Essays on Heuristic Solution Methods for Combinatorial Optimization Problems

Dissertation

at the Frankfurt School of Finance and Management

submitted by

Michael Hamann

supervised by

Prof. Dr. Jörn-Henrik Thun

Prof. Dr. Jürgen Strohhecker

Prof. Dr. Andreas Größler

Mannheim, March 2015

For Hannah

CONTENTS

LIST OF PAPERS

- Michael Hamann[1] (2014), “A novel pheromone model for ant colony optimization – Normalizing pheromone trails for effective search behaviour”.

- Michael Hamann[1], Jörn-Henrik Thun[2], Andrej Saweljew[3] (2014), “Ant colony optimization for the multi-period mixed-model assembly line sequencing problem”.

- Jürgen Strohhecker[2], Jörn-Henrik Thun[2], Michael Hamann[1] (2014), “Loading and sequencing heuristics for unrelated parallel machine scheduling with long, sequence-dependent setup times”.

[1] Frankfurt School of Finance and Management, Management Department, Sonnemannstr. 9-11, D-60314 Frankfurt am Main, Germany, m.hamann@fs.de

[2] Frankfurt School of Finance and Management, Management Department, Sonnemannstr. 9-11, D-60314 Frankfurt am Main, Germany

[3] University of Mannheim, Center for Doctoral Studies in Business, Graduate School of Economic & Social Sciences, D-68131 Mannheim, Germany

LIST OF FIGURES

LIST OF TABLES

1

An Introduction to Combinatorial Optimization and Metaheuristics

Combinatorial optimization, i.e. the identification of the best arrangement or selection of a finite number of discrete possibilities (Lawler, 1976), has its origin in the economic challenge to efficiently utilize scarce resources and to effectively plan and manage operations. Decision problems in the field of operations management were among the first to be modelled as combinatorial optimization problems, for example the sequencing of machines, the scheduling of production, or the design and layout of production facilities (Grötschel and Lovász, 1995). Today, combinatorial optimization problems are recognised in all areas of management when it comes to the minimization of cost, time, or risk, or the maximization of profit, quality, or efficiency (Talbi, 2009). Typical examples are variants of assignment and scheduling problems, location problems, facility layout problems, set partitioning and set covering problems, inventory control, and traveling salesman or vehicle routing problems (Grötschel and Lovász, 1995; Osman and Kelly, 1996).

The level of problem difficulty of such optimization problems is conceptualized by the theory of computational complexity (Garey and Johnson, 1979; Papadimitriou, 1994; Wolsey and Nemhauser, 2014). In this context, especially two complexity classes are of interest: $\mathcal{P}$ and $\mathcal{NP}$ (whereby the inclusion $\mathcal{P} \leq \mathcal{NP}$ holds). The problem class $\mathcal{P}$ contains all decision problems that can be *solved* in polynomial time in the size of the input on a deterministic sequential machine. These problems are considered as easy and efficiently solvable (Pintea, 2014). The class $\mathcal{NP}$ contains all decision problems for which the *validity* of possible solutions can be verified in polynomial time. Equivalently, they are solvable in polynomial time on a non-deterministic machine. Note that the concept of non-deterministic algorithms is not used to solve combinatorial optimization problems but serves as a means to define the complexity class $\mathcal{NP}$ (Garey and Johnson, 1979). The term *non-deterministic* refers to the fact that there is no actual solution determined – it is only confirmed (in polynomial time) that a given possible solution is correct. A subclass of problems in $\mathcal{NP}$ is called $\mathcal{NP}$*-complete*. These problems are considered to be the hardest to solve in $\mathcal{NP}$. If a decision problem is $\mathcal{NP}$-complete then its corresponding combinatorial optimization problem is called $\mathcal{NP}$*-hard* (Krantz and Parks, 2014; Pintea, 2014).

Many important combinatorial optimization problems are known to be $\mathcal{NP}$-hard, i.e. in the worst case the time needed for solving a problem instance to optimality grows exponentially with its size. Hence, these problems are – although typically easy to describe and understand – difficult to solve. Even for problems of moderate size it is practically impossible to determine all possibilities in order to identify the optimum. As a consequence, heuristic approaches, i.e. approximate solution algorithms, are considered as the only reasonable way to solve hard combinatorial optimization problems (Osman and Kelly, 1996).

Accordingly, there is a vast and still growing body of research on metaheuristics for combinatorial optimization that aim at balancing the trade-off between computation times and solution quality (Blum and Roli, 2003). Metaheuristics provide general frameworks for the creation of heuristic algorithms based on principles borrowed from classical heuristics, artificial intelligence, biological evolution, nervous systems, mathematical and physical sciences, and statistical mechanics (Glover, 1986; Osman and Kelly, 1996). The most important families in the field include but are not limited to greedy randomized adaptive search, evolutionary algorithms, variable neighbourhood search, simulated annealing, tabu search, scatter search, and ant colony optimization (ACO) (Gendreau and Potvin, 2005; Yang et al., 2013).

