tardiness. The results are not in line with the results from Section 4.4. We thus plot the neighbourhoods for some methods to investigate the reasons for the mixed results more closely.

Case	Machines	Jobs	Setup time	Processing time	interim time	SUT	COT	setup overlaps
42 / 54	10	100	U(COT, 3000)	U(0, 3000)	U(0, 1000)	min setup time job	U(0, 3000)	1
43 / 55	10	100	U(COT, 3000)	U(0, 3000)	U(0, 1000)	min setup time job	U(0, 3000)	3
44 / 56	10	100	U(COT, 3000)	U(0, 3000)	U(0, 1000)	min setup time job	U(0, 3000)	5
45 / 57	10	200	U(COT, 3000)	U(0, 3000)	U(0, 1000)	min setup time job	U(0, 3000)	1
46 / 58	10	200	U(COT, 3000)	U(0, 3000)	U(0, 1000)	min setup time job	U(0, 3000)	3
47 / 59	10	200	U(COT, 3000)	U(0, 3000)	U(0, 1000)	min setup time job	U(0, 3000)	5
48 / 60	25	250	U(COT, 3000)	U(0, 3000)	U(0, 1000)	min setup time job	U(0, 3000)	1
49 / 61	25	250	U(COT, 3000)	U(0, 3000)	U(0, 1000)	min setup time job	U(0, 3000)	3
50 / 62	25	250	U(COT, 3000)	U(0, 3000)	U(0, 1000)	min setup time job	U(0, 3000)	5
51 / 63	25	500	U(COT, 3000)	U(0, 3000)	U(0, 1000)	min setup time job	U(0, 3000)	1
52 / 64	25	500	U(COT, 3000)	U(0, 3000)	U(0, 1000)	min setup time job	U(0, 3000)	3
53 / 65	25	500	U(COT, 3000)	U(0, 3000)	U(0, 1000)	min setup time job	U(0, 3000)	5

Table 5.2: Random instances 42–53 makespan / 54–65 total tardiness

Cases 42 - 53
What can be seen from Table 5.3 (random instances, maskepan) is that the enhancement for each metaheuristic to the starting solution is less in the larger cases. Furthermore, in most cases the enhancement is larger with increasing setup operator size. Therefore, it seems that easier and smaller instances can be solved better within the 360-second time frame.

As the GAs performed well in the makespan minimization cases either with random instances (4.4.1) or in the instances which were created in such a way that the best solution is known (4.4.2), it is expected that they also show here a good performance. The GAs show, however, only minor improvements from

	GA 2 point	GA 1 point	TS Swap	TS Insert	VNS
42	2,19%	2,52%	23,38%	9,10%	25,10%
43	4,84%	7,44%	18,42%	14,39%	23,95%
44	10,67%	13,89%	21,21%	22,84%	25,00%
45	0,12%	0%	9,15%	5,40%	12,99%
46	5,18%	1,09%	27,96%	15,34%	26,98%
47	5,34%	6,72%	31,52%	18,39%	28,32%
48	0,15%	0%	11,47%	7,66%	7,17%
49	0,17%	0,25%	16,55%	11,32%	11,12%
50	3,85%	0,24%	19,11%	14,82%	11,04%
51	0%	0%	1,44%	1,27%	1,32%
52	0%	0%	2,50%	1,91%	1,91%
53	0,09%	0%	2,64%	2,44%	2,56%

Table 5.3: Enhancement to the starting solution for cases 42 - 53

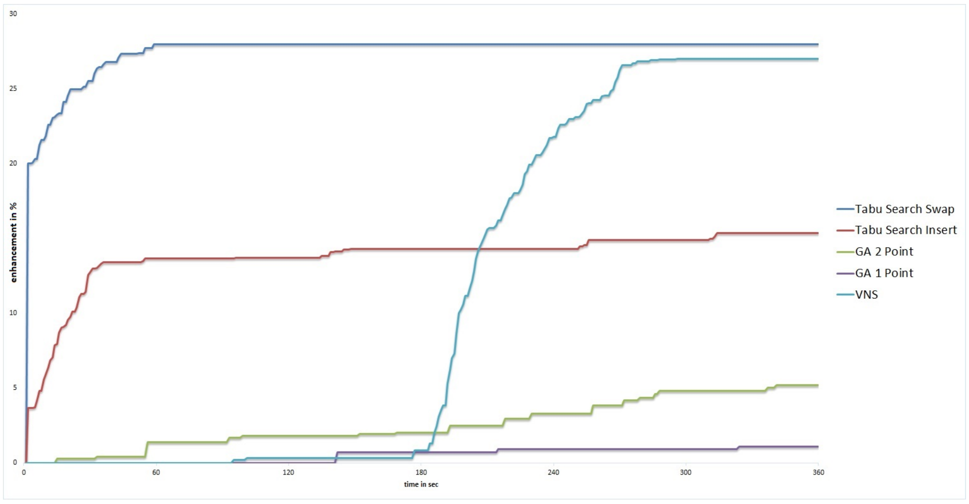


Figure 5.1: Diagram Case 46

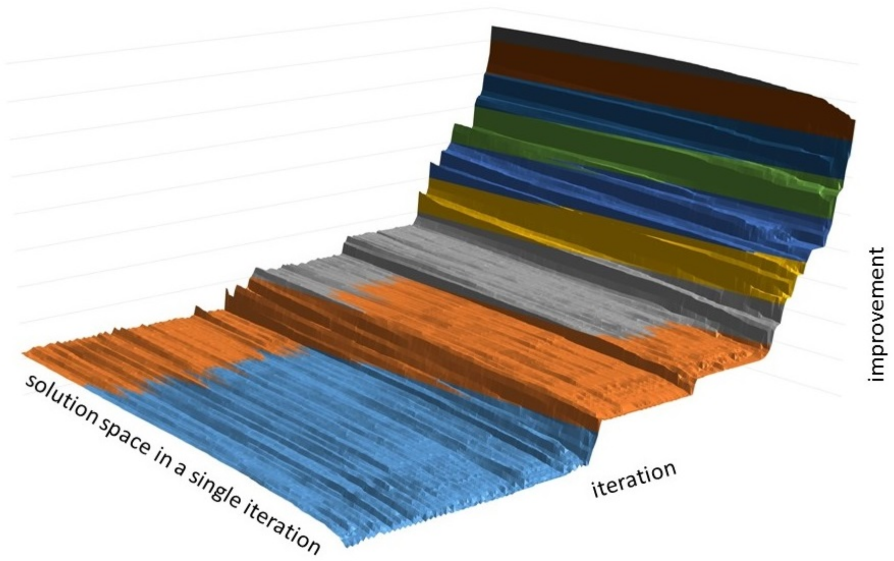


Figure 5.2: Neighbourhood TS Case 46

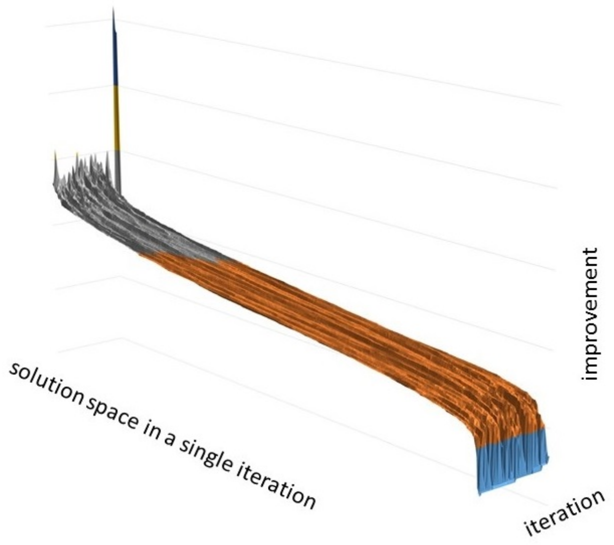


Figure 5.3: Neighbourhood GA Case 46

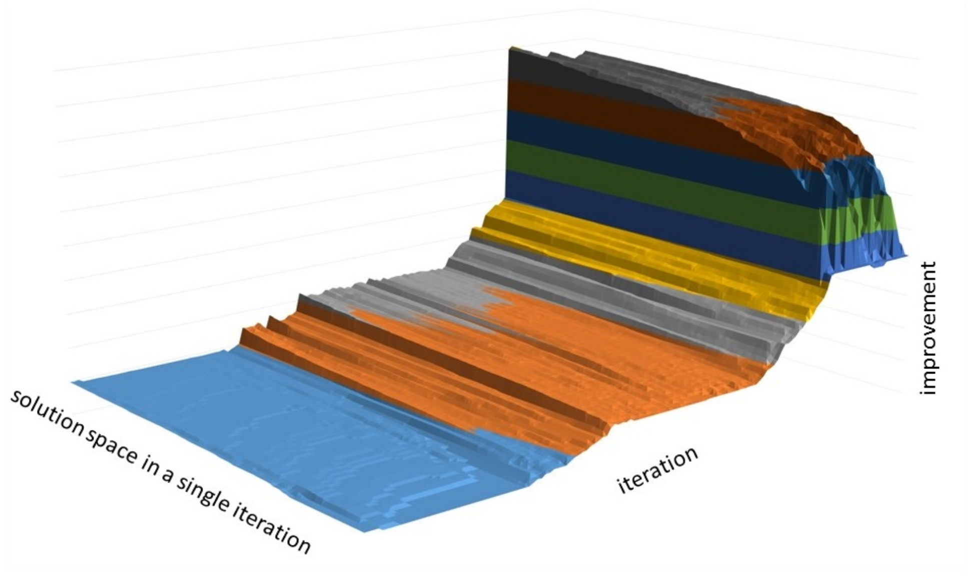


Figure 5.4: Neighbourhood VNS Case 46

the starting solution. When comparing only the GAs, both methods are, in five cases, better than the other. Larger cases with 250 jobs or more show rarely, respectively no improvement. In Figure 5.1 (Diagram Case 46) it can be seen that the GAs have a steady improvement over the time. The increase of the solution quality is nevertheless slow. On the other hand, as the slope is linear, it is expected that the GAs would with further running time also enhance the found solutions. Figure 5.2, 5.3 and 5.4 show how the GA 1 point, the TS Insert and the VNS see the neighbourhood for case 46. The fitness value is represented by the height in the figures. Since the minimization of the makespan is the objective, a lower fitness value is an enhancement. The width in the figures shows the solution space in one iteration.[10] The depth from right to left (back to front) shows the change of the neighbourhood with each iteration. Figure 5.3 shows the neighbourhood for the 1-point crossover GA. It can be seen clearly that the GA does not run the same amount of iterations as the TS 5.2. On the other hand, the GA has more solutions in a single iteration to solve. In each single iteration the GA has, furthermore, many solutions which are much worse than the best solution in the iteration. This is because the GA neighbourhood is, due to its structure, more diverse than the neighbourhood of the TS. Hence, it seems that only a small area of the overall solution space can be used to gain good solutions. Therefore, the GA cannot get any advantage over the local search methods by using more of the solution space. What can also be seen is that worse solutions decrease in the population over time. Therefore, it is likely

[10]Due to technical reasons it is not possible to show all solutions for each iteration. Therefore, sometimes only an excerpt of the best solutions is shown.

that the GA will come to a better solution with a longer running time, when the search space for the GA is more converging.

Here the TSs perform better than the GAs, although they were worse in Chapter 4 in the makespan cases. From the TSs, the Swap method dominates the Insert method. It is in all but one case better than the former. In the last eight cases, the Swap method finds the best solution among all metaheuristics. The Insert method also initially shows a steep incline before the incline decreases dramatically after approximately a 30-second running time. It seems nevertheless not to be stuck in a local optimum during the 360-second running time. From the graph in Figure 5.1, it can be seen that there is a steep decline/incline in the beginning and only little decline/incline in the later iterations. Interesting is the proportion of fitness values in a single iteration (Figure 5.2). While in the beginning the proportion of worse solutions in an iteration is higher, there is a convergence at the end towards good solutions. Hence, the TS is more and more stuck in a local or global optima.

The VNS finds the best solution among the metaheuristics in five cases. In the last eight cases is it close. When taking Figure 5.1 (Diagram Case 46) into consideration, the performance of the VNS seems clear. The 1-point crossover shows almost no gain, while the 2-point crossover shows a steady but slow linear gain. Therefore, the VNS also shows only a limited gain at the beginning. After 180 seconds, it is switched to the TS Swap heuristic, which immediately shows a much higher incline, which is similar to the incline the TS has at the very beginning, when the method is run separately. Therefore, the performance of the VNS is clearly dependent on the performance of the TS Swap in case 46. After 270 seconds it is switched to the Insert method. However, no further gains can be achieved by this. Therefore, the VNS stays below the result of the Swap method in case 46.

Overall, the results are surprising when they are compared to those with the target to minimize makespan in Chapter 4. While the GAs do not perform well at all, the TSs can at least get some enhancements. While the metaheuristics have the same number of jobs, the neighbourhood structure seems to be worse for the metaheuristics. Especially, the GAs are not able to run many iterations. So, the problem in Chapter 4 was either very easy, or the complexity here has increased exponentially.

Cases 54 - 65
Overall, it can be clearly seen in Table 5.4 (random instances, total tardiness) that the methods show a higher improvement in the smaller instances. Therefore, it shows that new algorithms should not only be tested on small problem instances but also on large real-world problem instances.
Although the GAs do not find the best solution among all methods in any case, especially in the smallest instances, their results are close. They have problems in the large instances. Furthermore, the GAs find better solutions when the

	GA 2 point	GA 1 point	TS Swap	TS Insert	VNS
54	4,59%	5,24%	28,09%	9,89%	34,78%
55	62,45%	58,99%	66,68%	68,30%	58,99%
56	68,29%	67,13%	75,31%	74,82%	70,05%
57	0,48%	0,63%	32,20%	32,22%	23,23%
58	8,05%	7,67%	35,64%	35,64%	24,54%
59	18,43%	16,64%	45,36%	45,32%	27,25%
60	0,22%	0,32%	13,03%	13,03%	6,66%
61	0,60%	1,60%	13,96%	13,96%	6,95%
62	3,06%	1,62%	15,23%	15,23%	8,04%
63	0%	0%	1,03%	1,03%	1,91%
64	0%	0,19%	1,06%	1,06%	1,85%
65	0%	0,41%	1,38%	1,38%	2,34%

Table 5.4: Enhancement to the starting solution for cases 54–65

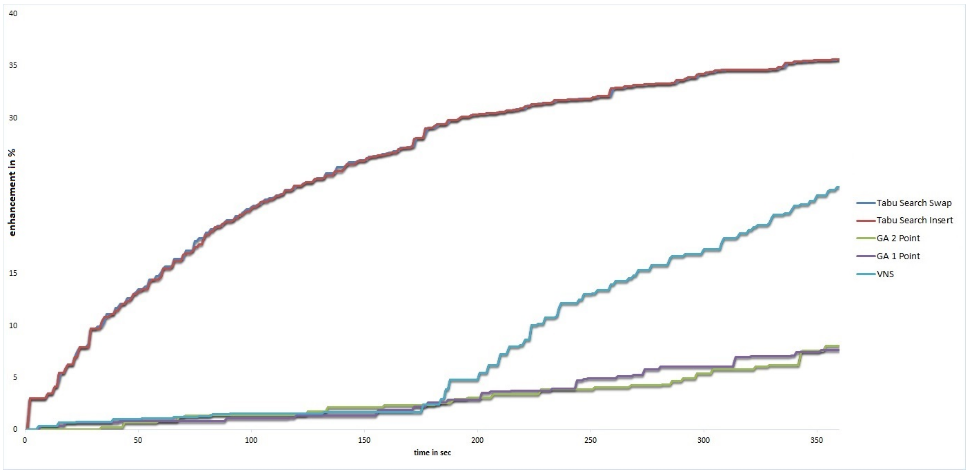


Figure 5.5: Diagram Case 58

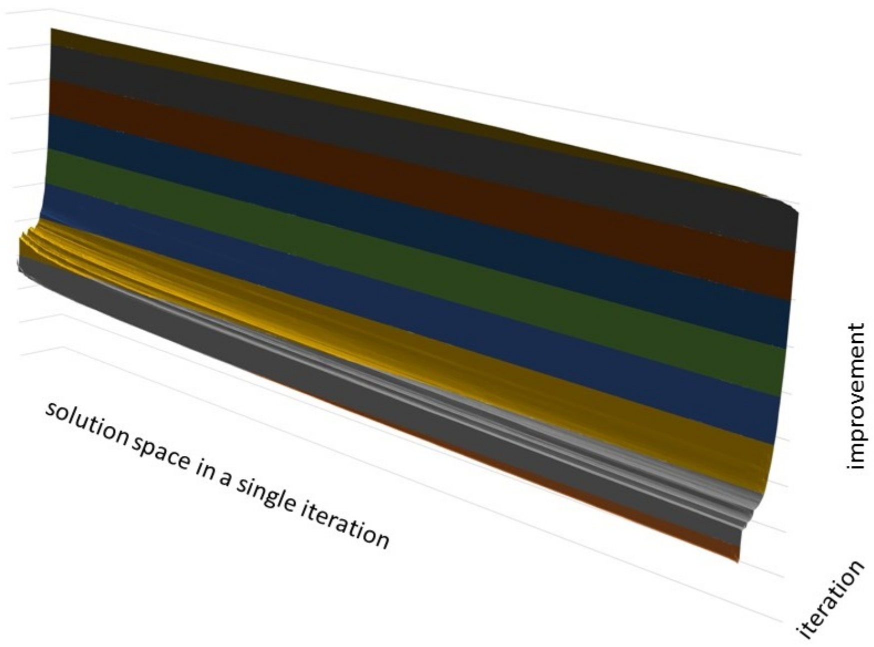


Figure 5.6: Neighbourhood TS Case 58

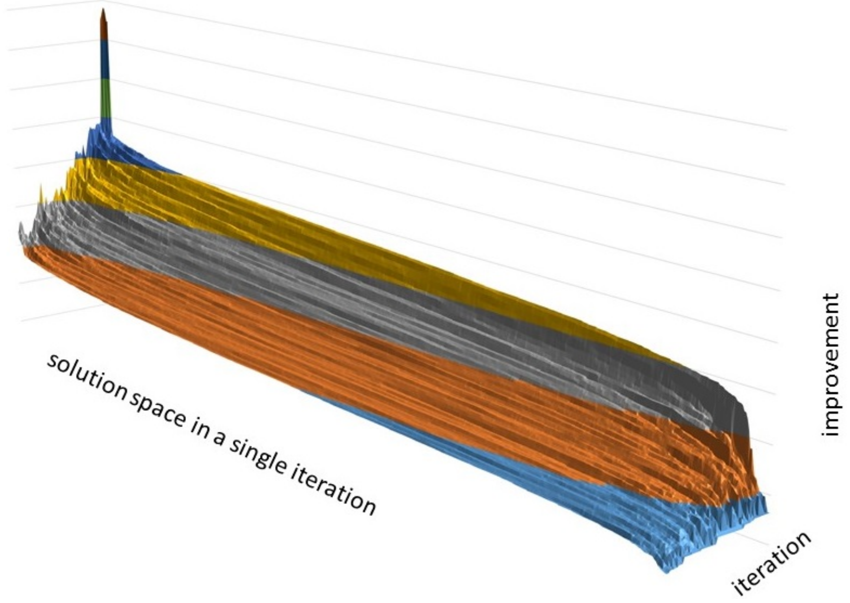


Figure 5.7: Neighbourhood GA Case 58

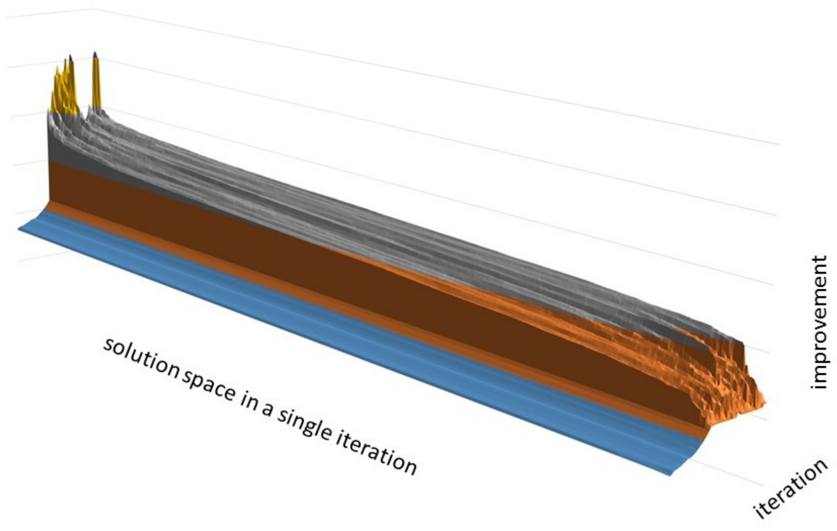


Figure 5.8: Neighbourhood VNS Case 58

number of setup operators is set high. Therefore, it seems that the GAs have problems finding feasible solutions. This, coupled with a large problem instance, leads to a weak performance. But it is much better than in the cases with the task to minimize makespan. Figure 5.5 (Diagram Case 58) shows as 5.1 (Diagram Case 46) a steady enhancement for the GAs. What can especially be seen in Figure 5.7 (Neighbourhood GA Case 58) is how a population eliminates weak solutions over time and new, better solution arise (blue level). As the population has not converged in Figure 5.7 (Neighbourhood GA Case 58) there is still an incline in each iteration, it seems clear that the GA would need more running time to find better solutions.

In each case, the TSs are better than the GAs. They are able to improve the starting solution by as much as 75,31%. The Swap method is in three cases better than the Insert. In all other cases the result is astonishingly the same. Although it is possible that the TS found the global optimum in these cases, the improvement is much lower than in the smaller instances. Therefore, it might also be just a local optimum. Each of the TS has found in six cases the best solution among the methods, making it the strongest methods in this context. In contrast to Figure 5.1 (Diagram Case 46), Figure 5.5 (Diagram Case 58) shows that there is not a superb performance in the beginning and no performance later. Instead, there is a steady but decreasing enhancement over time in case 58. Especially 5.6 shows that the enhancement for each iteration in the TS is much higher. The reasons for this remain unknown.

The VNS is the best method in the four cases. It outperforms especially in

the three largest instances. It is close in two further cases. In the other six cases, it performs much better than the GAs but has a gap to the TSs.
Figure 5.5 (Diagram Case 58) shows the problem with the VNS in case 58, one of the cases in which a gap to the solutions from the TSs remains. The VNS follows, during the first 180 seconds, the performance of the GAs. The performance of the GAs is weak but continuous, which is why the VNS does not switch to another method. After 180 seconds, it is then switched to the TSs in the VNS.
Figure 5.8 (Neighbourhood VNS Case 58) appears to be similar to Figure 5.4 (Neighbourhood VNS Case 46). The GAs shows only small enhancements at the beginning with a neighbourhood which has a diverse solution quality. When the switch to the TSs is undertaken, the solution values become more homogenous and the enhancement per time unit increases.

Nevertheless, also here, as in the makespan cases no conclusion for the different performance can be given. More detailed proposals will be made at the end of the chapter, where the results of the case studies from which the best solution is known are analysed.

5.4.2 Instances with known best solution

Since we cannot provide lower bounds for the randomly generated instances, we also create instances, similar to Subsection 4.4.2, so that the best solution is known beforehand. The structure is again chosen in such a way that in the optimal schedule the setup operators can have waiting time and the machines have waiting times. Cases 66–77 have the objective to minimize makespan. Cases 77–89 have the objective to minimize total tardiness. Instances for the makespan minimization problem are generated as follows:[11]

1. Job and machine setup times are 0 in the first step. The maximum available capacity of all machines between 0 and t is h. On each machine one job is added such that each machine has a workload of 100% between 0 and t. The workload of all machines between 0 and t is then also h time units. Given that the workload is 100% of the available capacity, the schedule must be optimal. (Figure 5.9)

[11]Note that the minimization of total tardiness is an enhancement of this makespan minimization generation procedure and thus described at the end of this procedure.

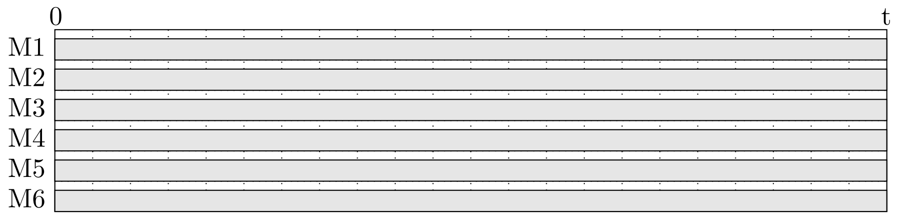


Figure 5.9: Generation of instances with known best solution parallel machines: Step 1

2. For each job, its processing time is randomly reduced by a setup time with the minimal setup duration between one and $\frac{t \cdot \text{number of setup operators}}{\text{number of machines} \cdot 2}$, and the time the machine needs to wait until it is set up the first time. Furthermore, it must hold: $t \cdot \text{number of setup operators} \geq \sum_{i=1}^{J} s_i$. The sum of the waiting times on all machines until the setup times start can be minimized if the operators set up the jobs in ascending order of the setup times until all machines are running. The available capacity with regard to the minimum waiting times still matches the sum of setup and processing times. Hence, the schedule is optimal. (Figure 5.10)

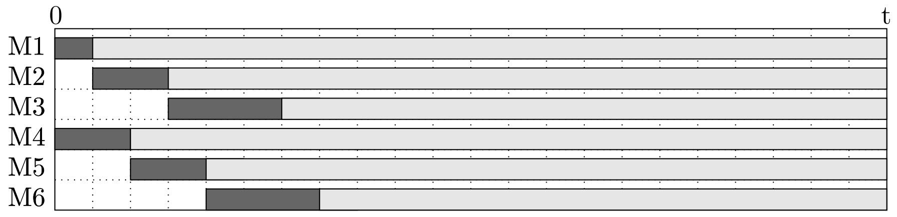


Figure 5.10: Generation of instances with known best solution parallel machines: Step 2

3. The processing times are interrupted by inserted setup times. Hence, new jobs are created. The minimum duration of the setup needs to be greater or equal to the m smallest setup time. Setup times can be inserted at any position on the machines between the end of the smallest setup and t, where the available capacity on the machine is given and minimum one of the setup operators has idle time. Furthermore, between two setups there needs to be a minimum processing time of a single time unit.