Although metaheuristics have proven their potential to identify high quality solutions for many complex real-life combinatorial optimization problems from different domains, the effectiveness of any heuristic strongly depends on its specific design (Osman and Kelly, 1996; Glover and Kochenberger, 2003). Hence, the abilities of researchers and practitioners to construct and parameterize heuristic algorithms strongly impact algorithmic performance in terms of solution quality and computation times. As a consequence, there is a need for a deeper understanding of how heuristics need to be designed so that they achieve a high effectiveness when searching the solution spaces of combinatorial optimization problems.

2

PURPOSE AND STRUCTURE OF THE THESIS

This thesis provides insights into heuristic optimization techniques for solving combinatorial optimization problems that occur in various areas of management. It develops and presents heuristic solution methods that are tested on different optimization problems, namely the multidimensional knapsack problem (MKP), the multi-period mixed-model assembly line sequencing problem (MPMMS) and the production-scheduling problem in the case of unrelated parallel machines. All these problems are highly relevant in industrial practice and, being $\mathcal{NP}$-hard, difficult to solve.

The thesis is composed of three articles, which have been submitted to academic journals. The structure is as follows.

Chapter 3 and chapter 4 both present applications of the ant colony optimization metaheuristic. Algorithms of this family, so-called *ant algorithms*, transfer the biological principles of swarm intelligence to mathematical optimization algorithms, i.e. they make use of some of the mechanisms that enable the emergence of intelligent behaviour in swarms of social insects such as termites, bees, or ants (Dorigo et al., 1991a; Dorigo and Stützle, 2003, 2004; for an overview of the most important ACO variants and applications see Dorigo et al., 2006, or Dorigo and Stützle, 2010).

The ACO metaheuristic has been inspired by the foraging behaviour of ants. Its origin is in biological research of the nineteen hundred-nineties. Biologists examined the foraging behaviour of the argentine ant *Iridomyrmex humilis* using the so-called two-bridge experiment where the nest of an ant colony is connected with a food source by bridges of different length. Figure 2.1 shows the original double-bridge experiment as reported by Goss et al. (1989); a) depicts the general experiment design while b) and c) show photos taken during the experiment four and eight minutes after the placement of the bridge.

In the experiment, laboratory colonies of *I. humilis* are given access to a food source connected with the nest by a bridge consisting of two branches of different length. The branches are arranged so that a foraging ant (going in either direction, i.e. from nest to food source or vice versa) has to select one or the other (at two decision points 1 and 2 denoted in Figure 2.1. a), respectively). Except for their length, the branches are basically identical so that the ant has no preference for either of the two branches due to its disposition. Five to ten min after nest and food have been connected, foraging ants have crossed the bridge and discovered the food. In this early phase of the experiment, the foraging ants at first choose equally (randomly) between the short and the long branch. After a while, however, one branch – the short one – becomes visibly preferred. Note that the ant species *Iridomyrmex humilis* studied in this experiment has only a limited individual capability for orientation, i.e. the ants are almost blind. Nevertheless, the ant colony as a whole is capable of identifying the shortest route between nest and food source with great reliability (Aron et al., 1989; Deneubourg et al., 1990; Goss et al., 1989).

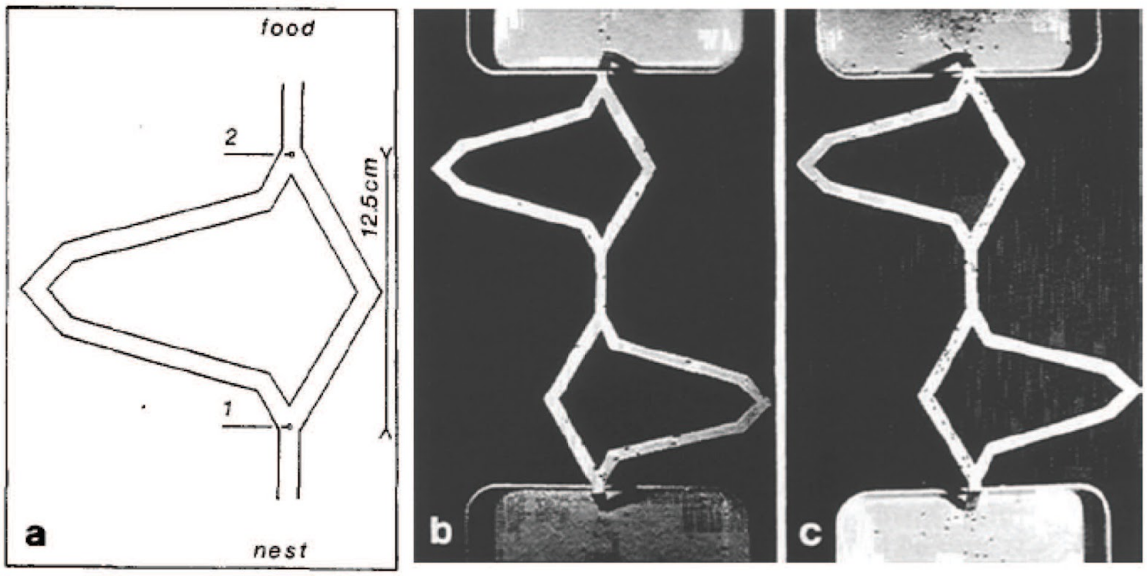

Figure 2.1: Double-bridge experiment with Iridomyrmex Humilis (Goss et al., 1989)