   (a) Randomly choose a feasible starting position for a setup.

(b) Randomly choose a finishing for the setup between the m minimum setup time and twice the value of m minimum setup time and the interruption of the capacity due to setup operator unavailability or the starting of the next job.

(c) Continue until the chosen job limit has been reached, or no capacity is left.

The available capacity with due regard to the minimum waiting times still matches the sum of setup and processing times. Hence, the schedule is optimal. (Figure 5.11)

4. Each created job now gets assigned additional values.

   (a) For each job–machine combination, except for the combination which has already been created, randomly pull a number from 0 to 100. If the number is greater than 50, the job can run on the machine.

   (b) For each job–machine combination, the job can run on, except for the combination which has already been created, uniform randomly assign a processing time between the assigned processing time and twice the value of the first assigned processing time in the schedule. However, the second overall job–machine combination is assigned the same processing time as the already created job–machine combination in the schedule. Thus, no simple sorting heuristic can easily identify which job should run on which machine.

   (c) For each job–job combination, except for the combination which has already been created, assign a setup time between the first chosen assigned setup time for the job in the schedule and twice the value of the first already assigned setup time in the schedule. However, the second overall job–job combination is assigned the same setup time as the already created job–job combination in the schedule. Thus, no simple sorting heuristic can easily identify which job combinations are part of the optimal schedule.

   (d) Due dates for the jobs: Each job is assigned a due date by which the processing of the job in the built schedule is finished. Thus, there is no tardiness in the optimal schedule. Since we cannot divide our best-found solution by the best-found solution of zero, we set the value of the optimal solution to one.

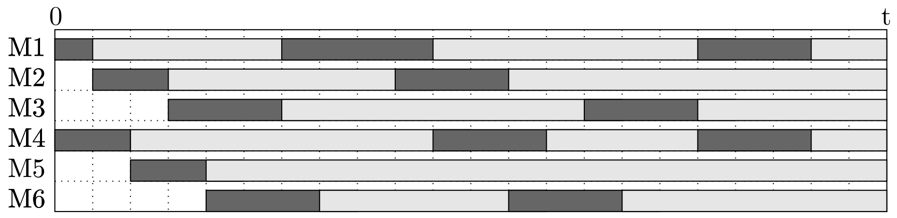


Figure 5.11: Generation of instances with known best solution parallel machines: Step 3

Case	Machines	Jobs	Setup operators
66 /78	10	100	1
67 /79	10	100	3
68 /80	10	100	5
69 /81	10	200	1
70 /82	10	200	3
71 /83	10	200	5
72 /84	25	250	1
73 /85	25	250	3
74 /86	25	250	5
75 /87	25	500	1
76 /88	25	500	3
77 /89	25	500	5

Table 5.5: Known best solution 66–77 makespan / 78–89 total tardiness

	Starting solution	GA 2 point	GA 1 point	TS Swap	TS Insert	VNS
66	13,53%	11,87%	12,47%	6,93%	7,30%	7,97%
67	5,47%	4,80%	5,47%	1,90%	1,33%	2,37%
68	5,37%	4,10%	5,37%	2,30%	2,20%	2,20%
69	5,97%	5,67%	5,67%	1,97%	2,20%	2,77%
70	5,50%	4,00%	4,67%	0,93%	1,07%	1,70%
71	6,63%	5,37%	5,13%	1,07%	0,77%	1,47%
72	17,50%	17,43%	16,53%	2,97%	3,60%	3,47%
73	19,20%	18,43%	17,60%	2,50%	2,83%	2,83%
74	15,00%	15,00%	13,20%	2,23%	2,43%	1,93%
75	9,67%	9,62%	9,62%	1,62%	2,20%	2,02%
76	9,93%	9,93%	9,83%	1,27%	1,47%	1,17%
77	9,53%	9,53%	9,50%	0,93%	1,60%	1,00%

Table 5.6: Gap to the optimum solution for cases 66–77

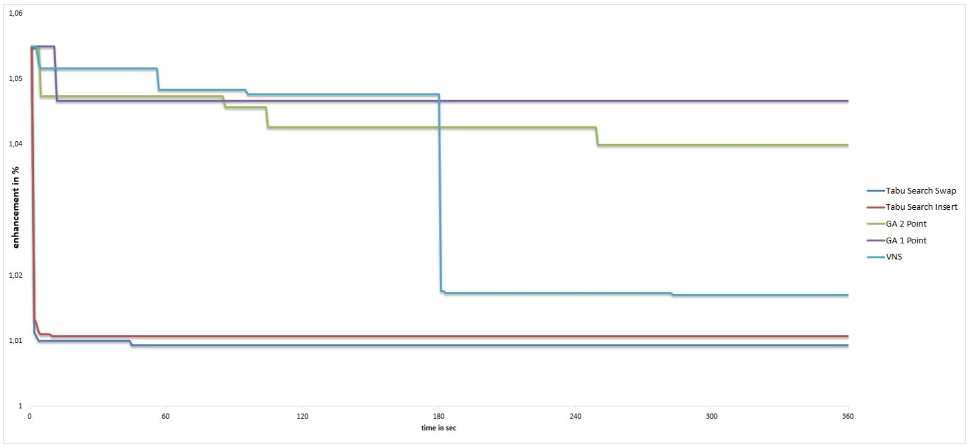


Figure 5.12: Diagram Case 70

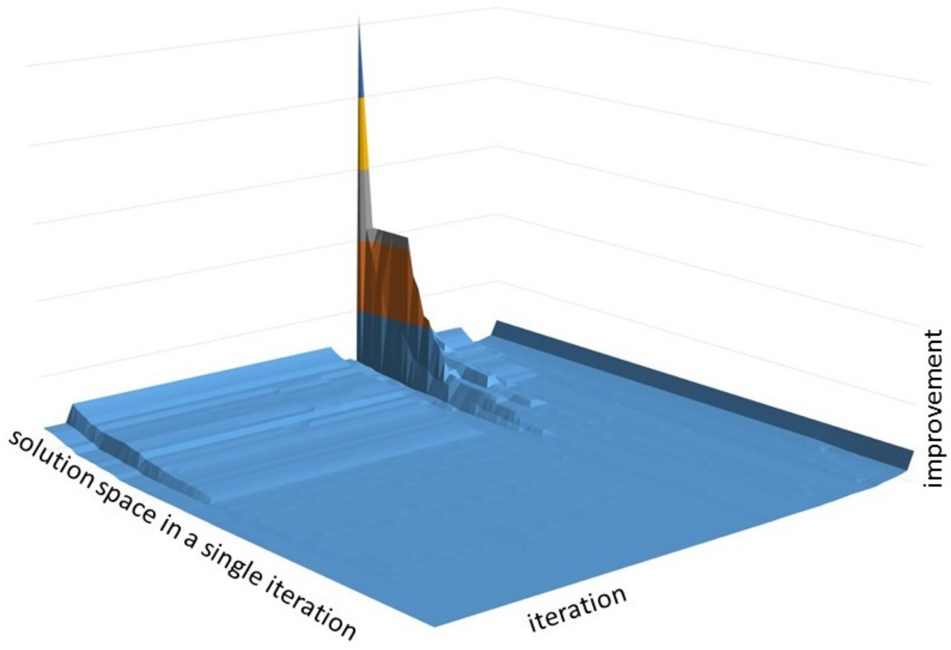


Figure 5.13: Neighbourhood TS Case 70

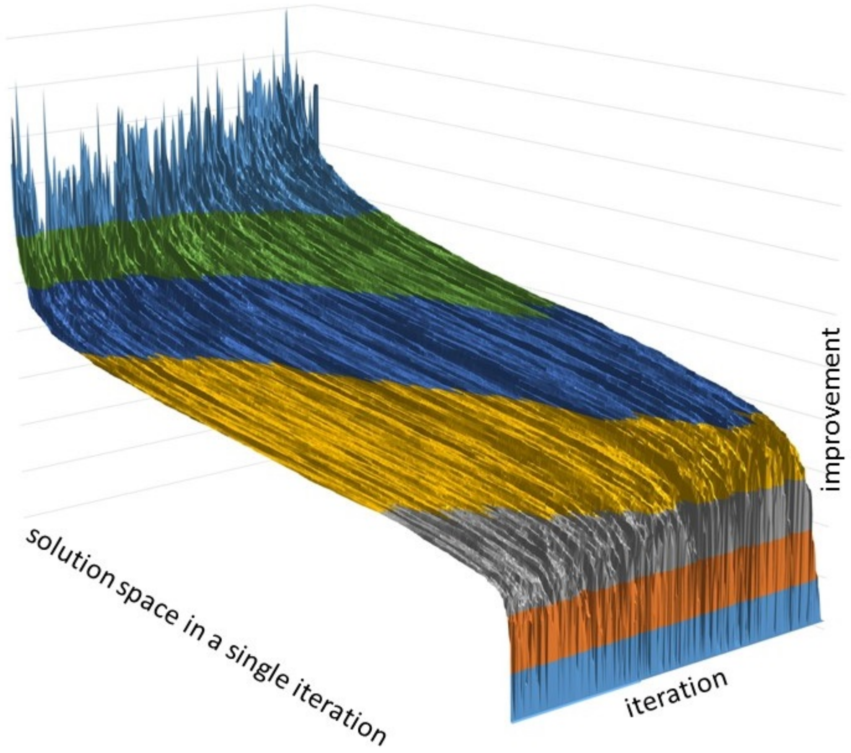


Figure 5.14: Neighbourhood GA Case 70

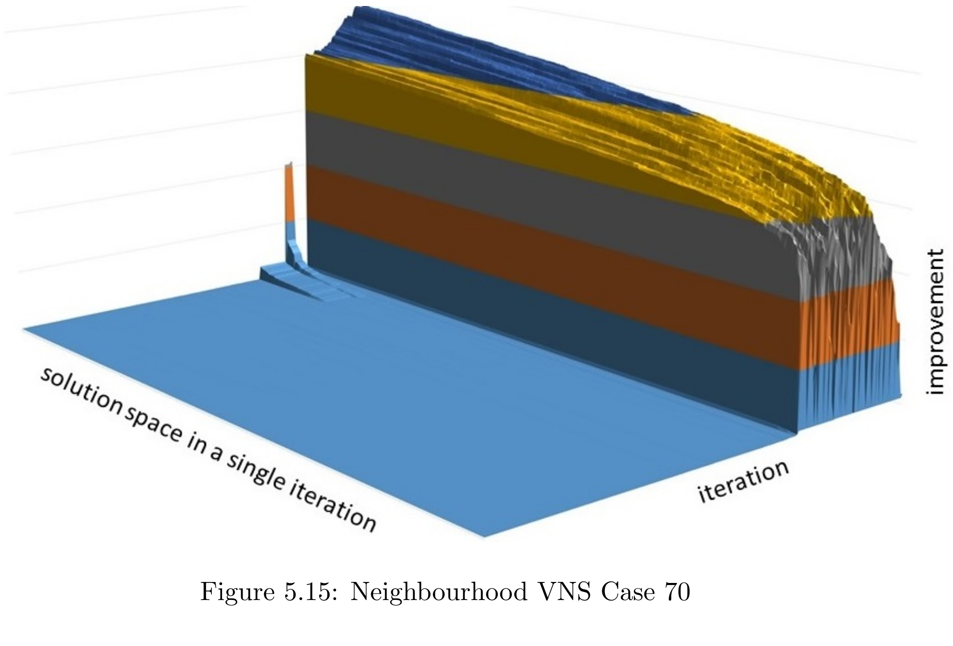


Figure 5.15: Neighbourhood VNS Case 70

Cases 67–77
For the minimize makespan known best solution cases 66–77, the possible gains for using a metaheuristic are somewhat limited. Most cases allow only savings in the single percentage area. The maximum gap to the optimal solution is 19,2% in case 73 (Table 5.6).
The GAs thus show only minor improvements. This is somehow explainable, as they, as already stated, search their neighbourhood rather broadly. It should be noted, however, that in the makespan minimization cases where the best solution is known in Chapter 4, the gap to the optimum is much higher for the starting heuristics, but the GAs also find solutions closer to the optimum. Figure 5.12 (Diagram Case 70) shows that the GA 1-point also finds better solutions at the beginning, but is then stuck. The GA 2-point shows a steady but slow performance and will likely come to a better solution with a longer running time. The neighbourhood for the GA (Figure 5.14) seems completely different than in Figure 5.3. The solution values of a single iteration ranges from values close to the optimum, to solution values which are eight times higher than the optimum solution. Nevertheless, it can also be seen that with the running time, weaker solutions get eliminated. Therefore, it is estimated that with further running time better solutions will also be found here. Especially noteworthy from 5.3 and 5.14 is that more iterations can be calculated in the same amount of time.

On the other hand, the TSs find in this neighbourhood also solutions which are very close to the optimum. In six of the twelve cases, the Swap exchange finds the best solution of all methods while the Insert method finds the best solution of all methods in three cases. Figure 5.12 (Diagram Case 70) shows furthermore that both TSs come to a much better solutions close to the optimum after a very short running time. Figure 5.13 shows clearly, how the TS Swap sees case 70. The area looks somewhat flat. There is a small enhancement in the beginning, after in many cases close or similar solutions are found. Only one area of the neighbourhood has much worse solutions. The results from the TSs can be said to be in line with the expectations. As said in Chapter 4, the TSs are expected to work well when they start in a neighbourhood which is close to the optimum as the TSs focus densely on the chosen neighbourhood and do not easily switch to another neighbourhood.

While the VNS in two cases finds only better solutions than the other metaheuristics, it has apart from case 68 in which the TS Insert and the VNS find the best solution, in four cases the second-best solution (case 73 together with the TS Insert). In Figure 5.12 (Diagram Case 70), the VNS first follows the path of the GAs and after 180 seconds switches to the TSs. The switch allows the VNS to also come closer to the optimum while keeping a gap from the solution values of the TSs. Figure 5.15 shows again how clearly the neighbourhoods are in the respective methods in the VNS seen. While in the beginning the solution quality of the neighbourhood is diverse, as soon as it is switched to the TS, the solution values of the neighbourhood become more homogenous. As the solution is, however, remarkably close to the optimum or has hit the optimum, no further large enhancements can be found.

Cases 78 - 89
The biggest surprise of the results of cases 78–89 (Table 5.7) (known best solution, total tardiness) is that the GAs show a very strong performance, as they were much worse with the same target when random instances where analysed. In four out of the 12 cases, they can find the optimal solution in each of the single tests in the cases. The 2-point crossover finds the best solution among all metaheuristics in 10 cases. The 1-point crossover comes to the same good solution values as the 2-point in five of the 10 cases. However, in the largest and most restricted case (87), the GAs come out last. Therefore, indicating again that they have problems with the size of the instance and the number of the setup operators. Figure 5.16 (Diagram Case 82) shows, in contrast to the other progression graphs in Section 5.4, a strong enhancement at the beginning for the GAs. The neighbourhood structure for the GA looks like the ones before in case 82 (5.18) , except that iterations get rid of weak solutions much faster and therefore include the newer, better, solutions faster. Hence, the enhancement per time unit is better for the GA until it is close to the optimum in case 82. Furthermore, the solution values in a single iteration seem to be much more homogenous. The reasons for this cannot be identified, unfortunately, by the data analysed so far. Further research seems necessary.

The TSs also find much better solutions in comparison with the starting solution. The TS Insert finds the best solution among all metaheuristics in one case while the Swap finds the best solution among all metaheuristics in no case. In seven cases the TS Swap finds a better solution than the TS Insert, which only finds a better solution in five cases. The solution value is in all cases, however, close to each other. Figure 5.16 (Diagram Case 82) shows that the TSs also give a good performance at the beginning, but are then stuck in a local optimum from which they find no better solution values. Figure 5.17 (Neighbourhood TS Case 82) indicates the problem the TSs clearly have. Different from the other TS neighbourhoods seen so far, the initial neighbourhood for the TS has a much more diverse solution quality. Hence, the TS might take a path which does not lead to the optimum solution easily.

The VNS finds the best solution among the heuristics in three cases. In a further case it does find a better solution to any of the other methods. In the cases with up to 200 jobs, the solutions are close to the best-found solutions of the other methods. In the larger instances, a gap still remains. Nevertheless, the VNS is still in all but one case better than the TSs. Figure 5.19 (Neighbourhood VNS Case 82) shows that the neighbourhood for the VNS is similar to the neighbourhood of the GAs. Further performance enhancements for the TSs or a transition from the GA methods to the TS methods can therefore not be seen, as the neighbourhood/population is already close to the optimum and homogenous. Therefore, the VNS is performing again very well, as it can focus

	Starting solution	GA 2 point	GA 1 point	TS Swap	TS Insert	VNS
78	2336	2,33	7	858,67	910,33	0,33
79	645,67	0	0	642,33	634	0
80	806,67	0	0	606,67	603	0
81	2137,67	0,33	0,33	1094,33	1083,33	15,33
82	2047,67	0	0	916	919,67	0
83	1464,67	0	0	947	947,33	1,33
84	3248	159,33	281	973	1013	987
85	2434	14	27	1058,33	1014,67	404,67
86	2855	18	27,67	1004,67	1013	517,33
87	8905	8569,67	6425,67	4107,33	4075,67	4287
88	4725,67	2184	3161,33	2448	2419,33	2402,67
89	5106	1584,33	1747,67	2820	2823	2734

Table 5.7: Gap to the optimum solution in time units for cases 78–89

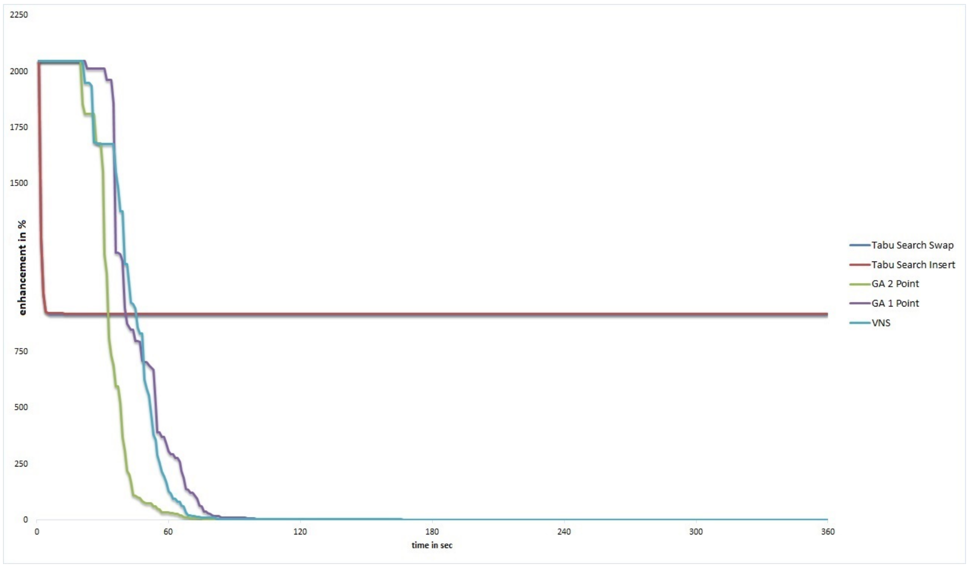


Figure 5.16: Diagram Case 82

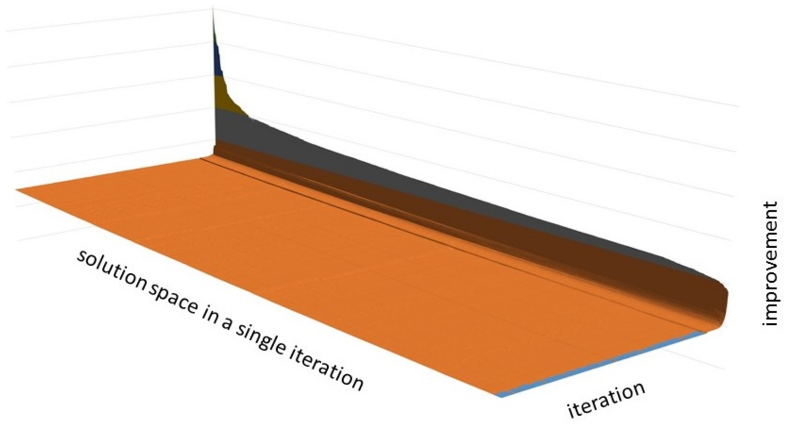


Figure 5.17: Neighbourhood TS Case 82

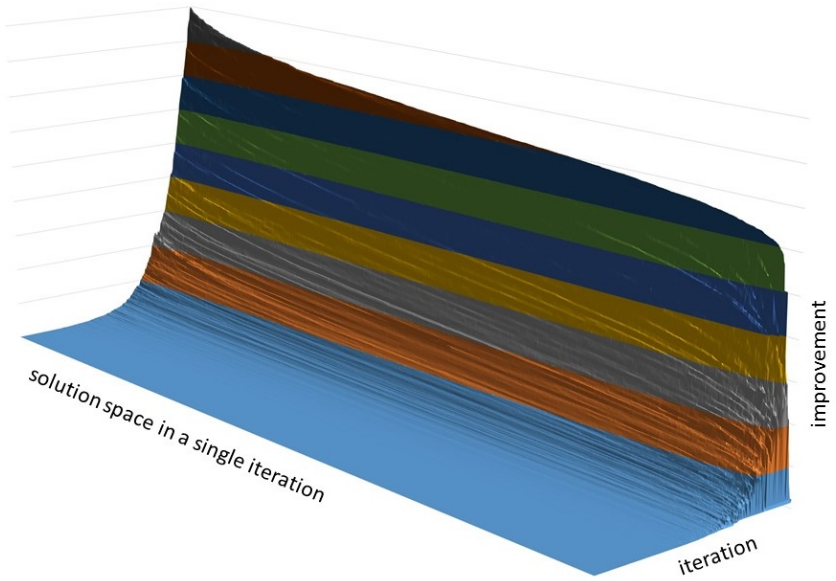


Figure 5.18: Neighbourhood GA Case 82

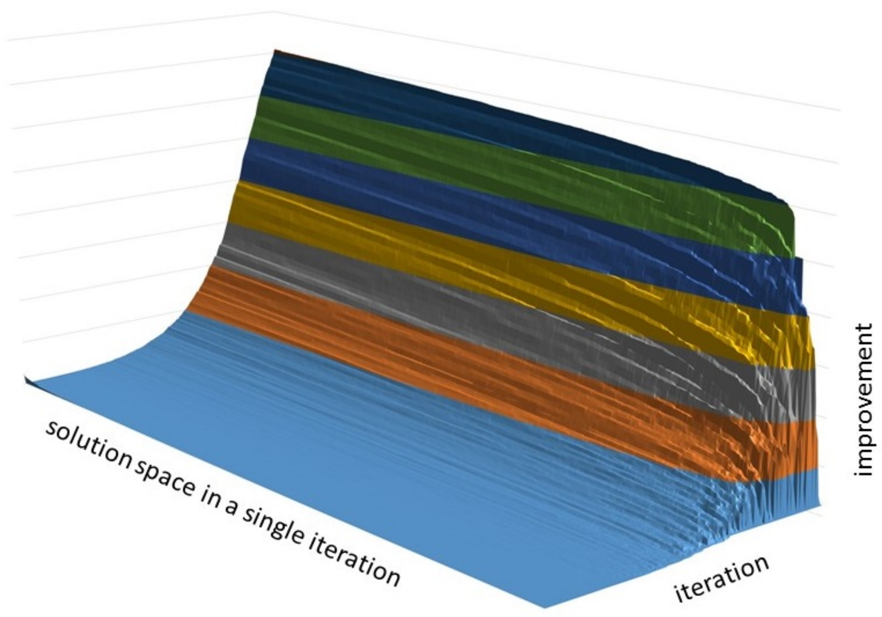


Figure 5.19: Neighbourhood VNS Case 82

on the method which brings the most advantages in this problem setting.

5.4.3 Case study

As the problem case has been extensively described already in 5.1, only the production data will be shown. Five old production instances were used for test runs, which are referred to as instance 90. The planning period covers the next four weeks. The number of jobs to be allocated and scheduled on the 19 machines is between 123 and 161 where on average 132 jobs need to be scheduled. The objective is to minimize total tardiness. For some products, only the customer quantity is produced. Other products, which are ordered more often, are delivered from stock. If the minimum stock level is reached, a production order with a fixed quantity is produced. The quantity is usually set such that for four month no new production for the article must be undertaken. Hence there are significant differences in the production times. Furthermore, it must be considered that the production times are machine-dependent, and the setup times are sequence-dependent. As a result, no average or median values can be given.
In the data we received from the company, the shortest production time for a job is 5 minutes and the longest is 4320 minutes. The shortest setup time for a job is 0 minute and the longest is 1440 minutes.
Furthermore, at each point in time, three setups are allowed to overlap.

Since the goal in this section is a comparison of the planning procedures of the company, we use their due date heuristic to find a starting solution and compare the VNS with it, because it showed the most steady performance. For the due date heuristic, the total tardiness is 156922 minutes, while for the VNS the total tardiness is 128316 minutes on average in a schedule. Thus, each job is on average 0,83 days late in the due date heuristic, while it is only 0,68 days late when the VNS is used. The number of finished jobs is nearly twice as high with 1,51 jobs per day for the VNS in contrast to the 0,77 jobs per day for the due date heuristic. While the utilization of the setup operators increases only marginally from 11,07% to 16,68%, the utilization of the machines increases from 2,7% to 5,8%. An explanation for the low utilization of the machines and setup operators might be that there is a bottleneck machine, which increases the makespan. It can be concluded that significant gains for the company can be achieved by using more advanced algorithms for their production planning.
When the expected performance of the other untested algorithms is described, the tests where the best solution is known and random instances show distinct results in each setting. In the comparable setting of random instances (200 jobs, three setup operators), the GAs give a weak performance and the TSs are the best algorithms of all tested ones, while the VNS comes out in the middle. In the comparable known best solution case (200 jobs, three setup operators), the TSs bring up the worst solutions while the GAs perform best. The VNS comes out in the middle. Since the random instances are from their structure closer to the case study, one would expect that the TS would provide the best results for this setting.

Furthermore, it is clearly shown that new algorithms should also be checked for their performance in large instance sizes to identify their practicability in the real world.

5.5 Conclusion

This chapter showed the continuation of the parallel dedicated machines subject to setup constraints chapter. Moreover, for the second company which was visited setup operators were not explicitly considered in the production planning. Therefore, the main aim of this chapter was to extend the production problem structure, the parallel machine scheduling problem with setup operator availability.

First, in 5.2., a formal problem definition was given. We distinguished the production time by a machine-dependent processing time, in which no setup operator was needed, and a sequence-dependent setup as well as dismounting time in which a setup operator was needed. In Section 5.3., we outlined the five different methods (two GAs, two TSs, and a VNS) which were used to solve the problems, with the focus on either minimizing total tardiness or makespan in Section 5.4.

Randomly generated instances with up to 25 machines and 500 jobs were tested in Subsection 5.4.1. The VNS provided the best solutions among the heuristics in the first four of the twelve makespan minimization cases. In the other eight cases, the Swap exchange provided the best results. The GAs, on the other hand, performed weakly, as they often only found minor improvements.
In the random instances total tardiness minimization problems, the GAs again showed a weak performance. The TSs, by contrast, showed a robust performance. They were able to improve the solutions by as much as 75,31% (case 56). Although the VNS was not as good as the TS in most cases, it did provide in some of the instances the best solutions. In all test cases, so far, it should be noted that the improvement for larger instances was small. This is, nevertheless, not a severe problem as the real-world problems, which form our focus, are smaller. They are as stated in 5.4.3. between 116 and 163 jobs. But it is shown that new algorithms should generally be checked for their performance in large instance sizes to identify their limitations and their practicability for the real world.

As there were problems with lower bounds, instances were created in 5.4.2 in such a way that the optimal solution is known. Cases 66–77 dealt with the makespan minimization problem. The GAs again only showed a weak performance. The TSs, on the other side, were able to get very close to the optimal solution. The VNS also got close to the optimal solution but stayed behind the TSs.
In cases 78–89 (minimize total tardiness), the GAs showed a surprisingly strong performance. Especially in the smaller cases with more than one setup operator, they were able to find particularly good or optimal solutions. While the VNS

was better than the TSs in most cases, it often still had a gap to the GAs.

Finally, the VNS, which has shown the steadiest performance in all test instances in Section 5.4 so far, was tested in Subsection 5.4.3 on real data provided by the company. It was able to nearly double the number of finished jobs per day in contrast to the used due date heuristic by the company. The utilization of the machines also increased.

Section	Type	Target	Density	Performance
Parallel dedicated machines	Random instances	Makespan no breaks	low	GAs best, TS good
		Tardiness no breaks	low	TSs better than rest
		Makespan breaks	low	bigger gap with breaks rest same
		Tardiness breaks	low	GAs have problems
	Known best solution	Makespan	high	all metaheuristics have strong performance

Table 5.8: Results 2

The different results clearly suggest that the performance of the methods depends on the problem structure. It was first thought that GAs do not perform in these types of problems, but it is now clear how context-dependent they are. On the other hand, the TSs provided promising results in most test settings. While the VNS did not provide the best solutions in most cases, it showed a good and steady performance in all instances. When the overall results of the two chapters are compared, new insights are gained (5.8 and 5.9). Clearly in all known best solution cases, all metaheuristics perform well. Interesting is that the GAs perform better than the TSs in the tardiness cases, while they had problems in the tardiness cases with breaks in Chapter 4. Furthermore, if the number of setup operators is high, also the known best solution tardiness cases can be solved better. Considering the results from Subsections 4.4.2 and 5.4.2, this is clear as the density (the proportion of working to idle times in a schedule) is higher in these cases, as the proportion between working setup operators and the setup operator with forced idle time in the second part of the created schedule in the known best solution cases is more in favor of the working setup operators. Hence, the schedule is denser. Moreover, in the random instance cases, the higher number of setup operators, the better the performance of the metaheuristics. Hence, a dense schedule is easier for the metaheuristics to solve than a schedule which has overall more freedom. The 360-second running time seems to limit the performance of the metaheuristics in the larger instances. While still some achievements can be made, they are higher in the smaller instances. This is thought to be due to the large neighbourhoods, which would probably require more running time. This is especially true for the GAs as they need some running time to get the population into a converging state. Otherwise, the GAs are more likely a random search. As stated before, it is unknown why the TSs are better in Chapter 5 in the cases with makespan than the GAs. The comparison between the overall results brings no answer for this question unfortunately.

The chapter provides additional insights into the rarely investigated field of setup operators in the field of production scheduling. To provide a conclusion for production companies, it can be said that they are likely to gain more efficiency in the production with the use of more advanced algorithms in contrast to the currently used sorting heuristics. Moreover, it can be clearly seen that setup operators also seem to be a bottleneck in the parallel machines problem. The gains the algorithms can achieve increase exponentially with the use of more setup operators. Therefore, companies should check whether or not it is possible to use more setup operators. Furthermore, this work contributes also to the so far sparse field of setup operator scheduling. Although we provide some explanations for the different performances of the metaheuristics, no proof can be given. Further research regarding the overall performance of metaheuristics seems necessary. The algorithms were furthermore only tested on the distinct case of parallel machines. As most companies have a more general problem structure there is a need for further investigation.

Section	Type	Target	Density	Performance
Parallel machines	Random instances	Makespan	low high	GAs not good, worse with larger instances, better with more setup operators/ TS better than GAs, better with more setup operators, worse with large instances
		Tardiness	low / high	GAs better than case with makespan, better with more setup operators, worse with large instances / TSs better than GAs, better in small instances, with more setup operators
	Known best solution	Makespan	high	good results for TSs and GAs , TSs better
		Tardiness	high	very good results for GAs , good results for TSs, better in small instances, better with more setup operators

Table 5.9: Results 3

Chapter 6

Job shop with parallel machines scheduling subject to setup constraints

6.1 Introduction

The problem in this chapter was brought to our attention by a medium-sized company from the automotive sector. They produce aluminium parts. The company is in the limits for revenue, staff employed, and balance sheet total of the IfM [2002]. The characteristics of the company indicate that it is an SME. The company is run by the owner. While he has no direct contact with customers any more, he is still involved in the daily operational business [cf. Mank, 1991]. The company also shows low formalization [cf. Mugler, 2006] and uses universal machines [cf. Pfohl, 2006]. Referring to our empirical study in Section 4.3, the production planning is done by hand in this company. It is possible to identify the person who created a plan using the production plan. An ERP system is furthermore in use. A lot of processes are, however, not realized in the ERP system but by email. Lot sizing is not an issue for this company. The company is therefore in line with most of the companies from the study in Section 2.3. Hence, the results can be used to create recommendations.
The problem described within this chapter is an extension of the "parallel machine setup operator scheduling problem", namely a "job shop with parallel machines subject to setup constraints"'. The difference from the "parallel machine setup operator problem is that there are no single jobs that can be produced in any given sequence; however, there are job chains that must be produced in a predetermined way. In another mode of expression it is also common to say there are jobs, which themselves exist of operations which all must be processed in a given way, so that the job is finished. There are time intervals in which no setup neither production can be undertaken.

The plant we analysed has 13 machines which are arranged in groups depending on the operations they can process.
Every day between 1,000 and 5,000 components are produced in the plant. The production order of a component is called a job. Each job comprises operations with precedence relations (operation chains) such as drilling, milling, and turning. The assignment and scheduling of operations on machines is done through the planning department, which consists of three people.
Before an operation can be processed on a machine, the machine has to be setup for the operation. The setup involves the change of tooling, the adjustment of the apparatus for the pieces which are to be processed, and the programming of the CNC control system with job-specific parameters. The time needed for a setup of an operation on a machine is not only dependent on the operation but also on the preceding operation done using the machine.
The setup can only be undertaken by the so-called setup operators. In this case, a setup operator is a person who has the necessary skills and tools for setting up machines. Each setup operator can, at any point in time, set up one machine at the most.
Setup operators and machines work in three shifts, seven days a week in the following time intervals: 6:00 a.m.–2:00 p.m., 2:00 p.m.–10:00 p.m., and 10:00 p.m.–6:00 a.m. In each shift, there are three setup operators available. Furthermore, every shift includes a break after four working hours for an hour.

Since penalties for late deliveries are high in this sector, the main goal for the company is the minimization of the total tardiness. Due to the complexity of the planning context and the limited computer support, the planning department generally schedules operations regarding their due date at the earliest possible time on a feasible machine. There is no planned schedule for the setup operators themselves. Instead, they set up the next available operation. If several operations can be chosen, they set up the operation, which was available first. Before going to develop the actual algorithm, we analysed the outcomes of the basic procedure of the planning department. While it seemed clear that no other procedure was basically possible for them to perform at the time, the results of the hitherto-used procedure of the planning department were less than satisfactory.

6.2 Problem definition

In what follows, we consider a set of machines $1, \ldots, M$. There are setup operators $h = 1, \ldots, R$ and jobs $j = 1, \ldots, J$. Each job consists of operations $i = (j, 1), \ldots, (n_j, j)$ which have to be processed in a given sequence (chain). For notational convenience we assume that operations are numbered accordingly. Each operation (i, j) requires processing time $p_{i,j}$ and is preceded by a setup of duration $s_{i,j}$. Pre-emption is allowed neither for operations nor for setups.

Depending on the operation, it can be processed on different machines. Binary parameter $v_{i,j,m}$ indicates whether operation (i, j) can be done on machine m or not. Each machine can process one operation at a time. For a schedule to be feasible, we require the number of overlapping setups not to be greater than the number of the setup operators R. The goal is to minimize total tardiness $\sum_{j=1}^{J} t_j$. Note that we can assume that each job follows its preceding setup without any idle time in between. We additionally consider a generalization of the problem setting described above. We incorporate breaks $a = 1, \ldots, A$. There can be no setup nor the processing of a job between the beginning of a break bs_a and the end of a break be_a. If a setup $s_{i,j}$ or processing of an operation $p_{i,j}$ has started before the break bs_a and crosses it while being carried out, the processing, respectively the setup is prolonged by be_a - bs_a.

Indices	
m	Machine index, $m = 1, \ldots,$ M
j	Job index, $j = 1, \ldots,$ J
i	Operation index, $i = 1, \ldots, n_j$
h	Setup operator index, $h = 1, \ldots, R$
a	Break index, a = 1, …, A
Parameters	
$p_{i,j}$	Processing time of operation (i,j)
$s_{i,j}$	Setup time of operation (i,j)
K	Adequate large number
$v_{i,j,m}$	Binary parameter which equals 1, if operation (i,j) can be done on machine m, else it equals 0
bs_a	Beginning of break a
be_a	End of break a
L_j	Delivery date of job (j)
Variables	
$\beta_{i,j,a}$	Binary variable which turns 1 if operation (i,j) would be processed during a break
$\eta_{i,j,a}$	Binary variable which turns 1 if operation (i,j) would be set up during a break
$\theta_{i,j,a}$	Binary variable which turns 1 if operation (i,j) would be setup or produced during a break
$\epsilon_{i,j,a}$	Binary variable which turns 1 if operation (i,j) starts before break a
$\gamma_{i,j,m}$	Binary variable which turns 1 if operation (i,j) is processed on machine m
$\delta_{i,j,l,k,m}$	Binary variable which turns 1 if operation (l,k) starts directly after operation (i,j) on machine m
$\alpha_{i,j,h}$	Binary variable which turns 1 if operation (i,j) is assigned to setup operator h, else it turns 0
$\sigma_{i,j,l,k,h}$	Binary variable which turns 1 if operation (l,k) starts directly after operation (i,j) and is assigned to setup operator h
$S_{i,j}$	Starting time of operation (i,j)
t_j	Tardiness of job j
C_{max}	Makespan
n_m	Number of tasks assigned to machine m
q_h	Number of setups assigned to setup operator h

Table 6.1: Notation job shop with parallel machines

6.2.1 Standard model formulation

$$min \quad \sum_{j=1}^{J} t_j \tag{6.1}$$

$$t_j \geq S_{n_j,j} + p_{n_j,j} + s_{n_j,j} - L_j \qquad \forall \quad j = 1, \ldots, J \tag{6.2}$$

$$S_{i+1,j} - p_{i,j} - s_{i,j} - S_{i,j} \geq 0 \qquad \forall \quad j = 1, \ldots, J, i = 1, \ldots, n_j - 1 \tag{6.3}$$

$$\sum_{m=1}^{M} \gamma_{i,j,m} = 1 \qquad \forall \quad j = 1, \ldots, J, i = 1, \ldots, n_j \tag{6.4}$$

$$\gamma_{i,j,m} \leq v_{i,j,m} \qquad \forall \quad j = 1, \ldots, J, i = 1, \ldots, n_j, m = 1, \ldots, M \tag{6.5}$$

$$S_{l,k} - (S_{i,j} + p_{i,j} + s_{i,j}) \geq (\delta_{i,j,l,k,m} - 1) \cdot K$$

$$\forall \quad j = 1, \ldots, J, l = 1, \ldots, J, i = 1, \ldots, n_j, k = 1, \ldots, n_l, m = 1, \ldots, M \tag{6.6}$$

$$\sum_{j=1}^{J} \sum_{i=1}^{n_j} \sum_{\substack{l=1 \\ (i,j \neq k,l)}}^{J} \sum_{k=1}^{n_l} \delta_{i,j,k,l,m} = \sum_{j=1}^{J} \sum_{i=1}^{n_j} \gamma_{i,j,m} - 1 \qquad \forall \quad m = 1, \ldots, M \tag{6.7}$$

$$\sum_{\substack{l=1 \\ (k,l \neq i,j)}}^{J} \sum_{k=1}^{n_l} \sum_{m=1}^{M} \delta_{i,j,k,l,m} \leq 1 \qquad \forall \quad j = 1 \ldots, J, i = 1, \ldots, n_j \tag{6.8}$$

$$\sum_{\substack{j=1 \\ (i,j \neq k,l)}}^{J} \sum_{i=1}^{n_j} \sum_{m=1}^{M} \delta_{i,j,k,l,m} \leq 1 \qquad \forall \quad l = 1, \ldots, J, k = 1, \ldots, n_l \tag{6.9}$$

$$S_{k,l} - (S_{i,j} + s_{i,j}) \geq (\sigma_{i,j,k,l,h} - 1) \cdot K$$

$$\forall \quad j = 1, \ldots, J, l = 1, \ldots, J, i = 1, \ldots, n_j, k = 1, \ldots, n_l, h = 1, \ldots, R \tag{6.10}$$

$$\sum_{h=1}^{R} \alpha_{i,j,h} = 1 \qquad \forall \quad j = 1,\ldots,J, i = 1,\ldots,n_j \quad (6.11)$$

$$\sum_{j=1}^{J}\sum_{i=1}^{n_j} \sum_{\substack{l=1 \\ (i,j\neq k,l)}}^{J} \sum_{k=1}^{n_l} \sigma_{i,j,k,l,h} = \sum_{j=1}^{J}\sum_{i=1}^{n_j} \alpha_{i,j,h} - 1 \qquad \forall \quad h = 1,\ldots,R \quad (6.12)$$

$$\sum_{\substack{l=1 \\ (k,l\neq i,j)}}^{J} \sum_{k=1}^{n_l}\sum_{h=1}^{R} \sigma_{i,j,k,l,h} \leq 1 \qquad \forall \quad j = 1,\ldots,J, i = 1,\ldots,n_j \quad (6.13)$$

$$\sum_{\substack{j=1 \\ (i,j\neq k,l)}}^{J} \sum_{i=1}^{n_j}\sum_{h=1}^{R} \sigma_{i,j,k,l,h} \leq 1 \qquad \forall \quad l = 1,\ldots,J, k = 1,\ldots,n_l \quad (6.14)$$

$$S_{i,j} \geq 0 \qquad \forall \quad j = 1,\ldots,J, i = 1,\ldots,n_j \quad (6.15)$$

$$\gamma_{i,j,m}, \sigma_{i,j,k,l,h}, \delta_{i,j,k,l,m}, \alpha_{i,j,h} \in \{0,1\}$$

$$\forall \quad j = 1,\ldots,J, l = 1,\ldots,J, i = 1,\ldots,n_j, k = 1,\ldots,n_l, m = 1,\ldots,M$$

$$h = 1,\ldots,R \quad (6.16)$$

Objective function (6.1), along with constraint (6.2), represents the goal to minimize total tardiness. Tardiness is checked for each last operation of a job n_j, j (6.2) and summed up (6.1). Constraint (6.3) defines that the successor operation of a job $S_{i+1,j}$ can only start after the predecessor operation is finished $(S_{i,j} + p_{i,j} + s_{i,j})$. Constraint (6.4) represents that each operation (i,j) is on exactly one machine m $\gamma_{i,j,m}$. Constraint (6.5) ensures that every operation (i,j) can only be assigned to a machine which can handle this operation $v_{i,j,m}$. Constraint (6.8) and (6.9) ensure that each operation has no more than one direct predecessor (6.8) and one successor (6.9). Constraint (6.7) prohibits sub cycles as the number of successor predecessor relationships on each machine is limited to the number of jobs assigned to the machine - 1. Constraint (6.6) makes sure that if an operation (l,k) follows operation (i,j) on machine m $\delta_{i,j,l,k,m}$ the start time of $l, kS_{l,k}$ must be greater or equal to the end of the predecessor operation on the machine ($S_{i,j} + p_{i,j} + s_{i,j}$). Constraints (6.10)–(6.14) are made in the same way as constraints (6.4), (6.6) - (6.9). The difference is that here, operations must be assigned to a setup operator h. If an operation (k,l) directly follows operation (i,j) on setup operator h ($\sigma_{i,j,l,k,h}$). (l,k) can only start $S_{k,l}$ after the start of $S_{i,j}$ and the finish of its setup $s_{i,j}$. Constraints (6.15), (6.16) define the variable's domains.

6.2.2 Extension

$$S_{i+1,j} - p_{i,j} - s_{i,j} - S_{i,j} - (be_a - bs_a) \cdot \beta_{i,j,a} \geq 0$$

$$\forall \quad j = 1, \ldots, J, i = 1, \ldots, n_j - 1, a = 1, \ldots, A \tag{6.17}$$

$$S_{l,k} - (S_{i,j} + p_{i,j} + s_{i,j}) - (be_a - bs_a) \cdot \beta_{i,j,a} \geq (\delta_{i,j,l,k,m} - 1) \cdot K$$

$$\forall \quad j = 1, \ldots, J, l = 1, \ldots, J, i = 1, \ldots, n_j, k = 1, \ldots, n_l, m = 1, \ldots, M$$

$$a = 1, \ldots, A \tag{6.18}$$

$$S_{k,l} - (S_{i,j} + s_{i,j}) - (be_a - bs_a) \cdot \beta_{i,j,a} \geq (\sigma_{i,j,k,l,h} - 1) \cdot K$$

$$\forall \quad j = 1, \ldots, J, l = 1, \ldots, J, i = 1, \ldots, n_j, k = 1, \ldots, n_l, h = 1, \ldots, R$$

$$a = 1, \ldots, A \tag{6.19}$$

$$bs_a \leq S_{i,j} - K \cdot \zeta_{i,j,a}$$

$$\forall \quad j = 1, \ldots, J, i = 1, \ldots, n_j, a = 1, \ldots, A \tag{6.20}$$

$$bs_a > S_{i,j} + s_{i,j} - K \cdot \eta_{i,j,a}$$

$$\forall \quad j = 1, \ldots, J, i = 1, \ldots, n_j, a = 1, \ldots, A \tag{6.21}$$

$$bs_a > S_{i,j} + s_{i,j} + p_{i,j} - K \cdot \theta_{i,j,a}$$

$$\forall \quad j = 1, \ldots, J, i = 1, \ldots, n_j, a = 1, \ldots, A \tag{6.22}$$

$$\zeta_{i,j,a} + \eta_{i,j,a} < 2 + K \cdot \beta_{i,j,a}$$

$$\forall \quad j = 1, \ldots, J, i = 1, \ldots, n_j, a = 1, \ldots, A \tag{6.23}$$

$$\zeta_{i,j,a} + \theta_{i,j,a} < 2 + K \cdot \beta_{i,j,a}$$

$$\forall \quad j = 1, \ldots, J, i = 1, \ldots, n_j, a = 1, \ldots, A \tag{6.24}$$

$$min \quad C_{max} \tag{6.25}$$

$$C_{max} \geq S_{n_j,j} + p_{n_j,j} + s_{n_j,j} \qquad \forall \quad j = 1, \ldots, J \tag{6.26}$$

When the goal is to minimize makespan, constraints (6.24) and (6.25) replace constraints (6.1) and (6.2). Constraints (6.17), (6.18), (6.20), (6.22), and (6.24) incorporate break times in the non-overlapping of operations, whereas constraints (6.19), (6.20), (6.21), and (6.23) incorporate breaks in the non-overlapping of setups. As in Section 5.2 for the explanation of t, there is no benefit of trying to explain the functioning of the incorporation of breaks in the model in contrast to the linear mathematical formulation. Therefore, only the general functioning is explained. For each operation (i,j), the operation is not allowed to be processed or setup during a break. If it runs into a break, either the setup or the processing is prolonged by the duration of the break. Therefore, it must be checked for each start time, end of setup time and end of processing time, if a break is hit in between such that the operation is then prolonged.

6.3 Methods

In Chapter 6 the algorithms from Chapter 5 are continued, which means that here also two TSs and two GAs are applied as well as their combination in the VNS. This allows us to compare the performance between the metaheuristics

better. Therefore, only the differences are explained. In other words, there is no specific description of the algorithms, as they are similar to the ones presented in Section 5.3. It must be acknowledged, though, that the programming for the algorithms here is much closer to the programming of the algorithms in Chapter 4 due to the form of the solution representation. Both chapters have chains in their problem setting. In Chapter 4 there are job chains, here we have operation chains.
The solution representation is also the main difference to the algorithms from Chapter 5. A solution can be represented by a sequence of jobs together with the name of the machine they are assigned to, similar to the solution representation from Chapter 4. This form of solution is sufficient for the problem structure described in this chapter, since the sequence of operations belonging to the same job cannot be changed. Therefore, the sequence "J2 M1, J3 M2, J2 M0, J1 M3" would mean setting up machine 1 with the first operation of job 2, then machine 2 with the first operation of job 3, then machine zero with the second operation of job 2, and so on. The sequence also indicates to which machine a setup operator should move next. It would also be possible to represent a solution by the sequence of operations to be set-up. The sequence "$O_{1,1}M1, O_{1,2}M2, O_{2,2}M0, O_{2,1}M3$" would mean to setup first machine one with $operation_{1,1}$, then machine two with $operation_{1,2}$, then machine zero with $operation_{2,2}$, and so on. The problem structure allows the transfer of each representation form to each other at any time. In both cases, the search procedures are similar. Nevertheless, the search has a higher efficiency in the job representation. This is because it does not have to check for the validity or repair a found solution as the job representation does not show any movement of operations within a job.
Starting solutions are also created in a two-stage approach. As there is no explicit machine influence on the operations, such as a machine-dependent running time, less heuristics for the creation of the starting solution are used. For the objective to minimize total tardiness, we need to assign dummy due dates to the operations of a job to make the decision in which sequence operations will be processed in the sort-heuristics. The due dates are assigned as follows:

$$Duedate_{i,j} = Duedate_j - (\sum_{x=i+1}^{n_j} s_{x,j} + p_{x,j}).$$

First stage heuristics:

1. Schedule operations in non-descending order with regard to their due date as early as possible on the machine which can start the operation first. Repeat until all operations and jobs are processed.

2. Schedule operations in non-descending order with regard to their slack time as early as possible on the machine which can finish the operation first. Repeat until all operations are processed.

3. Schedule operations with regard to the operation chain in non-descending order of the number of machines which they can be served by, on the

machine which can handle the least amount of operations, as early as possible. If multiple operations can be served by the same number of machines, schedule the operations in non-descending order of their due dates. Repeat until all operations are processed.

Second stage heuristics:

The second stage procedure is closely related to the one from Section 5.3. The reason is again comparability of the algorithms across the chapters. When a setup operator becomes available, one of the following strategies is used by the heuristics to find the next assignment for the setup operator.

1. Of all operations allowed to be processed with regard to the operations-chains setup the operation with the earliest due date. Repeat until all operation are processed.

2. Of all operations allowed to be processed with regard to the operations-chains setup the operation with the lowest slack time. Repeat until all operation are processed.

3. Of all operations allowed to be processed in regard to the operations-chains setup the operation with the shortest setup time. Repeat until all operations are processed.

4. Of all operations allowed to be processed with regard to the operations-chains, setup the operation with the longest processing time. Repeat until all operations are processed.

5. Of all operations allowed to be processed with regard to the operations-chains, setup the operation from the chain with the largest sum of remaining setup and processing times. Repeat until all operation are processed.

6.4 Computational Results

As in Sections 4.4 and 5.4, the algorithms are tested on an Intel i7 Quadcore 1,7 GHz, 8 GB memory computer. The programming language is JAVA. There are randomly generated instances (6.1), instances of which the best solution is known (6.2) and there are instances with real-life data received from the company (6.3).

6.4.1 Random instances

For each test case (Table 6.2), three instances are solved with randomly generated numbers. The instances are created in the same way as in Subsections 4.4.1 and 5.4.1 in such a way that they not only reflect the production situation of the company in focus but also provide information about the performance of the algorithms in larger instances. The cases are built as follows:

1. For each operation, randomly pull a number between 0 and 100. If the number is greater than 50, the operation can run on the machine. If the operation cannot run on any machine after the procedure, uniform randomly assign it to one machine.

2. For each operation, assign it uniform randomly a position in a random job.

3. For each operation, the operation can run on uniform randomly assigned setup time between 0 and 3000.

4. For each operation, the operation can run on uniform randomly assigned processing time between 0 and 3000.

5. Due dates for the jobs. The due dates are calculated as follows: Sum up all production and setup times of the operations within jobs to a variable s. Multiply s with a positive factor to s*.[12] For each job assign uniform randomly a due date between 0 and s*.

Furthermore, some cases consider

6. Breaks. Setup operators and machines are available 24 hours whereby a break of one hour for both needs to be undertaken every eight hours.

The computing time is limited to 360 seconds per instance and method. Between 10—25 machines, 10–50 jobs, 100–500 operations, breaks and no breaks, and 1–5 setup operators are considered in the instances (Tables 6.2, 6.3). At first, an attempt was made to compare the results with the lower bounds, which were created by relaxing some and later all integer variables in the MIP. As in Subsection 5.4.1, the lower bounds were so weak that we compare the relative improvements of the methods. Cases 91–114 have the objective to minimize the makespan. As in Chapter 4, the makespan here is also defined as the time between the start of the first setup until the end of production for the last job including the break times. Here cases 91–102 only consider the simplified problem, without breaks. Cases 103–114 include breaks. Cases 115–138 have the objective to minimize total tardiness. Here cases 115–126 consider only the simplified problem, without breaks. Cases 127–138 consider all extensions as well.

Cases 91–102
The first thing to notice in Table 6.3 (random instances, makespan, no breaks) is, that the higher the number of setup operators is, the higher is the enhancement for the metaheuristics in contrast to the starting solution. It seems, that the constraint of the number of setup operators limits the performance of the metaheuristics. In general, it has to be acknowledged that the gains are in many cases only very little. Especially in the large instances, only minor improvements can be achieved.

[12]In our case we choose 1,25 as a factor.

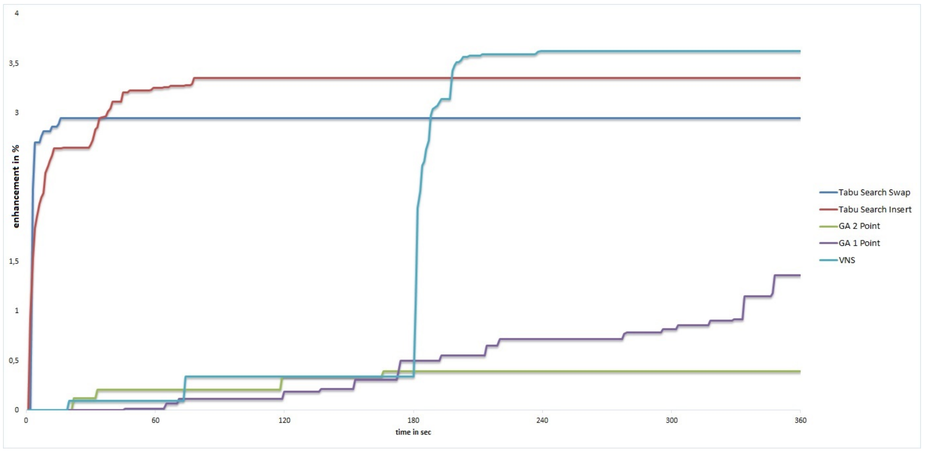


Figure 6.1: Diagram Case 95

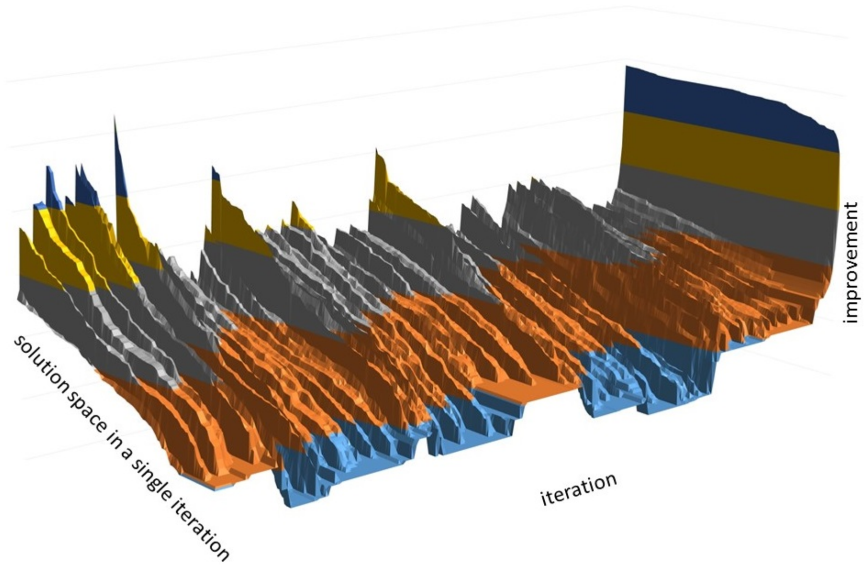


Figure 6.2: Neighbourhood TS Case 95

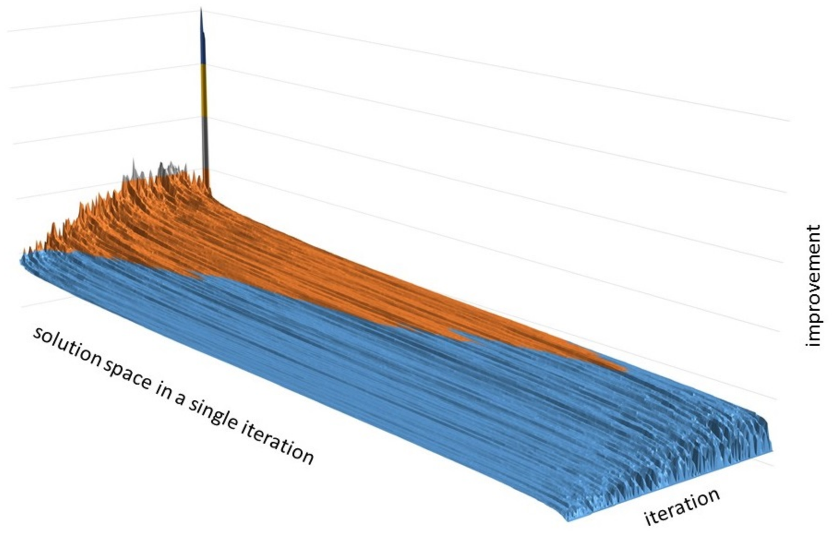


Figure 6.3: Neighbourhood GA Case 95

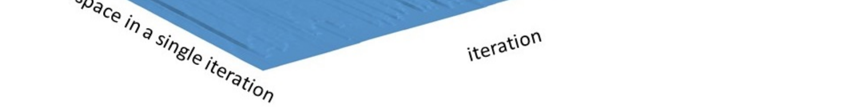


Figure 6.4: Neighbourhood VNS Case 95

Case	Machines	Jobs	Operations	Setup time	Processing time	setup overlaps	breaks
91/115	10	10	100	U(0, 3000)	U(0, 3000)	1	/
92/116	10	10	100	U(0, 3000)	U(0, 3000)	3	/
93/117	10	10	100	U(0, 3000)	U(0, 3000)	5	/
94/118	10	20	200	U(0, 3000)	U(0, 3000)	1	/
95/119	10	20	200	U(0, 3000)	U(0, 3000)	3	/
96/120	10	20	200	U(0, 3000)	U(0, 3000)	5	/
97/121	25	25	250	U(0, 3000)	U(0, 3000)	1	/
98/122	25	25	250	U(0, 3000)	U(0, 3000)	3	/
99/123	25	25	250	U(0, 3000)	U(0, 3000)	5	/
100/124	25	50	500	U(0, 3000)	U(0, 3000)	1	/
101/125	25	50	500	U(0, 3000)	U(0, 3000)	3	/
102/126	25	50	500	U(0, 3000)	U(0, 3000)	5	/

Table 6.2: Random instances 91–102 makespan / 11–126 total tardiness

The GA 1-point crossover shows a relatively strong performance in the small cases in contrast to the other methods. It shows the best performance of all metaheuristics in cases 91–93. Overall, it is in all but three cases better than the 2-point crossover. In three iterations it does not find a better solution at all, similar to the 2-point crossover. As in Section 5.4., the GAs have problems with large instances. The GA clearly eliminates the worst solutions in the beginning, and has a more homogenous population with increasing running time (Figure 6.3) (Neighbourhood GA Case 95). In connection with Figure 6.1 (Diagram Case 95), it can also be stated, that continuously new best solutions are found. Therefore, it is expected, that with further running time, the solution quality will further enhance.

The TS Insert is better than the TS Swap in all but one case. In four cases it has the best solution among all heuristics, while the TS Swap has the best solution among the heuristics only in one case. Furthermore, the TS Swap surprisingly finds no better solution at all in the first three cases, while the TS Insert, as the best metaheuristic in these cases shows improvements by as much as 11,13%. Figure 6.1 (Diagram Case 95) demonstrates how both TSs have a strong performance in the beginning, before they reach after little running time, an area, from which they find no better solution. Figure 6.2 demonstrates clearly how different the neighbourhood for the TS Swap looks. After a strong incline in the solution quality at the beginning, with a neighbourhood which has more or less homogenous solution values in an iteration, the neighbourhood gets much more diverse with the running time.

It can also be seen that the solutions of the TS deteriorate over time before they come back to an area with good solutions. That means, that the TS has

	GA 2 point	GA 1 point	TS Swap	TS Insert	VNS
91	0,92%	1,88%	0,00%	0,74%	1,71%
92	3,92%	12,10%	0,00%	7,09%	6,50%
93	5,51%	15,10%	0,00%	11,13%	5,79%
94	0,46%	0,68%	0,90%	1,03%	1,07%
95	0,39%	1,37%	2,95%	3,35%	3,63%
96	1,47%	4,49%	4,69%	4,36%	4,29%
97	0,29%	0,49%	0,16%	0,61%	0,67%
98	0,37%	0,41%	1,20%	1,66%	1,24%
99	1,30%	0,72%	2,16%	2,53%	2,58%
100	0,00%	0,09%	0,03%	0,16%	0,16%
101	0,00%	0,00%	0,20%	0,33%	0,28%
102	0,10%	0,04%	0,43%	0,62%	0,42%

Table 6.3: Enhancement to the starting solution for cases 91–102

either found the optimum solution, is closer to the optimum solution, maybe because the starting heuristics can find better solutions with a higher degree of freedom in the problem, or that the TS is in such a difficult area, from which it can hardly come closer to the optimum. Then the following hypothesis might hold. The higher the degree of freedom in the problem, the larger the solution space and the more difficult it becomes for the TSs to come to a good area and get near the optimum.

The method which has the best results overall is the VNS. It achieved in comparison to the other methods the best results in five of the 12 test cases. Furthermore, it also showed the biggest consistency in achieving gains. In the cases where it did not have the best results, it was close to the best results. The VNS, again, follows the graph of the currently used metaheuristic (Figure 6.1 (Diagram Case 95)). While the performance is low in the beginning, the switch to the TS methods shows a strong increase in the solution quality such that the VNS finds the best solution among all metaheuristics. Figure 6.4 shows the neighbourhood of the VNS. While the beginning is akin to the structure of the GA, the structure when the VNS makes use of the TS methods appears to be different from the structure when the TS is used alone. An explanation is that the solution given from the GAs to the TSs leads the latter to a different neighbourhood region, which is more homogenous and provides also better overall solutions as the results suggest.

Cases 115–126
Table 6.4 (random instances, total tardiness, no breaks) shows, as the other tables in which the best solution is not known in this chapter, the enhancement to the starting solution for cases 115–126. The GAs show marginal improvements. The 1-point crossover method is better than the 2-point in 10 cases. In the

	GA 2 point	GA 1 point	TS Swap	TS Insert	VNS
115	3,81%	6,92%	29,67%	29,08%	28,18%
116	2,72%	9,64%	19,90%	19,48%	19,25%
117	3,46%	11,66%	18,66%	18,35%	17,50%
118	2,03%	3,19%	23,67%	25,98%	20,16%
119	0,16%	0,34%	16,06%	17,29%	14,65%
120	0,24%	1,06%	14,45%	13,10%	12,87%
121	1,97%	1,87%	13,11%	13,15%	10,49%
122	0,41%	0,46%	14,11%	14,18%	9,96%
123	0,24%	0,83%	12,25%	13,31%	8,74%
124	1,50%	1,25%	1,48%	1,48%	2,73%
125	0,01%	0,08%	1,49%	1,49%	1,16%
126	0,00%	0,12%	1,88%	1,88%	1,33%

Table 6.4: Enhancement to the starting solution for cases 115–126

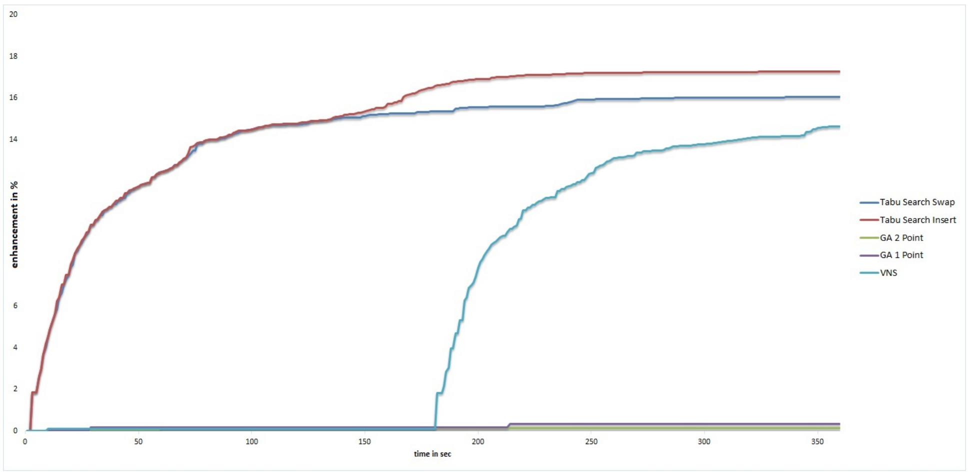


Figure 6.5: Diagram Case 119

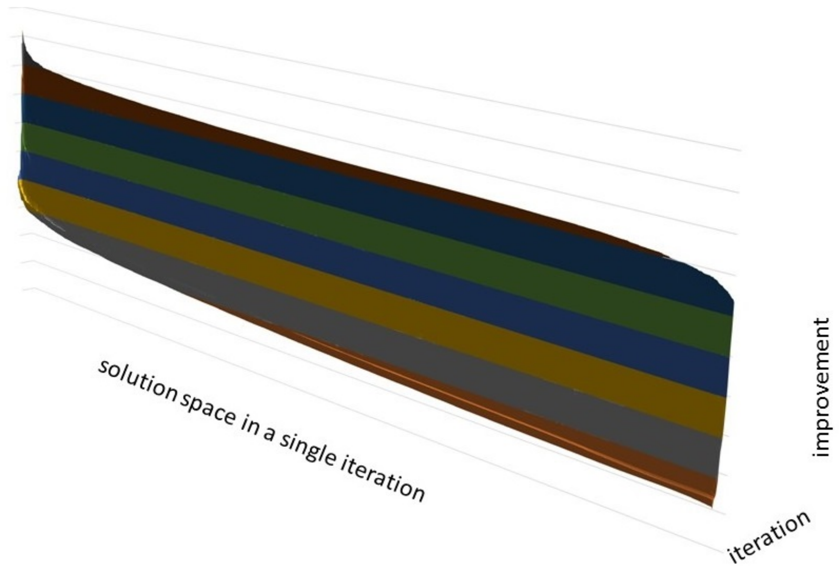


Figure 6.6: Neighbourhood TS Case 119

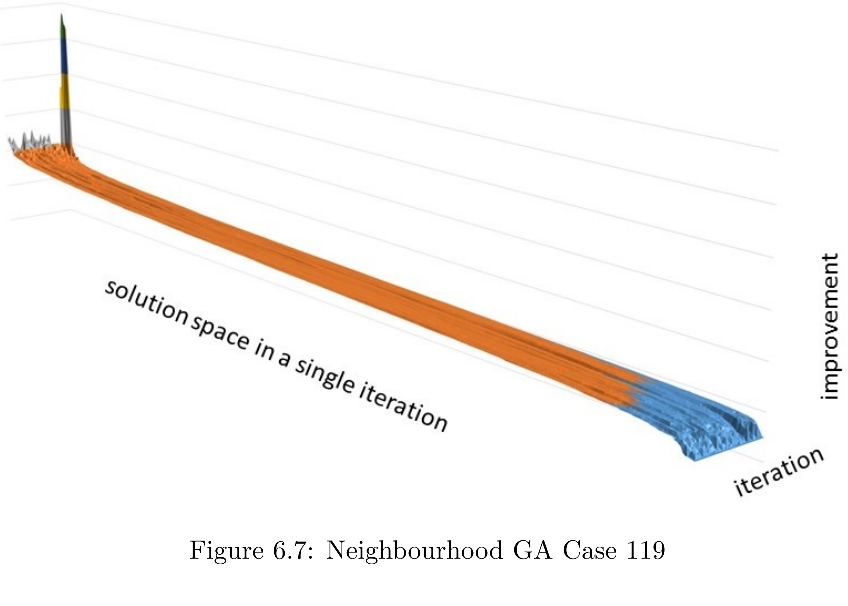


Figure 6.7: Neighbourhood GA Case 119

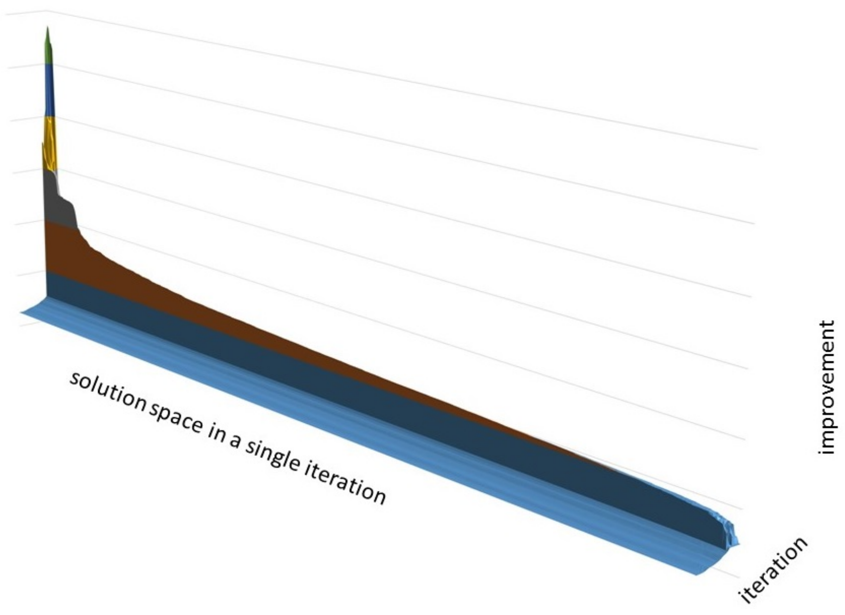


Figure 6.8: Neighbourhood VNS Case 119

cases, in which 1-point is better, there is a large gap between these two. Figure 6.5 (Diagram Case 119) clearly shows that the GAs rarely show improvement. This can also be seen in Figure 6.7. The GA does not show much improvement in the populations except the elimination of the weakest solutions right at the beginning. In contrast to Chapter 5, this result is surprising. While in 5 the GA has some enhancement, especially when there is an increase in the number of setup operators, here this is not true. The performance is overall low but slightly better when the number of setup operators is also low.

The TSs on the other hand perform much better relatively. The TS Swap finds the best solution among all methods in six cases. Two of the best solutions are shared with the TS Insert. Apart the TS Insert finds the best solution in five further cases. As the solutions are close, it can be stated, that both perform equally. The TSs in Figure 6.5 (Diagram Case 119) have a strong improvement at the beginning but then the improvement over time declines. Figure 6.6 also shows a strong improvement for the TS. But it is problematic that only few iterations have been run.

There are no surprises at least for the VNS search. It has only the best solution in one case, but is always close. In Figure 6.5, the VNS follows the path of the TSs after 180 seconds but does not have sufficient time to come to the same level as they are. In 6.8 the VNS also follows the forms of their respective methods.

Although in the smaller cases the improvement is at maximum 29,67% (case 115), it can be seen that the bigger the solution space is, the lower the improvements for the methods. Interestingly, in contrast to the other test cases seen in Section 5.4 and here so far, an increase in the number of setup operators does not lead to solution enhancements. As the starting solutions, which do not incorporate setup operators at all, it is possible, that they perform better in the cases 115–126 and such the metaheuristics have less room for improvement. Therefore, this could indicate that they are close to the lower bound. Nevertheless, from Figures 6.6 - 6.8 (neighbourhoods for the GA, TS and VNS in case 119), it can be seen that for all methods, a much smaller number of iterations can be solved than in the other chapters. Hence, the degree of freedom might be problematic for the methods.

Case	Machines	Jobs	Operations	Setup time	Processing time	setup overlaps	breaks
103/127	10	10	100	U(0, 3000)	U(0, 3000)	1	yes
104/128	10	10	100	U(0, 3000)	U(0, 3000)	3	yes
105/129	10	10	100	U(0, 3000)	U(0, 3000)	5	yes
106/130	10	20	200	U(0, 3000)	U(0, 3000)	1	yes
107/131	10	20	200	U(0, 3000)	U(0, 3000)	3	yes
108/132	10	20	200	U(0, 3000)	U(0, 3000)	5	yes
109/133	25	25	250	U(0, 3000)	U(0, 3000)	1	yes
110/134	25	25	250	U(0, 3000)	U(0, 3000)	3	yes
111/135	25	25	250	U(0, 3000)	U(0, 3000)	5	yes
112/136	25	50	500	U(0, 3000)	U(0, 3000)	1	yes
113/137	25	50	500	U(0, 3000)	U(0, 3000)	3	yes
114/138	25	50	500	U(0, 3000)	U(0, 3000)	5	yes

Table 6.5: Random instances 103–114 makespan / 127–138 total tardiness

Cases 103–114

The results from cases 103–114 (Table 6.6)(random instances, makespan, breaks) look similar to the results 91–102 (Table 6.3)(random instances, makespan, no breaks). Except for the smallest instances, no significant improvement can be seen.

In the smallest instances (103–105) the GAs are able to achieve the best results of all methods. The performance in the larger instances is, in many cases, close to zero. When the two GAs are compared, the 1-point crossover performs much better than the 2-point. It finds better solutions in eight of the 12 cases. Figure 6.9 (Diagram Case 107) shows a typical curve for the best solutions found. Both GAs nearly increase linear. The neighbourhood for the GA (Figure 6.11) shows as in Figure 6.7 (neighbourhood for the GA in case 119) a diverse solution quality in a single iteration.

	GA 2 point	GA 1 point	TS Swap	TS Insert	VNS
103	1,04%	1,94%	0,65%	1,75%	1,88%
104	3,76%	11,26%	0,57%	6,67%	7,98%
105	4,68%	16,46%	0,00%	6,73%	9,97%
106	0,71%	1,09%	1,22%	0,92%	1,26%
107	0,77%	1,79%	3,38%	3,21%	3,33%
108	1,63%	2,17%	7,38%	5,12%	5,97%
109	0,12%	0,18%	0,12%	0,24%	0,29%
110	0,76%	0,31%	0,90%	1,55%	1,29%
111	0,76%	0,68%	1,96%	3,09%	2,12%
112	0,06%	0,17%	0,07%	0,22%	0,18%
113	0,03%	0,01%	0,27%	0,37%	0,35%
114	0,00%	0,01%	0,72%	0,78%	0,72%

Table 6.6: Enhancement to the starting solution for cases 103–114

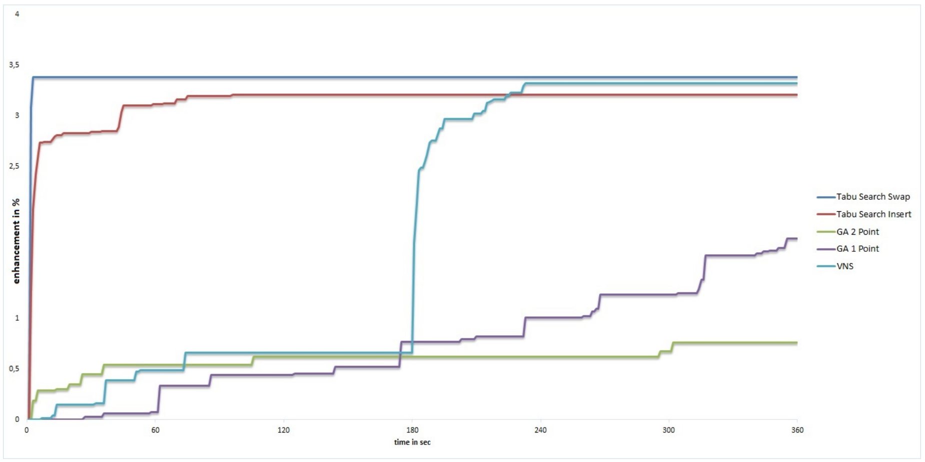


Figure 6.9: Diagram Case 107

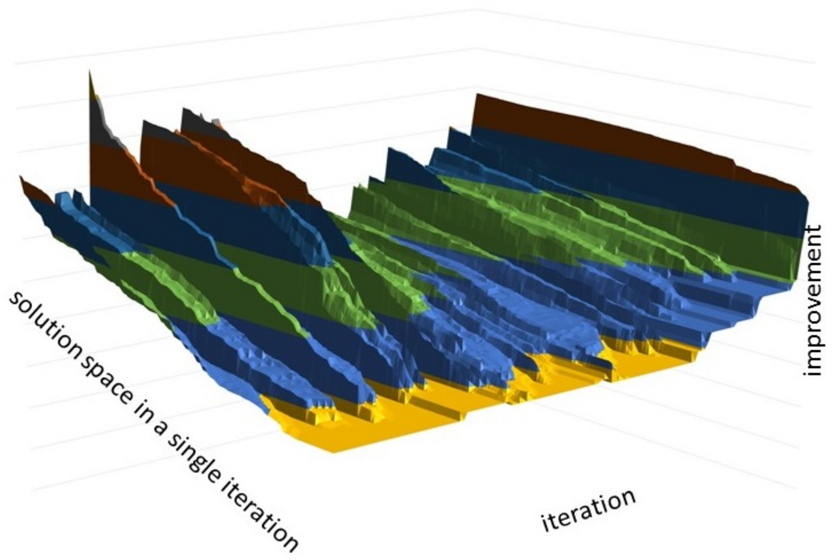


Figure 6.10: Neighbourhood TS Case 107

The TS Insert is better than the TS Swap in nine cases. In five of the nine cases it finds the best solution among all heuristics. The TS Swap is only better than the former in three cases. In one of the three cases (108) does it find the best solution among all methods. In Figure 6.9 (Diagram Case 107) it can be seen that the VNS has an incredible start at the beginning, but then is not able to find further enhancements. Figure 6.10 does not help neither. The neighbourhood seems so rough, that no explanation can be given.

While the VNS is better than all other methods in only one case. It is the second best in the other 11 cases. In Figure 6.9, the VNS, as seen before, switches to the TS methods after 180 seconds and also gains a strong performance incline before being stuck at a certain level. Figure 6.12 shows the typical performance of the VNS, we have seen before. The much more diverse neighbourhood in the GAs switches to a flat area when the TS methods follow. The explanation from Figure 6.4 (Neighbourhood VNS Case 95) that the GA methods lead the TSs to a more homogenous neighbourhood can still be applied.
Nevertheless, also here the overall weak performance of the metaheuristics cannot be answered.

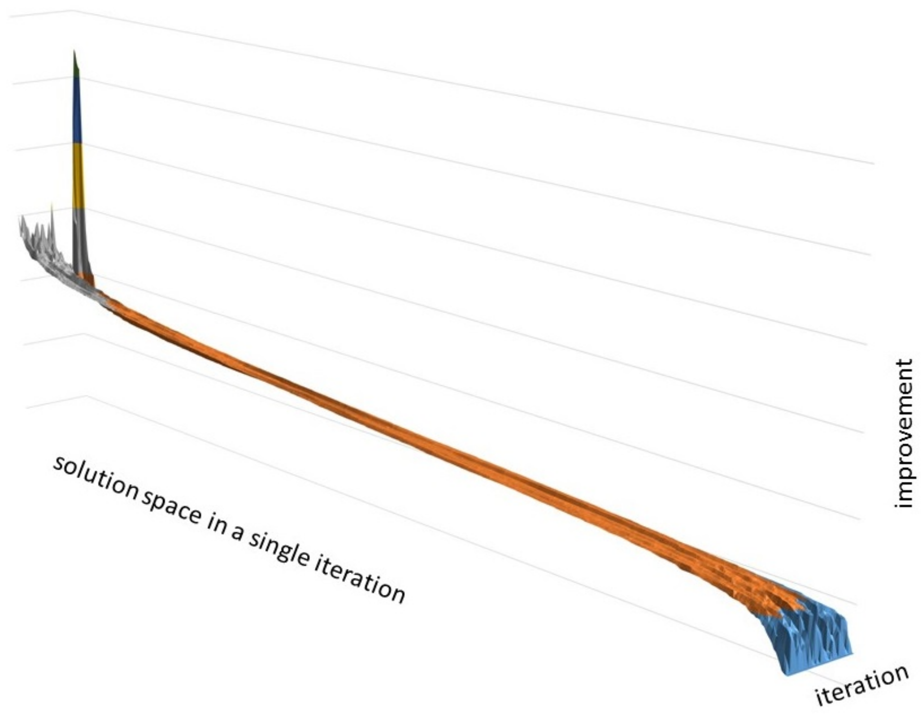


Figure 6.11: Neighbourhood GA Case 107

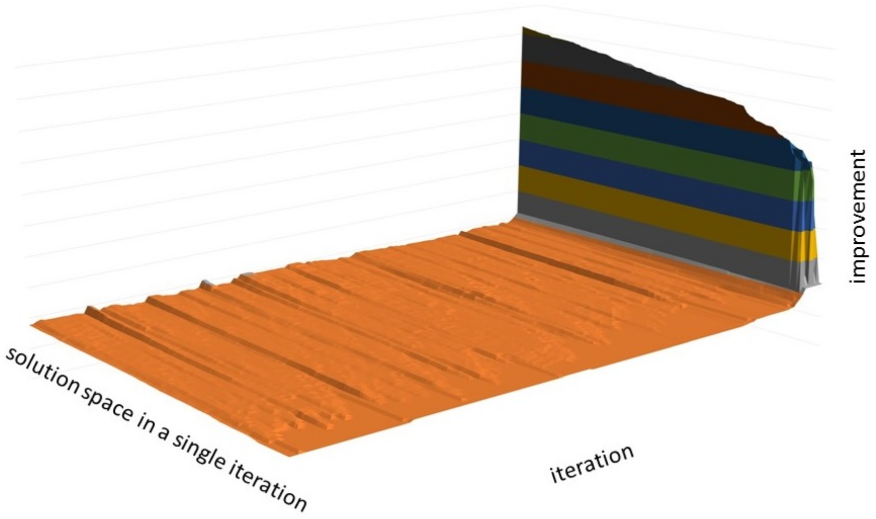


Figure 6.12: Neighbourhood VNS Case 107

Cases 127–138
The results of cases 127–138 (Table 6.7(random instances, total tardiness, breaks)) seem similar to the results from 115–126 (6.4(random instances, total tardiness, no breaks)). There are robust performance gains in the smaller instances and only little gains in the larger instances.

Both GAs perform relatively weakly. They are not able to find the best solution among all methods in any case. The maximum improvement to the starting solution is in any case 10,00%. In five of the 12 cases, one or both methods provide less than 1% improvement.
Figure 6.13 shows the same progression for the heuristics as Figure 6.5 (Diagram Case 119). The GAs nearly show no gains. Furthermore, from Figure 6.15 (Neighbourhood GA Case 131) the GA seems to have, as in Figure 6.7 (Neighbourhood GA Case 119), problems to converge its diverse population. It seems that the proportions of solutions in the different solution levels stay the same.

The TSs on the other hand, have a strong performance, especially in the smaller cases. They improve the solution by as much as 29,85%. In eight cases the TS Insert finds the best solution among all. In further three cases the TS Swap has the best solution. Figure 6.13 shows furthermore that both TSs keep improving the solutions over time. The enhancement per time unit decreases, however. Figure 6.14 shows a homogenous neighbourhood for the TS in the same iteration. As in case 119, the TS improves the solutions significantly from the starting solution.

Although the VNS only finds the best solution in one case among the heuristics used, it is always close. Figure 6.13 shows that the VNS gets its gains, when switched to the TS methods. Figure 6.16 (Neighbourhood VNS Case 131) shows the problems the GAs have even more clearly. While the number of the weakest solutions decreases, the figure shows, that until the VNS switches to the TSs method, there are no visible gains for the best solutions in the GAs. The neighbourhood in a single iteration for the VNS is much more homogenous and leads also to a clear solution improvement.
The difference in performance is interesting for the different objective functions. The neighbourhood structure also seems related to this. When the neighbourhood is more homogenous in a single iteration, overall the methods can find better solutions.

	GA 2 point	GA 1 point	TS Swap	TS Insert	VNS
127	4,09%	7,11%	29,79%	29,85%	29,09%
128	1,51%	7,71%	17,55%	16,50%	16,15%
129	3,95%	10,00%	15,94%	16,50%	15,30%
130	1,83%	2,44%	24,79%	24,99%	17,50%
131	0,18%	0,39%	17,15%	17,39%	14,24%
132	0,71%	0,02%	12,50%	12,48%	10,92%
133	2,92%	2,99%	14,66%	14,79%	12,16%
134	0,17%	0,13%	13,42%	13,87%	10,33%
135	0,09%	0,54%	10,67%	10,24%	8,81%
136	1,63%	1,71%	1,38%	1,38%	2,85%
137	0,03%	0,01%	1,35%	1,35%	1,22%
138	0,01%	0,00%	1,71%	1,71%	1,13%

Table 6.7: Enhancement to the starting solution for cases 127–138

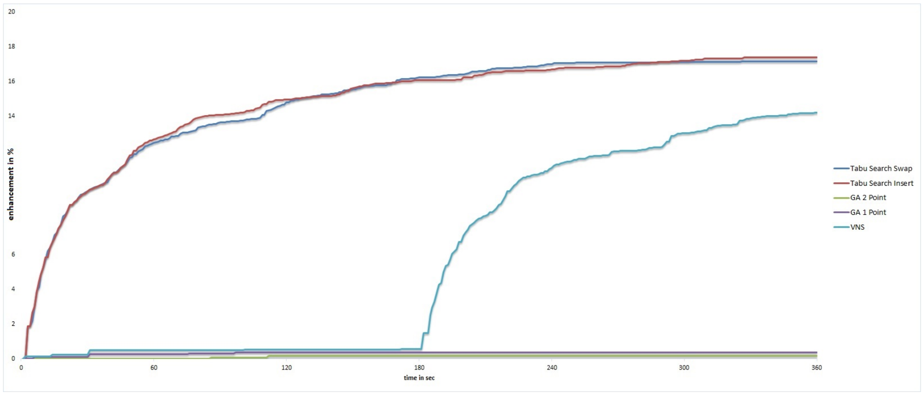


Figure 6.13: Diagram Case 131

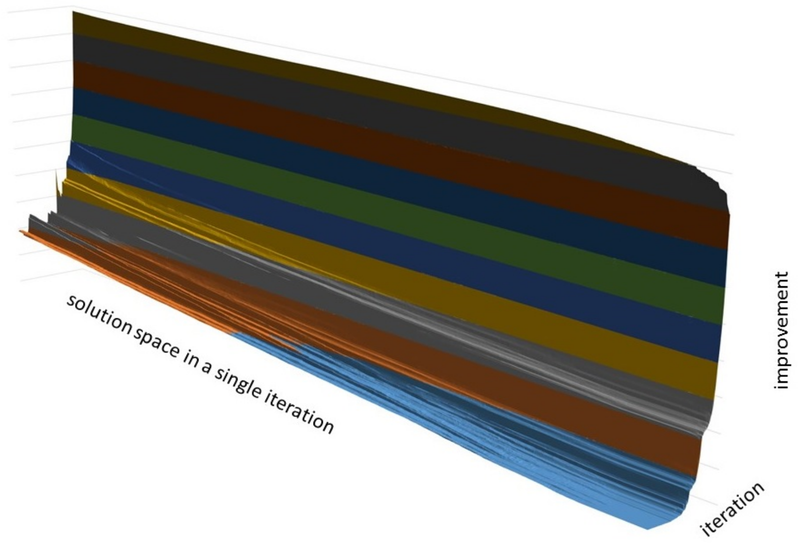


Figure 6.14: Neighbourhood TS Case 131

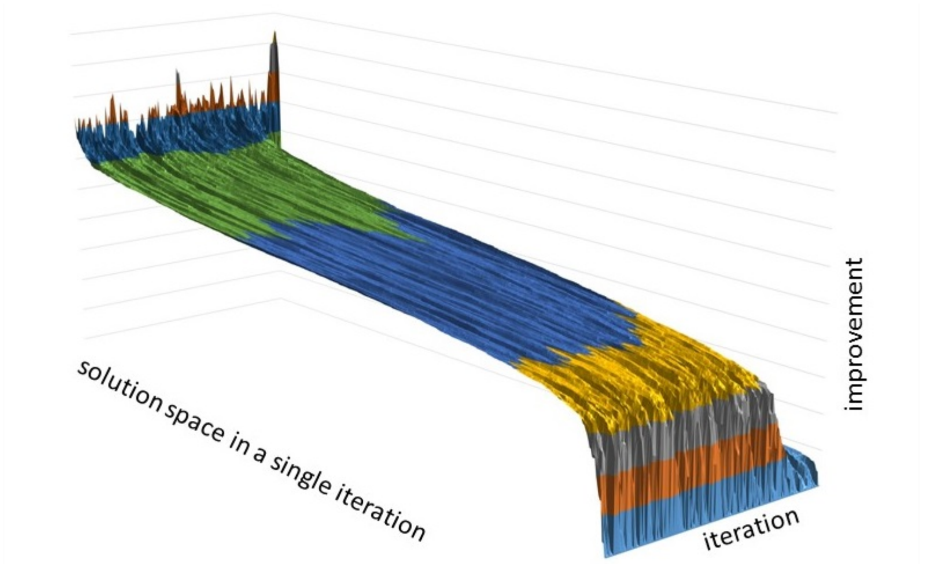


Figure 6.15: Neighbourhood GA Case 131

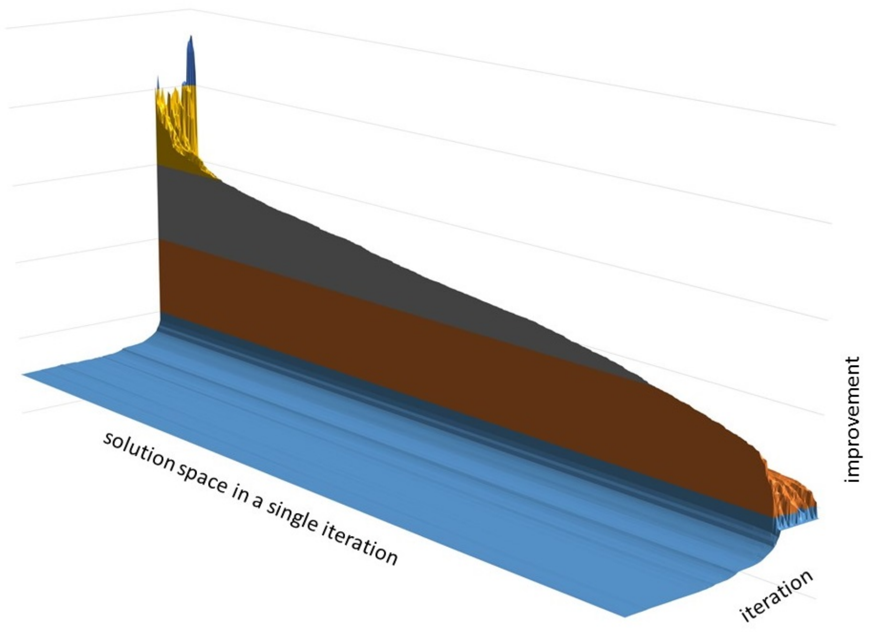


Figure 6.16: Neighbourhood VNS Case 131

6.4.2 Instances with known best solution

As in the other chapters, also in this problem context, the lower bounds only have limited significance. Hence, only the performance to the starting solution was measured and not the gap to the lower bounds. Therefore, again, instances are generated in such a way that the best solution is known. The structure of the generation of instances is related to the structure of the generation of instances in Subsection 5.4.2. Therefore, here also there is an optimal schedule which gives some machines as well as setup operators idle time. Cases 139–150 have the objective to minimize makespan. Cases 151–162 have the objective to minimize total tardiness. Instances for the makespan minimization problem are generated as follows:[13]

1. Operation and machine setup times are 0 in the first step. The maximum available capacity of all machines between 0 and t is h. On each machine one operation is added in such a way that each machine has a workload of 100% between 0 and t. The workload of all machines between 0 and t then stands at h time units. Given that the workload is 100% of the available capacity, the schedule must be optimal. (Figure 6.17)

[13]Note that the minimization of total tardiness is an enhancement of this makespan minimization generation procedure and thus described at the end of this procedure.

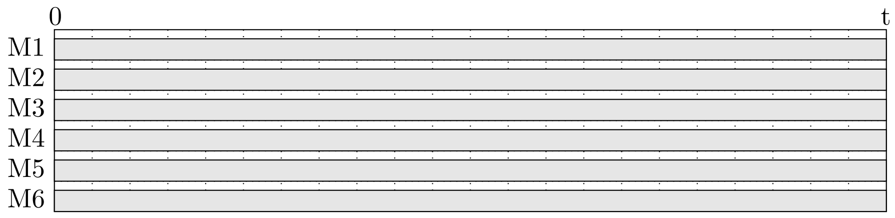


Figure 6.17: Generation of instances with known best solution job shop parallel machines: Step 1

2. For each operation, its processing time is randomly reduced by a setup time with the minimal setup duration between one and $\frac{1}{\text{number of operations}*2}$ t on that machine and the time the machine needs to wait until it is setup the first time. The sum of the waiting times on all machines until the setup times start can be minimized if operations are set up in ascending order of the setup times until all machines are running. The available capacity with due regard to the minimum waiting times still matches the sum of setup and processing times. Hence, the schedule is optimal. (Figure 6.18)

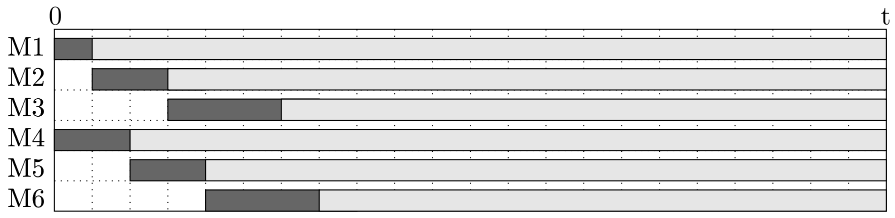


Figure 6.18: Generation of instances with known best solution job shop parallel machines: Step 2

3. The processing times are interrupted by the inserted setup times. Hence, new operations are created. The minimum duration of the setup needs to be greater than or equal to the m smallest setup time. Setup times can be inserted at any position on the machines between the end of the smallest setup and t, where the available capacity on the machine is given and minimum one of the setup operators has idle time. Furthermore, between each setup, there needs to be a minimum processing time of a single time unit.

   (a) Randomly choose a feasible starting position for a setup.

   (b) Randomly choose a finishing for the setup between the m minimum setup time and the maximum between $\frac{1}{\text{number of operations}}$ t and the interruption of the capacity due to setup operator unavailability or the starting of the next operation.

(c) Continue until the chosen operation limit has been reached, or no capacity is left.

The available capacity with due regard to the minimum waiting times still matches the sum of setup and processing times. Hence, the schedule is optimal. (Figure 6.19)

4. Each created operation now gets additional information assigned.

   (a) For every operation–machine combination, except for the combination which has already been created, randomly pull a number between 0 and 100. If the number is greater than 50, the operation can run on the machine.

   (b) Operations get also batched and sequenced to jobs.

      i. Select the earliest starting operation which has either no predecessor in the job or which starts after the production end of the last selected operation in the same job.

      ii. The selected operation is attached to the end of the currently created sequence of operations in the current job. The operation cannot be chosen for other jobs anymore.

      iii. If a job has 10 operations, or there are no more operations for the job to choose, the job is closed.

      iv. If there are so far not to a job assigned operations, create a new job and go back to i.

   (c) Due dates for the operations. Each operation gets a due date assigned on which the processing of the operation in the built schedule is finished. Thus, there is no tardiness in the optimal schedule. Since we cannot divide our best-found solution to the best-found solution of zero, we set the value of the optimal solution to one.

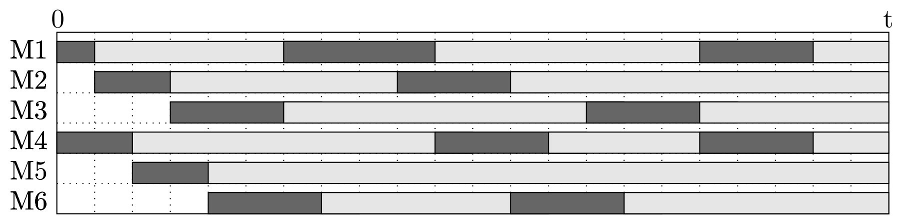


Figure 6.19: Generation of instances with known best solution job shop parallel machines: Step 3

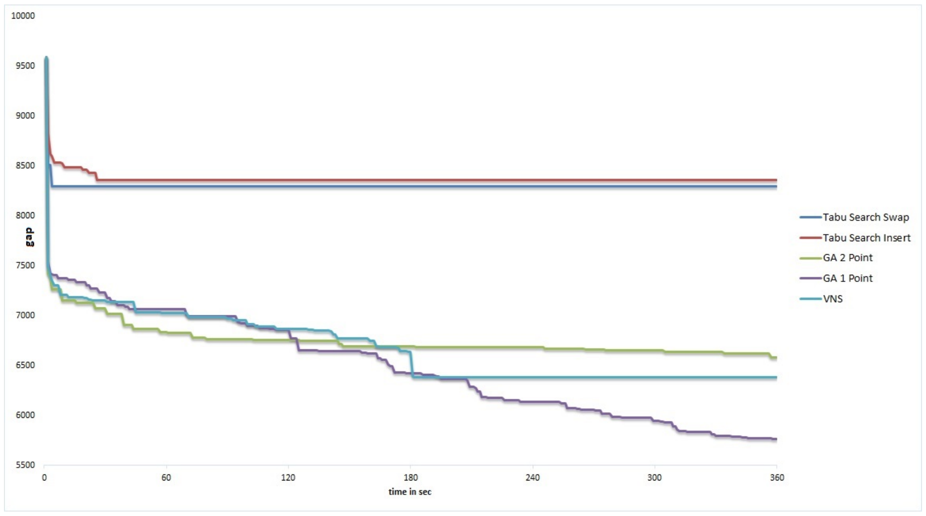


Figure 6.20: Diagram Case 143

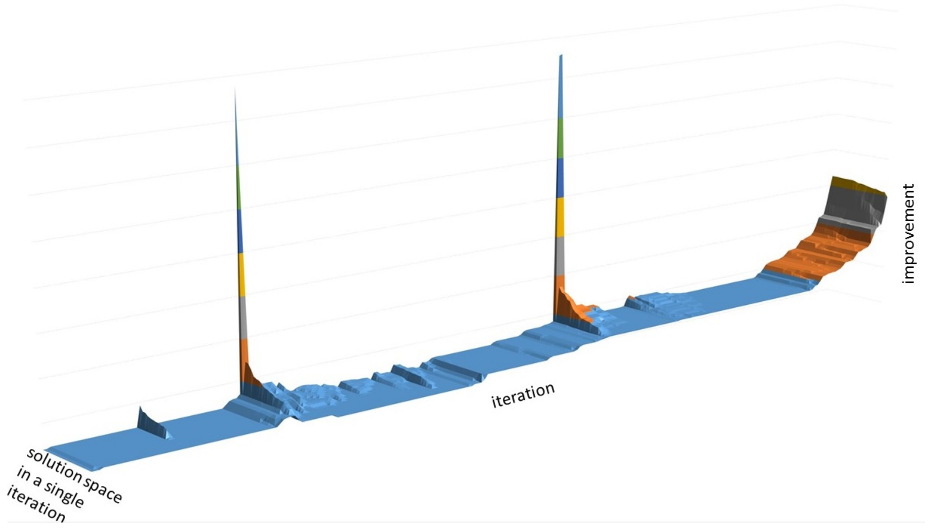


Figure 6.21: Neighbourhood TS Case 143

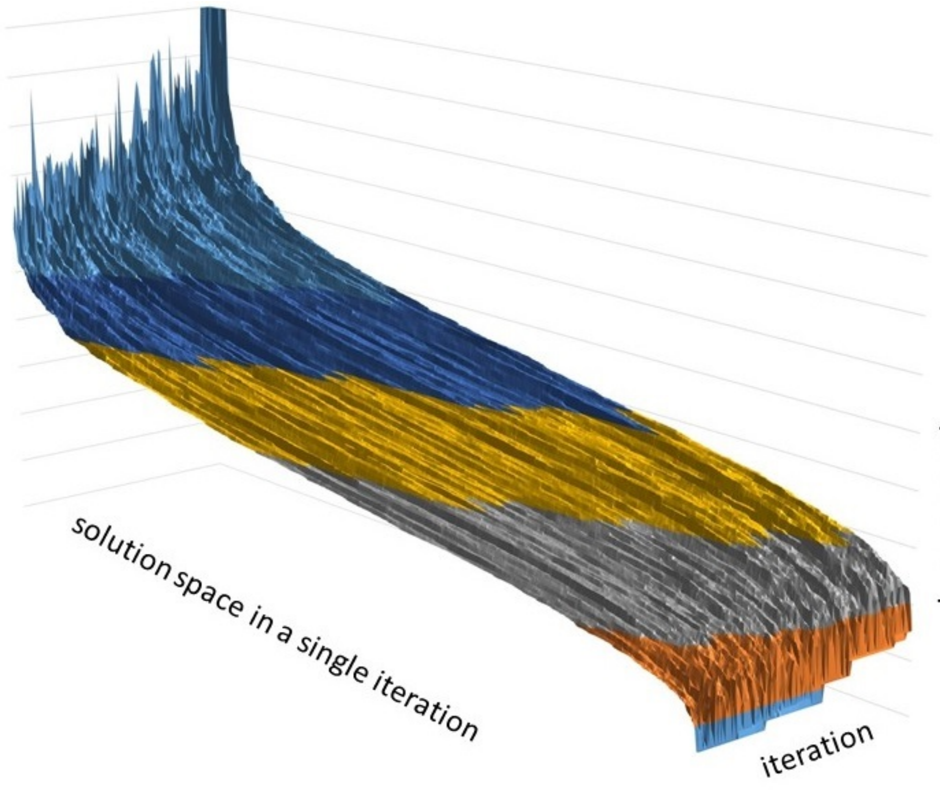


Figure 6.22: Neighbourhood GA Case 143

Case	Machines	Operations	Setup overlaps
139/151	10	100	1
140/152	10	100	3
141/153	10	100	5
142/154	10	200	1
143/155	10	200	3
144/156	10	200	5
145/157	25	250	1
146/158	25	250	3
147/159	25	250	5
148/160	25	500	1
149/161	25	500	3
150/162	25	500	5

Table 6.8: Known best solution 139–150 makespan / 151–162 total tardiness

Cases 139–150
The first thing to notice in Table 6.9 is that the starting solutions are as far as 264,35% (case 147) away from the optimum, while the metaheuristics can get as low as 35,08% to the optimum (VNS in case 144).
While the GAs performed weakly in the random instances makespan cases, they performed much better in cases when the optimal solution is known. In six of the twelve cases the best-found solution of a GA is better than the best-found solution of a TS. An explanation is that in the random cases, the solutions found are much closer to the optimal solution, although the lower bounds in the pre-tests suggested otherwise. Therefore, the GAs might have a disadvantage in the random cases as they are good in exploring a broader neighbourhood in a short time as this might be in cases 139–150. In ten of the twelve cases, the 1-point crossover is better than the 2-point. The 2-point crossover is only better than the 1-point in two cases. Besides, the 1-point crossover finds the best solution among all metaheuristics in five cases. The GAs show a strong improvement at the beginning in (Figure 6.20) (Diagram Case 143). They then continue to find better solutions over time but with less improvement per time unit. Nevertheless, it seems that when they would have more time, even better solutions could be found as they look like a linear function in the end. In the GA neighbourhood (Figure 6.22), it can be seen how weak solutions get eliminated over time and better solutions are continuously discovered. The figure suggests that with further running time, even better solutions are likely to be found.

The TS Swap is better than the TS Insert in eight cases. Hence, the performance is opposite to the performance in cases 91–102 (random instances, makespan). To the best of our knowledge, no explanation that a Swap exchange explores a neighbourhood more broadly than an Insert exchange exists. Both TSs have a strong improvement at the beginning in (Figure 6.20) (Diagram Case 143). They are, however, stuck at a certain level shortly thereafter. This could in-

	Starting solution	GA 2 point	GA 1 point	TS Swap	TS Insert	VNS
139	194,74%	97,87%	67,2%	194,74%	163,31%	91,35%
140	164,77%	78,95%	52,43%	71,77%	53,94%	54,73%
141	155,23%	97,03%	57,41%	139,90%	119,14%	75,80%
142	129,52%	76,5%	47,7%	87,31%	93,62%	65,44%
143	139,85%	64,58%	44,18%	107,38%	109,14%	59,62%
144	137,5%	68,33%	62,23%	120,78%	108,96%	35,08%
145	234,29%	182,36%	154,26%	167,15%	183,58%	145,29%
146	256,55%	172,96%	165,45%	188,47%	191,05%	163,83%
147	264,35%	196,38%	188,04%	165,33%	183,70%	177,60%
148	195,32%	131,29%	129,02%	123,82%	157,63%	107,06%
149	167,85%	121,28%	121,73%	102,9%	142,56%	95,01%
150	182,72%	137,30%	139,78%	121,02%	170,07%	108,59%

Table 6.9: Gap to the optimal solution for cases 139–150

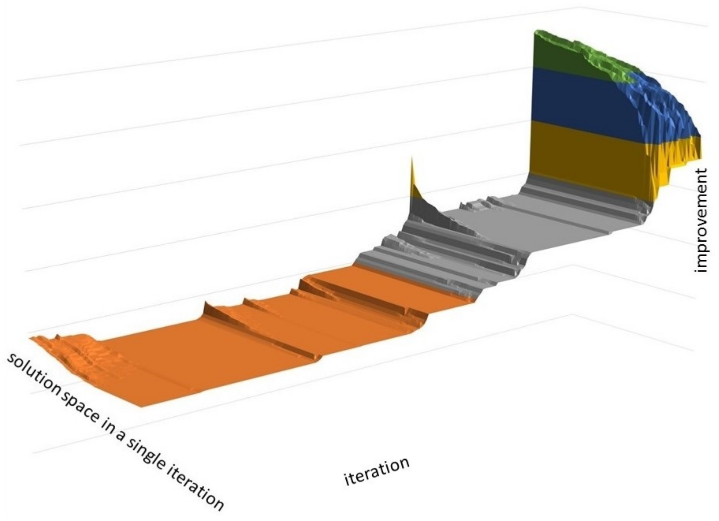


Figure 6.23: Neighbourhood VNS Case 143

dicate, that the optimal solution is in a completely different neighbourhood, which cannot be reached during the 360 seconds. Also the neighbourhood for the TS shows this kind of performance (Figure 6.21) (Neighbourhood TS Case 143). There is a strong improvement in the beginning, before the search browses through a plain area, with occasionally much weaker solutions.

The VNS shows the best performance of all metaheuristics. In six of the twelve cases, the VNS shows a better solution than any other method. In the other cases, the VNS is close to the best result. An explanation for this could be that, as all methods work very well, the GAs broadly identify the region in which the TSs can then search with more detail. On the contrary, in Figure 6.20 (Diagram Case 143) and Figure 6.23 (Neighbourhood VNS Case 143), the VNS follows the performance of the GAs until it is switched to the TSs. It can then gain only minor improvements. As this is an exception and not the norm in these cases, this is not investigated further.

Cases 151–162
In contrast to the random instances total tardiness cases in this chapter (Table 6.4), there are huge gains for any metaheuristic in the known best solution total tardiness cases (Table 6.9), therefore indicating that the solutions found in the random cases are close to the optimal solution. It has to be considered, though, that in cases 115–126 fewer iterations can be calculated during the 360-second running time. The reason for this remains unknown to date.

The GAs are sometimes able to reduce the gap to the lower bound by more than 60%. The 1-point crossover especially shows a strong performance. In all cases it is better than the 2-point crossover. In one case it is the best metaheuristic among all. After an initial improvement for both GAs in Figure 6.24 (Diagram Case 155), they continue to show improvement but at a slower rate. The graphs look then nearly linear for them. When the actual case, 6.26 (Neighbourhood GA Case 155), and case 119, Figure 6.7, (same problem setting random instances) are compared, there is clearly a different performance. Here, the GA runs on one hand through more iterations and on the other hand eliminates the weaker solutions in its population over time.

The TS Swap provides the best solution among the heuristics in four cases, while the TS Insert does so in five cases. In many cases, they can reduce the gap by more than 80%. The TSs have a strong improvement at the beginning in 6.24 (Diagram Case 155) and show almost no gains after 90 seconds. Nevertheless, they are already much closer to the optimum in contrast to the GAs. It can be clearly seen in Figure 6.25 (Neighbourhood TS Case 155), that the TSs runs through a more or less homogenous neighbourhood in each iteration. It should be also noted, that the TS runs through more iterations than in the same case (Case 119 Figure 6.6) with random instances. As stated before, the performance of a metaheuristic seems to be related to the neighbourhood structure. Homogenous neighbourhoods seem to provide better performance for the metaheuristics.

	Starting solution	GA 2 point	GA 1 point	TS Swap	TS Insert	VNS
151	347065	189337,67	123117,67	49254,33	51907	59115
152	324719,33	190752,33	127844	38746	45222,67	52017,67
153	295125,33	182238	126305,67	32958,33	30884,67	51815,67
154	518974	318028	234858	99231,33	99599,33	98448,33
155	449277	317448,33	222565	66025,67	71260,67	65864,67
156	435968	319124,33	228405	69542,67	66073,33	75388
157	1011233	767401	652216,67	188290	184463,67	255061,67
158	948611	773500,67	686941,33	142391	146483	207920,33
159	1111981	845270,33	700855	153162,33	148381,33	219719
160	1392076,33	1152400,67	1131627,33	852940,67	852940,67	846477
161	1331305,33	1165148,67	1128194	719269,67	718231,33	803533,33
162	1663244,67	1288367,67	1184460,67	742195	738784,67	886512

Table 6.10: Gap to the optimal solution in time units for cases 151–162

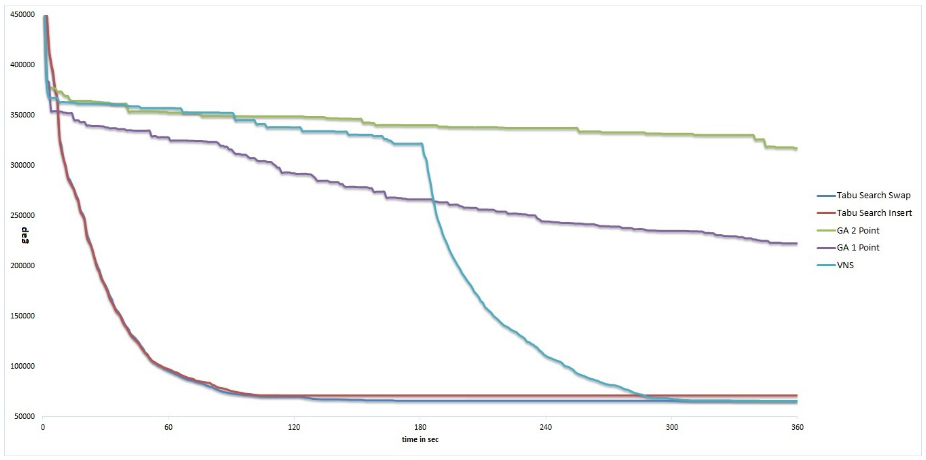


Figure 6.24: Diagram Case 155

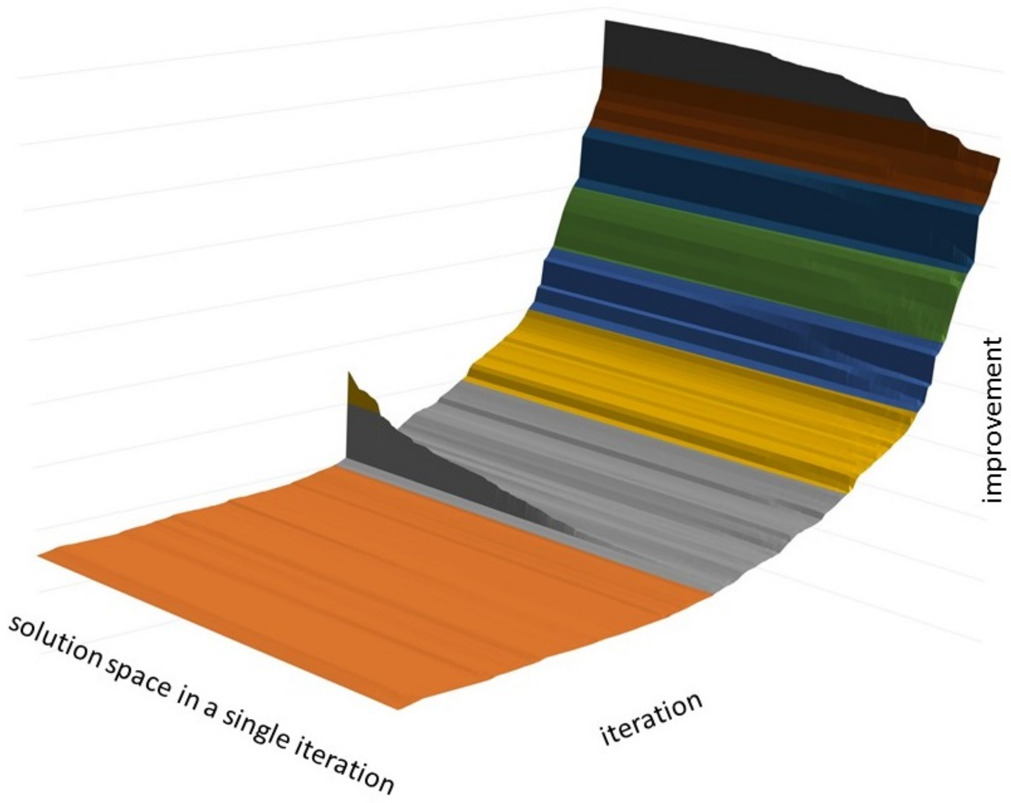


Figure 6.25: Neighbourhood TS Case 155

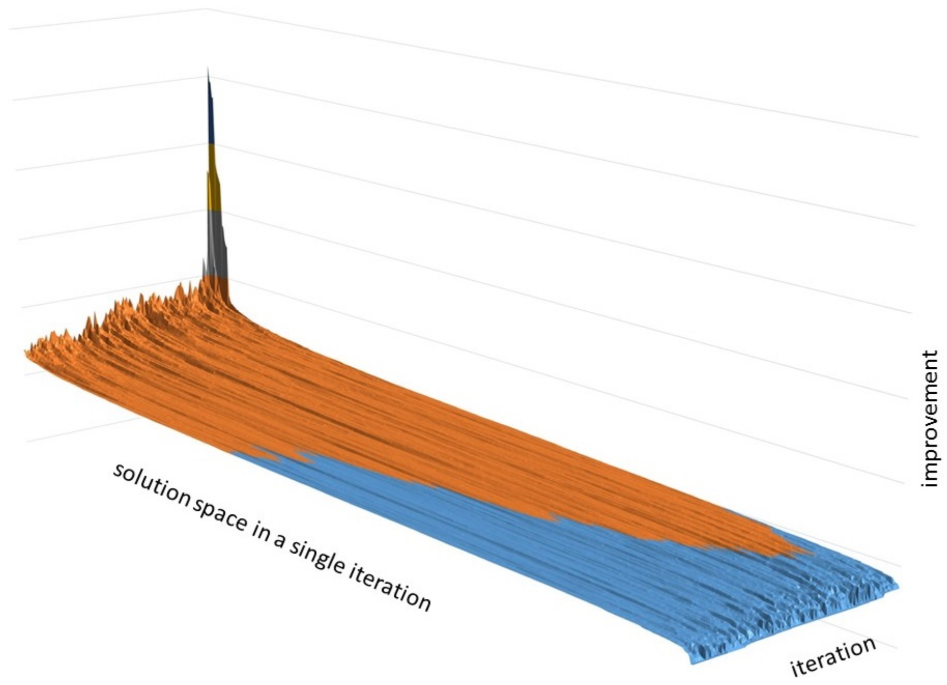


Figure 6.26: Neighbourhood GA Case 155

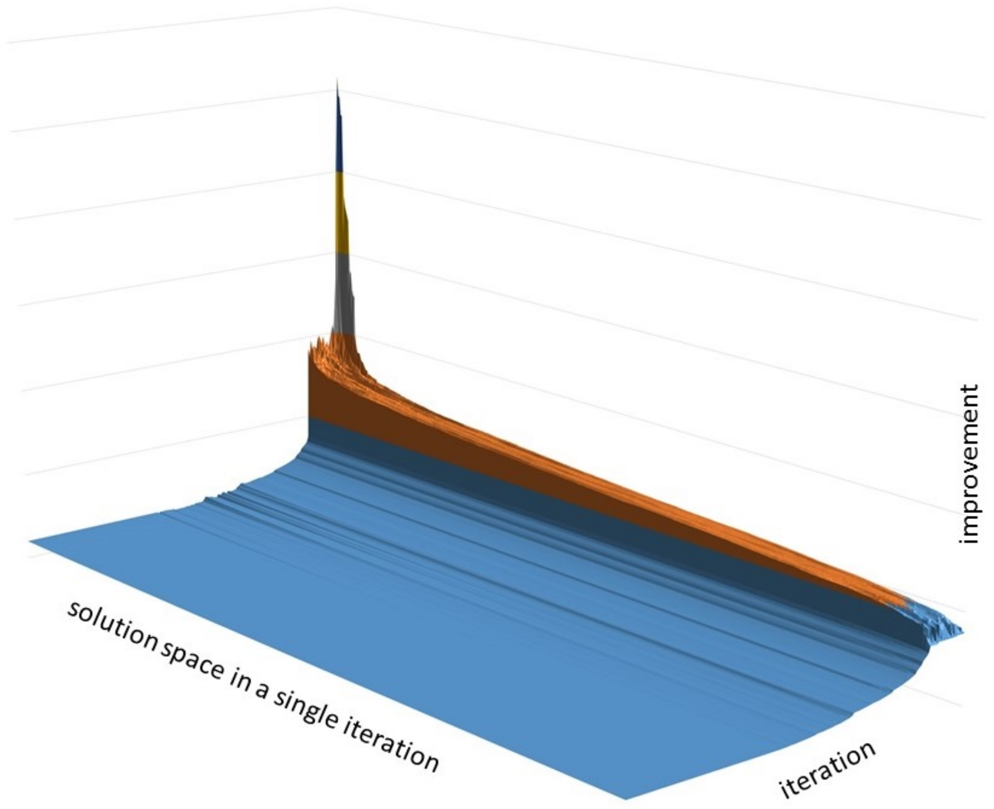


Figure 6.27: Neighbourhood VNS Case 155

The VNS also shows a strong performance. In three of the twelve cases, it finds the best solution among all metaheuristics. In the other cases, it is close by. The VNS, again, basically follows the form of the other methods in Figure 6.24 (Diagram Case 155). Until the switching point at 180 seconds, the performance looks linear. It is then switched to the TS Swap and immediately the graph shows the curve which the TS had at the beginning. The VNS again shows the typical structure often seen before (Figure 6.27). There are no specifically new and interesting peculiarities.

6.4.3 Case study

To describe the results for the company, we solved 20 production cases with real data from the company. We refer to them as case 163. A schedule is done only for the next two weeks, since there are major adaptions within a small time frame for customer orders. Therefore, a larger planning horizon would mean major variations and repairments in the planned schedule. During this period, 19 jobs with 190 operations need to be scheduled on average on the 13 machines.[14] In the smallest case, only 16 jobs with 160 operations need to be planned, while in the largest case 21 jobs with 210 operations need to be scheduled.
Although the end components for customers are similar, each component has distinct characteristics. Thus, the components can only be produced based on customer order. As a result, there are major variations in the production quantity and the production time for the operations. In particular, there are small quantities for reorders of faulty parts. In the data we received, the smallest production time is 0 minutes,[15] the longest production time is 4320 minutes, where the average median of all production times for the operations is at 690 minutes.
The smallest setup time is 5 minutes, the largest setup time is 720 minutes, and the average median of all setup times is 362 minutes. There are three setup operators available during the working hours.[16] The objective is to minimize total tardiness.

Since the goal in this sub chapter is a comparison of the planning procedures of the company, we use their due date heuristic to find a starting solution and compare the VNS against it.[17]

[14]The data suggested that each job has exactly 10 operations.

[15]This could be a handling operation. Although there is no handling time for the operation, is still needs to wait, as there might be a handling time for the operations in the queue before.

[16]Setup operators and machines work in three shifts, seven days a week in the following time intervals: 6:00 a.m.– 2:00 p.m., 2:00 p.m.–10:00 p.m., and 10:00 p.m.–6:00 a.m. Furthermore, each shifts includes a break after four working hours for half an hour.

[17]Although the VNS shows only the second-best performance in the case study tests, we compare the performance of the due date heuristic further against it, since it shows the most stable performance in all tests.

The first comparison is the total tardiness. In the due date heuristic operations are on average 16,132 days late while the tardiness for an operation is in the VNS only 10,02 days. Moreover, the total production output, i.e. the number of finished operations per day is higher for the VNS with 10,97 to 7,82 operations in the due date heuristic. The higher production output is due to the decrease of waiting times for the setup operators and for the machines. In the due date schedule setup, operators had a utilization of 64,29% and machines 38,46% of their time while in the VNS schedule the utilizations are 90,24% for setup operators and 61,90% for the machines. It can therefore be stated that the use of more advanced algorithms in production planning is suggested.
Although we did not test the other algorithms, we would expect the TSs to perform better than the GAs as they were better in the random instances as well as in the instances where the best solution is known with 200 jobs and three setup operators. Although the VNS is better than the TSs in the known best solution cases, it is still expected that the TSs provide better solutions in the case study. This is because they perform better in the random instances, which is closer to the case study regarding its structure.

6.5 Conclusion

This chapter was the final part of the investigation of setup operators in the production scheduling environment. We extended the general problem of "job shop scheduling with parallel machines" with additional constraints of setup operator availability. A formal problem description was given in 6.2. Apart from the consideration of setups which can only be undertaken if a setup operator is available, break times for the machines and setup operators were included.

Two GAs, two TSs, and a VNS were developed (6.3) to investigate the performance gains for using advanced algorithms against the commonly used sorting heuristics. The objective function was to minimize either makespan or total tardiness.

Randomly generated instances (6.4.1) with up to 25 machines, 50 jobs and 500 operations were tested with and without breaks in 6.4.1. If no breaks were considered, the VNS performed best. In five of the twelve cases, it found the best solution. The TSs found the best solution of all metaheuristics in three cases. The GAs showed a diverse performance. While they provided the best results in the smallest cases, they had major problems with the bigger cases.

The situation is likewise when breaks are considered.

But the results were different when the objective minimum total tardiness was considered. In these instances, either with breaks or without, the VNS performed well, the TSs were strong, while the GAs showed weak performances in all problem sizes.

If the known best solution cases were considered (6.4.2.), the results were again different. The VNS once again showed a robust performance. The GAs were very strong as well. The TSs, surprisingly, showed a much weaker performance. The GAs performed sufficiently in the minimize total tardiness cases but got clearly outperformed in all cases by the TSs. The VNS showed particularly good results in all instances, thus taking the best of each of the underlying methods. Finally, the VNS was tested in Subsection 6.4.3 on real data provided by the company. It was clearly shown that the use of a more advanced heuristic can lead to significant savings for companies.

Section	Type	Target	Density	Performance
Parallel dedicated machines	Random instances	Makespan no breaks	low	GAs best, TS good
		Tardiness no breaks	low	TSs better than rest
		Makespan breaks	low	bigger gap with breaks rest same
		Tardiness breaks	low	GAs have problems
	Known best solution	Makespan	high	all metaheuristics have strong performance

Table 6.11: Results 4

When the results of this final subsection are compared to the results of the previous subsection, it is clear that all metaheuristics perform well when the optimal schedule is denser. Also, in the known best solution cases, all metaheuristics perform very well. In the random instance cases, the metaheuristics perform better when there are more setup operators, i.e. when the schedule is dense. Furthermore, in line with the other chapters, the smaller the instances, the better the performance of the metaheuristics. This holds especially for the GAs. It is expected that they have problems with convergence in larger instances. Thus, it acts more like a random search. Therefore, it is concluded, that more running time is needed for the heuristics to come to an appropriate solution. About the influence of the target on the metaheuristic, no definitive answer can be given. In some makespan cases, the GAs are better than the TSs in some it is opposite. The same holds for the total tardiness problems.
Overall, the chapter provided additional information to the field of setup operator scheduling in production planning. When used, the algorithms described will probably decrease costs for companies which have a "job shop with parallel machines subject to setup constraints" problem. As described in Section 2.3, many SMEs do not have, however, the adequate IT infrastructure to use advanced algorithms. When a company has such a complex problem, it is clearly recommended to work on the IT infrastructure to be able to use these types of algorithms first. The increasement of setup operators is not recommended to

start with, as there are so many influencing variables on a plan, that the pure increasement might not solve the problems the companies face. This chapter is also the final part to the literature of our work in setup operator scheduling. While first insights into these type of scheduling problems have been provided, many new questions arose regarding the context-specific performance of the metaheuristics. Therefore, also in this chapter, a larger computational study with slightly different MIPs is recommended to identify the influence of the different constraints on the performance of the heuristics more clearly.

Section	Type	Target	Density	Performance
Parallel machines	Random instances	Makespan	low high	GAs not good, worse with larger instances, better with more setup operators/ TS better than GAs, better with more setup operators, worse with large instances
		Tardiness		GAs better than case with makespan, better with more setup operators, worse with large instances / TSs better than GAs, better in small instances, with more setup operators
	Known best solution	Makespan	high	good results for TSs and GAs , TSs better
		Tardiness	high	very good results for GAs , good results for TSs, better in small instances, better with more setup operators

Table 6.12: Results 5

Section	Type	Target	Density	Performance
Job shop with parallel machines	Random instances	Makespan no breaks		Bad performance for all / GAs better than TSs in small instances, better with higher number of setup operators, better in smaller
		Tardiness no breaks	low / high	TSs better than GAs, TSs has average performance, better in small instances, better with more setup operators
		Makespan breaks	low / high	comparable to no breaks
		Tardiness breaks	low / high	comparable to no breaks
	Known best solution	Makespan	high	GAs in smaller instances much better than TSs, in larger instances approximately equivalent
		Tardiness	high	Both strong performance, TSs clearly better than GAs

Table 6.13: Results 6

Chapter 7

Final Conclusion

The investigation of production planning in small and medium-sized companies showed that production plans were usually made with the help of sorting heuristics without considering the special problem structure of the companies production environment such as setup operator availability. The main aim of this work was to investigate the following:

1. If methods from the operations research context could be applied to more complex, real-life problems of medium-sized companies

2. Which methods worked well for these types of problems

3. Moreover, setup operators were explicitly considered in the models and solution algorithms.

To this end, an essential understanding of the fundamentals of production planning, Section 2.1, and of how small and medium-sized companies are different from large companies, Section 2.2, was needed. The theory of production planning was then combined with the differences between small and medium-sized companies and large companies in Section 2.3. Within this section, based on a quantitative survey we conducted ourselves and with the help of further studies, we investigated the planning situation of small and medium-sized companies. Our study and the further studies suggested an insufficient planning system inside SME. A central planning approach to scheduling which includes all relevant resource constraints can be concluded from the survey in Section 2.3. As the study has limited overall significance for the reasons described in 2.3., but is important to outline the overall planning situation in SME, we abstracted the problem of production planning for three typical real world cases, modeled them, showed the solvability and the limitations. Before we come to the cases, we went on to provide a state-of-the-art literature review (in Chapter 3) for these types of scheduling problems by focusing on the most problematic scarce resource—setup operator availability.
In the first case, we investigated "parallel dedicated machines subject to setup

constraints". We developed seven different metaheuristics to solve the problem. The results suggested that any company with this kind of problem should seriously consider using one of the more advanced algorithms to create a production plan. In the second case, i.e. the more general form of "parallel dedicated machines subject to setup contraints", "parallel machines subject to setup contraints" we developed the five heuristics with the most promising results from the first case and tested them on different instances. While some heuristics showed adequate results, others lacked in terms of certain problem parameter settings. Therefore, it was concluded that the problem context/setting had a major influence on the tool of choice. In the third context, we analysed the most general problem of our three cases, the "job shop with parallel machine scheduling subject to setup operator constraints". Again, the performance of some methods was strongly influenced by the problem context. While we could not find a general advantage of certain techniques, it was shown that these kind of problems are generally solvable in an acceptable time frame.

On the whole, the work is the first to provide insights into the production scheduling process of small and medium-sized companies with additional setup operator constraints. Practically, the results are of special importance to small and medium-sized companies, which could increase their production output by using the described algorithms in the scheduling department. To use the algorithms, adequate IT infrastructure is essential. As many SMEs lack sufficient IT infrastructure (2.3), they should check and update their basic infrastructure. Apart from IT, when companies are faced with parallel (dedicated) machine problems, increased awareness that setup operators are a limiting factor and the explicit consideration of those in the manual planning might help. If they are considered a limiting factor in the plans, which they likely are, the overall increase in the number of setup operators will help. Employing more setup operators explicitly may not be needed, but it should be checked whether through advanced training machine operators are able to do both tasks. This work also contributes to the sparse literature of setup operator scheduling. Nevertheless, while old questions have been answered, new questions arose. What remains unclear is the exact influence of the problem context on the ability of the method to find an appropriate solution in an acceptable time frame. Therefore, future work should try to identify the underlying reasons for the performance/non-performance of the algorithms. A cluster analysis could be useful in this respect. In addition, future studies should also take more factors such as machine breakdowns into account. Finally, the algorithms should be implemented in the companies, and it should be checked whether real savings can be achieved.

Bibliography

A. H. Abdekhodaee and A. Wirth. Scheduling parallel machines with a single server: Some solvable cases and heuristics. *Computers & Operations Research*, 18(3):295 – 315, 2002.

A. H. Abdekhodaee, A. Wirth, and H. S. Gan. Equal processing and equal setup time cases of scheduling parallel machines with a single server. *Computers & Operations Research*, 31(11):1867 – 1889, 2004.

A. H. Abdekhodaee, A. Wirth, and H. S. Gan. Scheduling two parallel machines with a single server: the general case. *Computers & Operations Research*, 33(4):994 – 1009, 2006.

M. Abdel-Basset, M. Gunasekaran, D. El-Shahat, and S. Mirjalili. A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem. *Future Generation Computer Systems*, 2018.

H. Abedinnia, C. H. Glock, E. H. Grosse, and M. D.Schneider. Machine scheduling in production: A content analysis. *Applied Mathematical Modelling*, 50:279–299, 2017.

H. Abedinnia, C. H. Glock, E. H. Grosse, and M. D.Schneider. Machine scheduling problems in production: A tertiary study. *Computers & Industrial Engineering*, 111:403–416, 2017.

J. O. Achugbue and F. Y. Chin. Scheduling the open shop to minimize mean flow time. *SIAM J. Comput.*, 11:709–720, 1982.

K. F. Ackermann and H. Blumenstock. Personalmanagement in mittelständischen unternehmen neubewertungen und weiterentwicklungsmöglichkeiten. In K. F. Ackermann and H. Blumenstock, editors, *Personalmanagement in mittelständischen Unternehmen*, pages 5 – 69. Schäffer-Poeschel Verlag, Stuttgart, 1993.

Z. Acs and D. Audretsch. Innovation in large and small firms. *Economics Letters*, 23:109 – 112, 1987.

A. Agnetis, M. Nicosia, and A. Pacifici. A job shop problem with one additional resource type. *Journal of Scheduling*, 14(3):225 – 237, 2011.

T. Aldowaisan. A new heuristic and dominance relations for no-wait flowshops with setups. *Computers & Operations Research*, 28(6):568 – 584, 2001.

T. Arbaoui and F. Yalaoui. Solving the unrelated parallel machine scheduling problem with additional resources using constraint programming. In *Asian Conference on Intelligent Information and Database Systems*, pages 716–725. Springer, 2018.

C. Bähr and M. Steier. Industrieregion Südwestfalen: Spitze in Deutschland - Führend in Nordrhein-Westfalen. Technical report, IW Consult GmbH, 2013. http://www.kreis-olpe.de/media/custom/2041_2100_1.PDF?1370507902.

D. Bai, Z. Zhang, Q. Zhang, and M. Tang. Open shop scheduling problem to minimize total weighted completion time. *Engineering Optimization*, 49(1):98–112, 2017.

J. Baker. Adaptive selection Methods for genetic Algorithms. In J. F. Greffenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms and Their Application*, page 101 111. L. Erlbaum Associates Inc, Hillsdale, 1st edition, 1985.

M. F. Bako and R. G. Vickson. One-operator-two-machine open shop and flow shop problems with setup times for machines and weighted number of tardy jobs objective. *Optimization Methods and Software*, 19(2):165 – 178, 2004.

P. V. Balakrsihnan and V. S. Jacob. Genetic algorithms for product design. *Management Science*, 42(8):1105 1117, 1996.

J. A. C. Baum, T. Calabrese, and B. S. Silverman. Dont go it alone: Alliance network composition and startups performance in canadian biotechnology. *Strategic Management Journal*, 21:267 – 294, 2000.

G. Berisha and J. S. Pula. Defining small and medium enterprises: a critical review. *Academic Journal of Business, Administration, Law and Social Sciences*, 1(1):17 – 28, 2015.

J. Bessant. The rise and fall of supernet: A case study of technology transfer policy for smaller firms. *Research Policy*, 28:601 – 614, 1999.

G. E. Blau, J. F. Pekny, V. A. Varma, and P. R. Bunch. Managing a portfolio of interdependent new product candidates in the pharmaceutical industry. *Pharmaceutical Industry. Journal of Product Innovation Management*, 21(4):227 245, 2004.

J. Blazewicz, P. Dell'Olmo, and M. Drozdowski. Scheduling of client server applications. *International Transactions in Operational Research*, 6(4):345 – 363, 1999.

K. Bley, C. Leyh, and T. Schäffer. Digitization of german enterprises in the production sector do they know how " digitized " they are? 08 2016.

C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268 – 308, 2003.

J. E. Bolton. *Small Firms: Report of the Committee of Inquiry on Small Firms Presented to the Parliament by the Secretary of State for Trade and Industry by Command of Her Majesty, Nov. 1971.* HM Stationery Office, London, 1971.

S. Boß and C. Deckert. Einsatz von pps-lsungen in kmus. *Zeitschrift für wirtschaftlichen Fabrikbetrieb*, 112(11):811 – 815, 2017.

K. E. Bourland and L. K. Carl. Parallel-maschine scheduling with fractional operator requirements. *IIE Transactions*, 26(5):45 – 65, 1994.

P. Brucker and S. Knust. Complexity results for single-machine problems with positive finish-start time-lags. *Computing*, 63(4):299 –316, 1999.

P. Brucker, A. Gladky, H. Hoogeveen, M. Y. Kovalyov, C. N. Potts, T. Tautenhahn, and S.L. van de Velde. Scheduling a batching machine. *J. Sched.*, 1(1):31–54, 1998.

P. Brucker, A. Drexl, R. Möhring, K. Neumann, and E. Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112(1):3 – 41, 1999.

P. Brucker, C. Dhaenens-Flipo, and S. Knust. Complexity results for parallel machine problems with a single server. *Journal of Scheduling*, 5(6):429 – 457, 2002.

P. Brucker, S. Knust, and G. Wang. Complexity results for flow-shop problems with a single server. *European Journal of Operational Research*, 165(2):398 – 407, 2005.

P. Brucker, S. Knust, and C. Oguz. Scheduling chains with identical and constant delays on a single machine. *Mathematical Methods of Operations Research*, 63(1):63 – 75, 2006.

J. Bruno, E. G. Coffman Jr., and R. Sethi. Scheduling independent tasks to reduce mean finishing time. *Comm. ACM*, 17:382–387, 1974.

S. Brunswicker and W. Vanhaverbeke. Open innovation in small and medium-sized enterprises (smes): External knowledge sourcing strategies and internal organizational facilitators. *Journal of Small Business Management*, 53(4):1241 – 1263, 2015.

T. Cayirli and E. Veral. Outpatient scheduling in health care: A review of literature. *Production and Operations Management*, 12(4):519–549, 2003.

F. Ceci and D. Iubatti. Personal relationships and innovation diffusion in sme networks: A content analysis approach. *Research Policy*, 41(3):565 – 579, 2012.

D. Chen, P. B. Luh, L. S. Thakur, and J. Moreno Jr. Optimization-based manufacturing scheduling with multiple resources, setup requirements, and transfer lots. *IIE Transactions*, 35(10):973 – 985, 2003.

L. Chen, K. Jansen, and G. Zhang. On the optimality of approximation schemes for the classical scheduling problem. In *25th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 657—668, 2014.

C.-L. Chen. Iterated population-based vnd algorithms for single-machine scheduling with sequence-dependent setup times. *Soft Computing*, pages 1–15, 2018.

T. C. E. Cheng and C. Sriskandarajah. One-operator two-machine flowshop scheduling with setup and dismounting times. *Computers & Operations Research*, 26:715 – 730, 1999.

H. W. Chesbrough, W. Vanhaverbeke, and J. West. *Open Innovation: Researching a New Paradigm.* Oxford University Press, Oxford, 2006.

R. Clements, B. Güenterberg, and H. E. Hauser. Unternehmensgröen-statistik 1997/98 daten und fakten. Technical report, Bundesministerium für Wirtschaft und Technologie, 1997.

H. Corsten and R. Gössinger. *Produktionswirtschaft: Einführung in das industrielle Produktionsmanagement.* Oldenbourg, München, 12th edition, 2009.

J. Curran and R. A. Blackburn. *Researching the Small Enterprise.* Sage Publications, London, 2001.

T.-K. Dao, T.-S. Pan, and J.S. Pan. Parallel bat algorithm for optimizing makespan in job shop scheduling problems. *Journal of Intelligent Manufacturing*, 29(2):451–462, 2018.

J. Vallikavungal Devassia, M. A. Salazar-Aguilar, and V. Boyer. Flexible job-shop scheduling problem with resource recovery constraints. *International Journal of Production Research*, pages 1–18, 2018.

M. Dios, V. Fernandez-Viagas, and J. M. Framinan. Efficient heuristics for the hybrid flow shop scheduling problem with missing operations. *Computers & Industrial Engineering*, 115:88–99, 2018.

J. Du and J. Y.-T. Leung. Minimizing total tardiness on one machine is np-hard. *Mathematics of Operations Research*, 15(3):483 – 495, 1990.

EBSCO. Search Results EBSCO Host. Website, 2017. http://search.ebscohost.com/login.aspx?profile=ehost&defaultdb=bth.

T. Edwards, R. Delbridge, and M. Munday. Understanding innovation in small and medium-sized enterprises: A process manifest. *Technovation*, 25:1119 – 1127, 2006.

M. Eid, M. Gollwitzer, and M. Schmitt. *Statistik und Forschungsmethoden*. Beltz, Stuttgart, 4th edition, 2015.

L. Epstein, A. Levin, A. J. Soper, and V. A. Strusevich. Power of preemption for minimizing total completion time on uniform parallel machines. *SIAM Journal on Discrete Mathematics*, 31(1):101–123, 2017.

A. Eremeev, M. Kovalyov, and P. Kuznetsov. Single product lot-sizing on unrelated parallel machines with non-decreasing processing times. In *Journal of Physics: Conference Series*, volume 944, page 012032. IOP Publishing, 2018.

W. Espelage and E. Wanke. Movement optimization in flow shop processing with buffers. *Mathematical Methods of Operations Research*, 51(3):495 – 513, 2000.

W. Espelage and E. Wanke. Movement minimization for unit distances in conveyor flow shop processing. *Mathematical Methods of Operations Research*, 57(2):173 – 206, 2003.

European-Commision. Benutzerleitfaden zur Definition von KMU. Report, 2015.

W. Eversheim. *Organisation in der Produktionstechnik: Grundlagen*, volume 1. VDI, Düsseldorf, 3rd edition, 1995.

V. Fernandez-Viagas, P. Perez-Gonzalez, and J. M. Framinan. The distributed permutation flow shop to minimise the total flowtime. *Computers & Industrial Engineering*, 2018.

FIR-RWTH-Aachen. Produktion am standort deutschland : Ergebnisse der untersuchung 2013. Report, 2013.

T. Frye, D. Hackenberg, V. Helmer, L. Kottenhahn, and R. Schmidt. Weltmarktführer und Bestleistungen der Industrie aus Südwestfalen. Technical Report 4, Industrie- und Handelskammer Arnsberg, Hellweg-Sauerland, Südwestfälische Industrie- und Handelskammer zu Hagen, Industrie- und Handelskammer Siegen, 2015. http://www.siegen-wittgenstein.de/media/custom/2170_413_1.PDF?1445263840.

H. Gan, A. Wirth, and A. Abdekhodaee. A branch-and-price algorithm for the general case of scheduling parallel machines with a single server. *Computers & Operations Research*, 39(9):2242 – 2247, 2012.

M. R. Garey and D. S. Johnson. *Computer and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, 1975.

M. R. Garey and D. S. Johnson. Two-processor scheduling with start-times and deadlines. *SIAM Journal on Computing*, 6(3):416–426, 1977.

M. R. Garey and D. S. Johnson. Strong NP-completeness results: motivation, examples, and implications. *J. Assoc. Comput. Mach.*, 25(3):499–508, 1978.

M. R. Garey, D. S. Johnson, and R. Sethi. The complexity of flowshop and jobshop scheduling. *Math. Oper. Res.*, 1(2):117–129, 1976.

C. A. Glass, Y. M. Shafransky, and V. A. Strusevich. Scheduling for parallel dedicated machines with a single server. *Naval Research Logistics*, 47(4):304 – 328, 2000.

F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13(5):533 549, 1986.

F. Glover. Tabu search - part 1. *ORSA Journal on Computing*, 1(2):190 206, 1989.

F. Glover. Tabu search - part 2. *ORSA Journal on Computing*, 2(1):4 32, 1990.

D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Wiley, New York, 1st edition, 1989.

T. Gonzalez and S. Sahni. Open shop scheduling to minimize finish time. *J. Assoc. Comput. Mach.*, 23(4):665–679, 1976.

R. L. Graham, , E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete Mathematics*, 5:287 – 326, 1979.

K. Haake. *Strategisches Verhalten in europischen Klein- und Mittelunternehmen*. Duncker & Humblot, Berlin, 1987.

K. Haase. *Lotsizing and Scheduling for Production Planning*. Springer, Berlin, 1994.

R. Hackstein. *Produktionsplanung und -steuerung - Ein Handbuch für die Betriebspraxis*. VDI, Dusseldorf, 2nd edition, 1988.

N. G. Hall and C. N. Sriskandarajah. Parallel machine scheduling with a common server. *Discrete Applied Mathematics*, 102(3):223 – 243, 2000.

K. Hasani, S. A. Kravchenko, and F. Werner. Block models for scheduling jobs on two parallel machines with a single server. *Computers & Operations Research*, 41:94 – 97, 2014.

K. Hasani, S. A. Kravchenko, and F. Werner. Minimizing interference for scheduling two parallel machines with a single server. *International Journal of Production Research*, 54(24):7148 – 7158, 2014.

K. Hasani, S. A. Kravchenko, and F. Werner. Minimizing total weighted completion time approximately for the parallel machine problem with a single server. *Information Processing Letters*, 114(9):500 – 503, 2014.

K. Hasani, S. A. Kravchenko, and F. Werner. Simulated annealing and genetic algorithms for the two-machine scheduling problem with a single server. *International Journal of Production Research*, 52(13):3778 – 3792, 2014.

K. Hasani, S. A. Kravchenko, and F. Werner. Minimizing the makespan for the two-machine scheduling problem with a single server: Two algorithms for very large instances. *Engineering Optimization*, 48(1):173 – 183, 2016.

M. Heck and H. Vettiger. Production planning and scheduling and sme. *International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering*, 8(7):2276 – 2288, 2014.

S. Helber and F. Sahling. A fix-and-optimize approach for the multi-level capacitated lot sizing problem. *International Journal of Production Economics*, 123(2):247 – 256, 2010.

F. Herrmann and M. Manitz. Ein hierarchisches Planungskonzept zur operativen Produktionsplanung und -steuerung. In T. Claus, F. Herrmann, and M. Manitz, editors, *Produktionsplanung und -steuerung: Forschungsansätze, Methoden und deren Anwendungen*, pages 7 – 22. Springer, Berlin, 2015.

M. Hiepler, D. Schnitzler, M. Schweitzer, L. Neul, and K. Schmidt. Digitalisierung in der Industrie: Erkenntnisse aus dem Mittelstand. Technical report, Siegener Mittelstandsinstitut, 2016.

J. H. Holland. *Adaption in natural and artificial systems.* University of Michigan Press, Ann Arbor, 1975.

J. H. Holland. *Adaption in natural and artificial systems.* MIT Press, Cambridge, 1992.

H. Hoogeveen, P. Schuurman, and G. J. Woeginger. Non-approximability results for scheduling problems with minsum criteria. *INFORMS Journal on Computing*, 13(2):157–168, 2001.

W. A. Horn. Minimizing average flow time with parallel machines. *Oper. Res.*, 21:846–847, 1973.

S. Huang, L. Cai, and X. Zhang. Parallel dedicated machine scheduling problem with sequence-dependent setups and a single server. *Computers & Industrial Engineering*, 58(1):165 – 174, 2010.

IfM. SMU Definition of the IfM Bonn. Website, 2002. http://ifm-bonn.org/mittelstandsdefinition/definition-kmu-des-ifm-bonn/.

K. Jansen, K. M. Klein, and J. Verschae. Closing the gap for makespan scheduling via sparsification techniques. In *43rd International Colloquium on Automata, Languages, and Programming, (ICALP)*, volume 72, pages 1–13, 2016.

K. Jansen. An eptas for scheduling jobs on uniform processors: using an milp relaxation with a constant number of integral variables. *SIAM Journal on Discrete Mathematics*, 24:457–485, 2010.

J. Jiao, Y. Zhang, and Y. Wang. A heuristic genetic algorithm for product portfolio planning. *Computers and Operations Research*, 34(6):1777 1799, 2007.

S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included. *Naval Res. Logist. Quart.*, 1:61–68, 1954.

K. A. De Jong. *An Analysis of the behavior of a class of genetic adaptive systems.* Dissertation, University of Michigan, 1975.

R. M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972)*, pages 85–103. Plenum, New York, 1972.

H. Kellerer and V. A. Strusevich. Scheduling parallel dedicated machines under a single non-shared resource. *European Journal of Operations Research*, 174:345 – 364, 2003.

H. Kellerer and V. A. Strusevich. Scheduling problems for parallel dedicated machines under multiple resource constraints. *Discrete Applied Mathematics*, 133:45 – 68, 2004.

L. Kerkhove and M. Vanhoucke. Scheduling of unrelated parallel machines with limited server availability on multiple production locations; a case study in knitted fabrics. *International Journal of Production Research*, 52(9):2630 – 2653, 2014.

M. Kim and Y. Hoon Lee. Mip models and hybrid algorithm for minimizing the makespan of parallel machines scheduling problem with a single server. *Computers & Operations Research*, 39(11):2457 – 2468, 2012.

S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):4 32, 1983.

D. Knop and M. Koutecky. Scheduling meets n-fold integer programming. In *Proceedings of the 13th Workshop on Models and Algorithms for Planning and Scheduling Problems (MAPSP)*, 2017.

C. P. Koulamas and M. L. Smith. Look-ahead scheduling for minimizing machine interference. *International Journal of Production Research*, 26(9):1523 – 1533, 1988.

C. P. Koulomas. Scheduling two parallel semiautomatic machines to minimize machine interference. *Computers & Operations Research*, 23(10):945 – 956, 1996.

S. A. Kravchenko and F. Werner. Parallel machine scheduling problems with a single server. *Mathematical and Computer Modelling*, 26(12):1 – 11, 1997.

S. Kravchenko and F. Werner. A heuristic algorithm for minimizing mean flow time with unit setups. *Information Processing Letters*, 79(6):291 – 296, 2001.

S. Kravchenko. Minimizing the number of late jobs for the two-machine unit-time job-shop scheduling problem. *Discrete Appl. Math.*, 98(3):209–217, 1999.

W. Kubiak and V. G. Timkovsky. A polynomial-time algorithm for total completion time minimization in two-machine job-shop with unit-time operations. *European J. Oper. Res.*, 94:310–320, 1996.

K. Kurbel. *Produktionsplanung und -steuerung im Enterprise Resource Planning und Supply Chain Management.* Oldenbourg, München, 7th edition, 2011.

E. L. Lawler and J. M. Moore. A functional equation and its application to resource allocation and sequencing problems. *Management Sci.*, 16(1):77–84, 1969.

E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Minimizing maximum lateness in a two-machine open shop. *Math. Oper. Res.*, 6(1):153–158, 1981.

E. L. Lawler. Optimal sequencing of a single machine subject to precedence constraints. *Management Sci.*, 19:544–546, 1973.

E. L. Lawler. A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness. *Ann. of Discrete Math.*, 1:331–342, 1977.

S. Lee, G. Park, B. Yoon, and J. Park. Open innovation in smesan intermediated network model. *Research Policy*, 39(2):290 – 300, 2010.

M. P. Leite and A. Ferreira. Smes and e-business: Implementation, strategies and policy. In M. M. Cruz-Cunha and J. Varajao, editors, *E-Business Managerial Aspects, Solutions and Case Studies*, pages 1 – 22. IGI Global, Hershey, 2011.

J. K. Lenstra and A. H. G. Rinnooy Kan. Computational complexity of discrete optimization problems. *Ann. Discrete Math.*, 4:121–140, 1979.

J. K. Lenstra, A. H. G. Rinnooy K., and P. Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343 – 362, 1977.

J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical programming*, 46(1-3):259–271, 1990.

S.-W. Lin and K.-C. Ying. Uniform parallel-machine scheduling for minimizing total resource consumption with a bounded makespan. *IEEE Access*, 5:15791–15799, 2017.

C. Y. Liu and R. L. Bulfin. Scheduling open shops with unit execution times to minimize functions of due dates. *Oper. Res.*, 36(4):553–559, 1988.

H. Lödding. *Verfahren der Fertigungssteuerung: Grundlagen, Beschreibung, Konfiguration*. Springer, Berlin, 3rd edition, 2015.

S. Lu, X. Liu, J. Pei, M. T. Thai, and P. M. Pardalos. A hybrid abc-ts algorithm for the unrelated parallel-batching machines scheduling problem with deteriorating jobs and maintenance activity. *Applied Soft Computing*, 66:168–182, 2018.

H. Luczak and W. Eversheim. Einleitung. In H. Luczak and W. Eversheim, editors, *Produktionsplanung und - steuerung: Grundlagen, Gestaltung und Konzepte*, pages 3 – 5. Springer, Berlin, Heidelberg, 2001.

P. Mank. *Personalpolitik in mittelständischen Unternehmen: Eigenarten, Versäumnisse, Chancen*. Frankfurter Allgemeine Zeitung, Frankfurt, 1991.

W. L. Maxwell. On sequencing n jobs on one machine to minimize the number of late jobs. *Management Sci.*, 16:295–29, 1970.

M. Mnich and A. Wiese. Scheduling and fixed-parameter tractability. *Mathematical Programming*, 154(1-2):533–562, 2015.

J. M. Moore. An n job, one machine sequencing algorithm for minimizing the number of late jobs. *Management Sci.*, 15:102–109, 1968.

S. M. Mousavi, I. Mahdavi, J. Rezaeian, and M. Zandieh. An efficient bi-objective algorithm to solve re-entrant hybrid flow shop scheduling with learning effect and setup times. *Operational Research*, 18(1):123–158, 2018.

M. Mrad and N. Souayah. An arc-flow model for the makespan minimization problem on identical parallel machines. *IEEE Access*, 6:5300–5307, 2018.

D. Much and H. Nicolai. *PPS-Lexikon*. Cornelsen, Berlin, 1995.

J. Mugler. *Grundlagen der BWL der KLein-und Mittelbetriebe*. Facultas, Wien, 2006.

A. Munier and F. Sourd. Scheduling chains on a single machine with non-negative time lags. *Mathematical Methods of Operations Research*, 57(1):111 – 123, 2003.

V. Nesello, A. Subramanian, M. Battarra, and G. Laporte. Exact solution of the single-machine scheduling problem with periodic maintenances and sequence-dependent setup times. *European Journal of Operational Research*, 266(2):498–507, 2018.

H. Nicolai, M. Schotten, and D. Much. Aufgaben. In H. Luczak and W. Eversheim, editors, *Produktionsplanung und -steuerung: Grundlagen, Gestaltung und Konzepte*, pages 29 – 74. Springer, Berlin, 1999.

M. Nouiri, A. Bekrar, A. Jemai, S. Niar, and A. C. Ammari. An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. *Journal of Intelligent Manufacturing*, 29(3):603–615, 2018.

J. Pacheco, S. Porras, S. Casado, and B. Baruque. Variable neighborhood search with memory for a single-machine scheduling problem with periodic maintenance and sequence-dependent set-up times. *Knowledge-Based Systems*, 145:236–249, 2018.

H.-C. Pfohl. Abgrenzung der Klein-und Mittelbetriebe von Großbetrieben. In H.-C. Pfohl, editor, *Betriebswirtschaftslehre der Mittel- und Kleinbetriebe*, pages 2 – 21. Erich Schmidt, Berlin, 2006.

H. Piroozfard, K. Y. Wong, and W. P. Wong. Minimizing total carbon footprint and total late work criterion in flexible job shop scheduling by using an improved multi-objective genetic algorithm. *Resources, Conservation and Recycling*, 128:267–283, 2018.

M. Porter. *Competitive strategy: techniques for analyzing industries and competitors*. Free Press, New York, 1980.

K. Rameshkumar and C. Rajendran. A novel discrete pso algorithm for solving job shop scheduling problem to minimize makespan. In *IOP Conference Series: Materials Science and Engineering*, volume 310, page 012143. IOP Publishing, 2018.

J. Schauer and C. Schwarz. Job-shop scheduling in a body shop. *Jounal of Scheduling*, 16(2):215 – 229, 2013.

A.-W. Scheer. Koordinierte planungsinseln: Ein neuer lösungsansatz für die produktionsplanung. *IWi-Hefte*, 86, 1991.

G. Schuh and A. Gierth. Aachener PPS-Modell. In G. Schuh., editor, *Produktionsplanung und -steuerung: Grundlagen, Gestaltung und Konzepte*, pages 11 – 27. Springer, Berlin, 2006.

G. Schuh and R. Roesgen. Aufgaben. In G. Schuh., editor, *Produktionsplanung und -steuerung: Grundlagen, Gestaltung und Konzepte*, pages 28 – 80. Springer, Berlin, 2006.

G. Schuh and C. Schmidt. Grundlagen des produktionsmanagements. In G. Schuh. and C. Schmidt, editors, *Produktionsmanagement*, pages 1 – 80. Springer, Berlin, 2014.

G. Schuh, U. Brandenburg, and S. Cuber. Aufgaben. In G. Schuh. and V. Stich, editors, *Produktionsplanung und -steuerung 1: Grundlagen der PPS*, pages 29 – 80. Springer, Berlin, 2011.

G. Schuh, C. Schmidt, and J. Helmig. Prozesse. In G. Schuh. and V. Stich, editors, *Produktionsplanung und -steuerung 1: Grundlagen der PPS*, pages 109 – 192. Springer, Berlin, 2011.

G. Schuh. *Produktkomplexität managen.* Carl Hanser, München, Wien, 2nd edition, 2005.

S. Schwerdfeger and R. Walter. Improved algorithms to minimize workload balancing criteria on identical parallel machines. *Computers & Operations Research*, 93:123–134, 2018.

O. Shahvari and R. Logendran. A comparison of two stage-based hybrid algorithms for a batch scheduling problem in hybrid flow shop with learning effect. *International Journal of Production Economics*, 195:227–248, 2018.

L. Sheremetov, J. Martínez-Muñoz, and M. Chi-Chim. Two-stage genetic algorithm for parallel machines scheduling problem: Cyclic steam stimulation of high viscosity oil reservoirs. *Applied Soft Computing*, 64:317–330, 2018.

J. B. Sidney. An extension of Moore's due date algorithm. In *Symposium on the Theory of Scheduling and its Applications (North Carolina State Univ., Raleigh, N. C., 1972)*, pages 393–398. Lecture Notes in Economics and Mathematical Systems, Vol. 86, Berlin, 1973. Springer. Incorporating the results of discussion by Hamilton Emmons and John Rau.

W. E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3(1-2):59–66, 1956.

N. Soni and T. Kumar. Optimization by simulated annealing. *International Journal of Computer Science and Information Technologies*, 5(6):7235 – 7238, 2014.

Y. N. Sotskov and N. V. Shakhlevich. NP-hardness of shop-scheduling problems with three jobs. *Discrete Appl. Math.*, 59(3):237–266, 1995.

D. Spath, M. Hämmerle, and P. Rally. *Wertschöpfung steigern.* Fraunhofer Verlag, Stuttgart, 2010.

K. E. Stecke and J. E. Aronson. Review of operator & machine interference models. *International Journal of Production Research*, 23(1):129 – 151, 1985.

D. Stokes and N. Wilson. *Small business management and entrepreneurship.* Cengage Learning EMEA, 2010.

L. Su and Y. Lee. The two-machine flowshop no-wait scheduling problem with a single server to minimize the total completion time. *Computers & Operations Research*, 35(9):2952 – 2963, 2008.

M. Tan, B. Duan, and Y. Su. Economic batch sizing and scheduling on parallel machines under time-of-use electricity pricing. *Operational Research*, 18(1):105–122, 2018.

Y. Tanimizu, M. Sakamoto, and H. Nonomiya. A co-evolutionary algorithm for open-shop scheduling with disassembly operations. *Procedia CIRP*, 63:289–294, 2017.

N. E. H. Tellache and M. Boudhar. Open shop scheduling problems with conflict graphs. *Discrete Applied Mathematics*, 227:103–120, 2017.

H. Tempelmeier and L. Buschkühl. Dynamic multi-machine lotsizing and sequencing with a simultaneous scheduling of a common setup resource. *International Journal of Production Economics*, 113(1):401 – 412, 2008.

H. Tempelmeier and K. Copil. Dynamic capacitated lot-sizing with parallel common setup operators. Working paper, University of Cologne, 2012.

H. Tempelmeier and K. Copil. Capacitated lot sizing with parallel machines, sequence-dependent setups and a common setup operator. *OR Spectrum*, 38(4):819 – 847, 2016.

H. Tempelmeier. *Material-Logistik*. Springer, Berlin, Heidelberg, New York, 6th edition, 2006.

K. Theile. *Ganzheitliches Management: Ein Konzept für Klein- und Mittelunternehmen*. Haupt, Berlin, 1996.

V. H. Timkovsky. Is a unit-time job shop not easier than identical parallel machines? *Discrete Appl. Math.*, 85(2):149–162, 1998.

V. van de Vrande, J. P. J. de Jong, W. Vanhaverbeke, and M. de Rochemont. Open innovation in smes: Trends, motives and management challenges. *Technovation*, 29(6-7):423 – 437, 2009.

Bureau van Dijk Electronic Publishing GmbH. DAFNE Database. Database, 2014.

F. Villa, E. Vallada, and L. Fanjul-Peyro. Heuristic algorithms for the unrelated parallel machine scheduling problem with one scarce additional resource. *Expert Systems with Applications*, 93:28–38, 2018.

R. W. Vossen. Relative strengths and weaknesses of small firms in innovation. *International Small Business Journal*, 16(3):88 – 94, 1998.

G. Wang and T. C. E. Cheng. An approximation algorithm for parallel machine scheduling with a common server. *The Journal of the Operational Research Society*, 5(2):234 – 237, 2001.

F. Werner and S. Kravchenko. Scheduling with multiple servers. *Automation and Remote Control*, 71(10):2109 – 2121, 2010.

D. C. Whybark and J. G. Williams. Material requirements planning under uncertainty. *Decision Sciences*, 7(4):595 – 606, 1976.

E. D. Wikum, D. C. Llewellyn, and G. L. Nemhauser. One-machine generalized precedence constrained scheduling problems. *Operations Research Letters*, 16(2):87 – 99, 1994.

D. P. Williamson, L. A. Hall, J. A. Hoogeveen, C. A. J. Hurkens, J. K. Lenstra, S.V. Sevast'Janov, and D. B. Shmoys. Short shop schedules. *Operations Research*, 45(2):288–294, 1997.

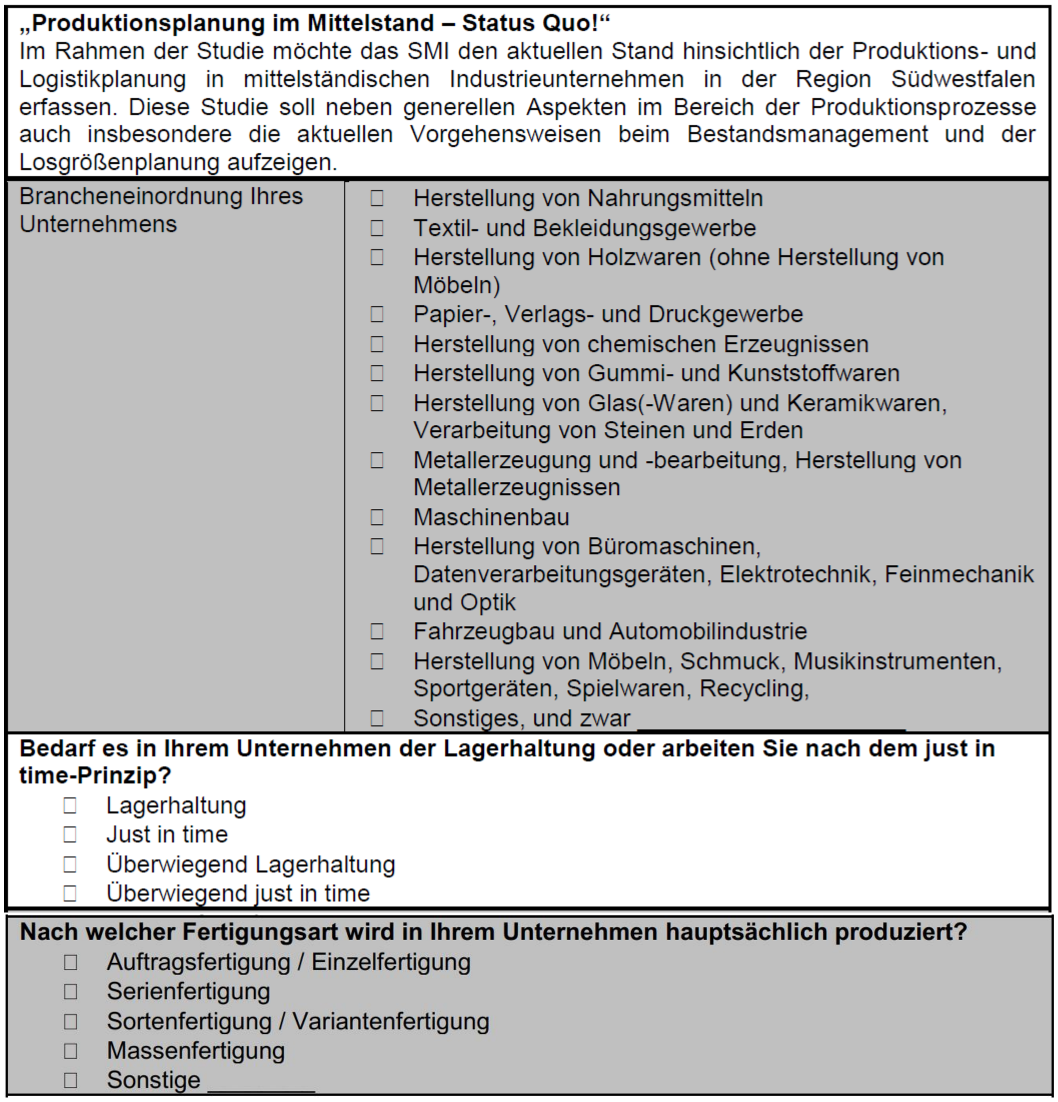
X. Xie, Y. Zheng, and Y. Li. Scheduling parallel machines with a single server: a dedicated case. In *Computational Sciences and Optimization (CSO), 2012 Fifth International Joint Conference on*. IEEE, 2012.

Y. Zeng, A. Che, and X. Wu. Bi-objective scheduling on uniform parallel machines considering electricity cost. *Engineering Optimization*, 50(1):19–36, 2018.

B. Zhao, J. Gao, K. Chen, and K. Guo. Two-generation pareto ant colony algorithm for multi-objective job shop scheduling problem with alternative process plans and unrelated parallel machines. *Journal of Intelligent Manufacturing*, 29(1):93–108, 2018.

Appendix A

Survey



„Produktionsplanung im Mittelstand – Status Quo!"
Im Rahmen der Studie möchte das SMI den aktuellen Stand hinsichtlich der Produktions- und Logistikplanung in mittelständischen Industrieunternehmen in der Region Südwestfalen erfassen. Diese Studie soll neben generellen Aspekten im Bereich der Produktionsprozesse auch insbesondere die aktuellen Vorgehensweisen beim Bestandsmanagement und der Losgrößenplanung aufzeigen.

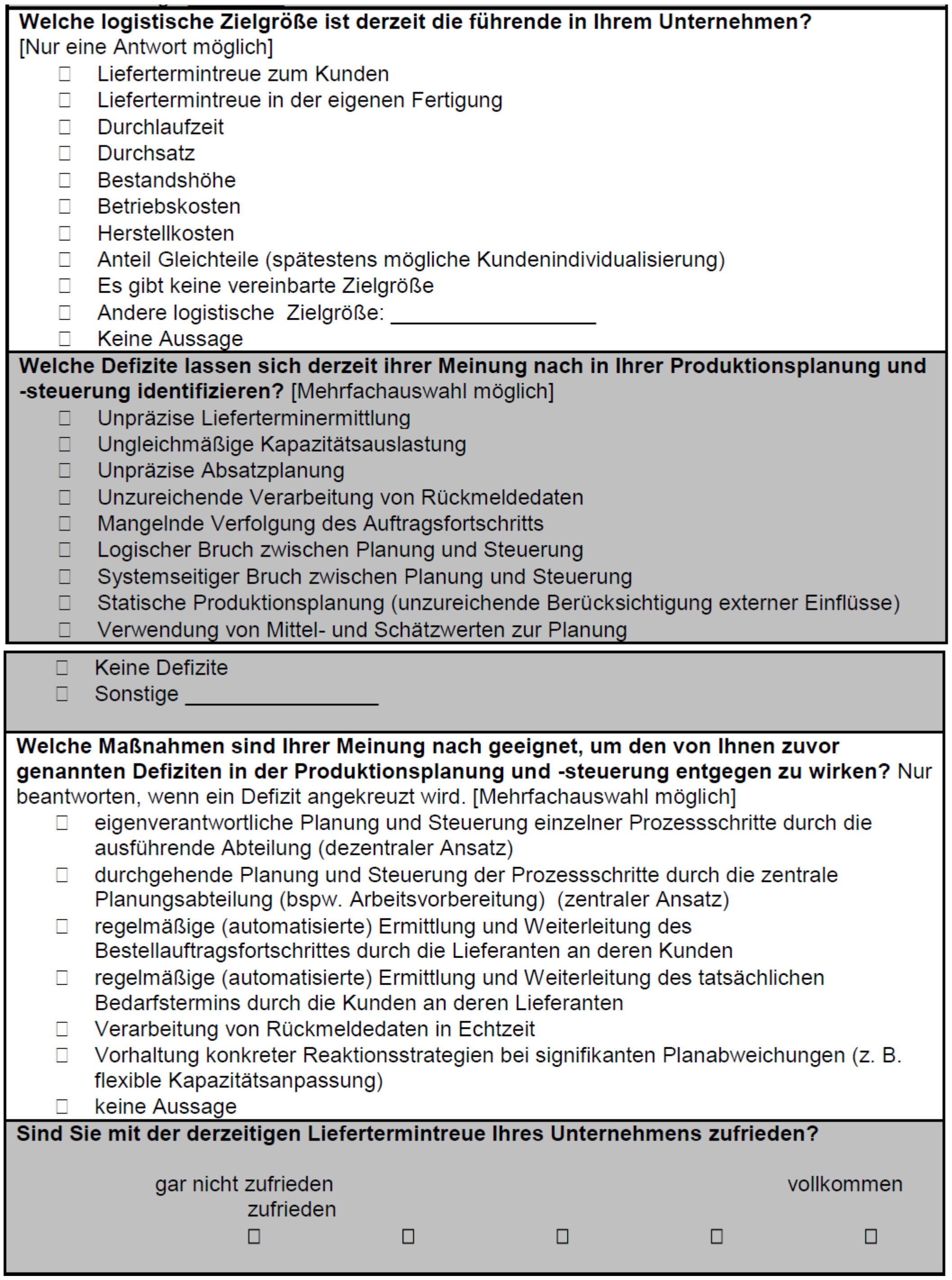
Brancheneinordnung Ihres Unternehmens	☐ Herstellung von Nahrungsmitteln ☐ Textil- und Bekleidungsgewerbe ☐ Herstellung von Holzwaren (ohne Herstellung von Möbeln) ☐ Papier-, Verlags- und Druckgewerbe ☐ Herstellung von chemischen Erzeugnissen ☐ Herstellung von Gummi- und Kunststoffwaren ☐ Herstellung von Glas(-Waren) und Keramikwaren, Verarbeitung von Steinen und Erden ☐ Metallerzeugung und -bearbeitung, Herstellung von Metallerzeugnissen ☐ Maschinenbau ☐ Herstellung von Büromaschinen, Datenverarbeitungsgeräten, Elektrotechnik, Feinmechanik und Optik ☐ Fahrzeugbau und Automobilindustrie ☐ Herstellung von Möbeln, Schmuck, Musikinstrumenten, Sportgeräten, Spielwaren, Recycling, ☐ Sonstiges, und zwar ______________________

Bedarf es in Ihrem Unternehmen der Lagerhaltung oder arbeiten Sie nach dem just in time-Prinzip?

☐ Lagerhaltung
☐ Just in time
☐ Überwiegend Lagerhaltung
☐ Überwiegend just in time

Nach welcher Fertigungsart wird in Ihrem Unternehmen hauptsächlich produziert?

☐ Auftragsfertigung / Einzelfertigung
☐ Serienfertigung
☐ Sortenfertigung / Variantenfertigung
☐ Massenfertigung
☐ Sonstige _________



Welche logistische Zielgröße ist derzeit die führende in Ihrem Unternehmen?
[Nur eine Antwort möglich]

- ☐ Liefertermintreue zum Kunden
- ☐ Liefertermintreue in der eigenen Fertigung
- ☐ Durchlaufzeit
- ☐ Durchsatz
- ☐ Bestandshöhe
- ☐ Betriebskosten
- ☐ Herstellkosten
- ☐ Anteil Gleichteile (spätestens mögliche Kundenindividualisierung)
- ☐ Es gibt keine vereinbarte Zielgröße
- ☐ Andere logistische Zielgröße: ________________
- ☐ Keine Aussage

Welche Defizite lassen sich derzeit ihrer Meinung nach in Ihrer Produktionsplanung und -steuerung identifizieren? [Mehrfachauswahl möglich]

- ☐ Unpräzise Lieferterminermittlung
- ☐ Ungleichmäßige Kapazitätsauslastung
- ☐ Unpräzise Absatzplanung
- ☐ Unzureichende Verarbeitung von Rückmeldedaten
- ☐ Mangelnde Verfolgung des Auftragsfortschritts
- ☐ Logischer Bruch zwischen Planung und Steuerung
- ☐ Systemseitiger Bruch zwischen Planung und Steuerung
- ☐ Statische Produktionsplanung (unzureichende Berücksichtigung externer Einflüsse)
- ☐ Verwendung von Mittel- und Schätzwerten zur Planung
- ☐ Keine Defizite
- ☐ Sonstige ________________

Welche Maßnahmen sind Ihrer Meinung nach geeignet, um den von Ihnen zuvor genannten Defiziten in der Produktionsplanung und -steuerung entgegen zu wirken? Nur beantworten, wenn ein Defizit angekreuzt wird. [Mehrfachauswahl möglich]

- ☐ eigenverantwortliche Planung und Steuerung einzelner Prozessschritte durch die ausführende Abteilung (dezentraler Ansatz)
- ☐ durchgehende Planung und Steuerung der Prozessschritte durch die zentrale Planungsabteilung (bspw. Arbeitsvorbereitung) (zentraler Ansatz)
- ☐ regelmäßige (automatisierte) Ermittlung und Weiterleitung des Bestellauftragsfortschrittes durch die Lieferanten an deren Kunden
- ☐ regelmäßige (automatisierte) Ermittlung und Weiterleitung des tatsächlichen Bedarfstermins durch die Kunden an deren Lieferanten
- ☐ Verarbeitung von Rückmeldedaten in Echtzeit
- ☐ Vorhaltung konkreter Reaktionsstrategien bei signifikanten Planabweichungen (z. B. flexible Kapazitätsanpassung)
- ☐ keine Aussage

Sind Sie mit der derzeitigen Liefertermintreue Ihres Unternehmens zufrieden?

gar nicht zufrieden				vollkommen zufrieden
☐	☐	☐	☐	☐

Wie beurteilen Sie Ihren Lagerbestand?

zu hoch				zu niedrig
☐	☐	☐	☐	☐

Nutzen Sie ein Verfahren zur Losgrößenplanung?

- ☐ Ja
- ☐ Nein

Wenn ja, ist dieses Verfahren systemgestützt?

- ☐ Ja
- ☐ Nein

Welche IT-Systeme sind derzeit in Ihrem Unternehmen im Einsatz? [Mehrfachauswahl möglich]

- ☐ Enterprise Resource Planning (ERP)
- ☐ Produktionsplanung und -steuerung (PPS)
- ☐ Fertigungsfeinplanung (MES)
- ☐ überbetriebliche Planung und Steuerung (SCM)
- ☐ Betriebsdatenerfassung (BDE)
- ☐ Maschinendatenerfassung (MDE)
- ☐ Produktdatenmanagement (PDM)
- ☐ noch zusätzlich andere IT-Systeme
- ☐ kein IT-System
- ☐ keine Aussage

Welche IT-Systeme sind derzeit über eine Schnittstelle mit Ihrem ERP-/PPS-System verbunden? [Mehrfachauswahl möglich]

- ☐ Fertigungsfeinplanung (MES)
- ☐ überbetriebliche Planung und Steuerung (SCM)
- ☐ Betriebsdatenerfassung (BDE)
- ☐ Maschinendatenerfassung (MDE)
- ☐ Produktdatenmanagement (PDM)
- ☐ noch zusätzlich andere IT-Systeme
- ☐ kein IT-System
- ☐ keine Aussage

Falls Sie systemgestützt planen, berücksichtigt Ihr IT System bei der Planung

- ☐ Maschinenkapazitäten
- ☐ Personalkapazitäten
- ☐ Maschineneinrichterverfügbarkeit
- ☐ reihenfolgeabhängige Rüstzeiten
- ☐ Wartung
- ☐ keine Aussage

Vielen Dank für Ihre Teilnahme!