It has been found that the basis for the ability of ant colonies to identify the shortest route between nest and food source (or more generally to "solve shortest-path problems") is an indirect form of communication, which is referred to as *stigmergy* (Grassé, 1959; Garnier et al., 2007): In nature, individual ants deposit chemical messenger substances on their way, so-called *pheromones*, i.e. a moving ant lays pheromone on the ground, thus marking its path by a trail of this substance. Functioning as an attractant, other ants of the colony are able to perceive – "smell" – these pheromones. The higher the concentration of pheromone, the more attractive is the respective way, i.e. the probability that other ants choose the same way increases with the pheromone concentration (Dorigo et al., 2000).

Based on this indirect communication via pheromones, the double-bridge experiment can be explained as follows. In the beginning, there is no pheromone on either of the branches of the bridge linking nest and food. Hence, there is no stimulus for the ants to prefer one way or the other. When the first ants arrive at decision point 1 on their way from the nest to the food source, they have no clue about which is the best choice, and, as a consequence, randomly choose one of the branches. Due to the different length of the branches, ants using the short branch arrive at the food roughly 20 seconds before ants using the long branch and, making their way back to the nest, reach decision point 2 even before the ants on the long branch have passed this point on their way from the nest to the food source. As a consequence, at this point in the experiment there is still no pheromone on the long branch while on the short branch there is a pheromone trail (deposited by the first group of ants during their first passage). This now influences the decision of the ants: the short branch is more attractive due to the pheromone trail and will be preferred by the ants, leading to a further accumulation of pheromone on the short branch. When the second group of ants reaches decision point 2 on its way back, it is faced with a similar situation. There are pheromone trails on both branches; however, the trail on the short branch is stronger since the first group of ants has passed it twice already. Again, the short branch is more attractive and is chosen by the second group of ants with a high probability. In the course of the experiment, the difference in the amount of pheromone on the two branches becomes larger, which in turn increases, with a positive feedback effect, the number of ants choosing the shorter route so that in the end basically all ants choose the short branch (Goss et al., 1989; Dorigo and Gambardella, 1997).

In the ACO metaheuristic, artificial ants communicate in a similar fashion during their iterative solution construction. Analogous to the deposit of real pheromone by real ants on the way between nest and food source in the double-bridge experiment, artificial ants deposit artificial pheromone on their solutions for the respective optimization problem. More precisely, the artificial ants follow the artificial pheromone trails during the solution construction process, i.e. solution components with stronger pheromone trails will be selected with a higher probability. The pheromone deposit by individual artificial ants depends on the solution quality. Good solutions will receive a larger amount of pheromone than weak solutions in order to guide the search of the artificial ant colony towards good solutions. Hence, one could say that individual ants exchange information about the quality of chosen solutions via pheromone trails, i.e. the higher the pheromone concentration, the better (and more attractive for other ants) the respective solution (at least theoretically). Consequently, the pheromone trails on superior solutions (or their components, respectively) typically grow stronger during the search, while the trails on inferior solutions remain weak or fade completely over time. Although individual ants are able to generate feasible solutions, solutions of good quality are usually only achieved through this kind of interaction and indirect communication within the colony (Dorigo and Stützle, 2004). Hence, the pheromone trails can be considered as the global memory of the colony where information about desirable solution components is stored (Boysen, 2005). Based on this information the colony is able to iteratively improve the solutions until the (near) optimal solution is found.

Chapter 3 introduces a novel pheromone model for ant colony optimization, which aims at improving the search behaviour of artificial ants so that the ant colony explores the solution space of a combinatorial optimization problem more effectively. In order to evaluate the search behaviour, different performance measures are analysed that help to understand how solutions iteratively develop in the course of the search. Moreover, the algorithm is tested on various benchmark MKP problems from the literature. The results are compared to a selected set of ant algorithms.

Chapter 4 presents a novel multi-period variant of the mixed-model sequencing problem and develops a correspondingly adapted ant algorithm, which makes use of the pheromone concept introduced in chapter 3. The algorithm is tested on 48 different randomly generated problem instances of various size and compared with the solver Gurobi as well as problem-specific probabilistic and deterministic greedy algorithms.

In chapter 3 the focus is on the methodological advancement of the ant colony optimization metaheuristic. The main objective is to analyse and improve the design and search behaviour of a generic problem-unspecific ant algorithm, not to find an algorithm that performs particularly well for a specific optimization problem. Chapter 4 concentrates on the introduction of a novel optimization problem, namely the multi-period mixed-model assembly line sequencing problem. Hence, the presented algorithm features more problem-specific intelligence and is less generic in nature.

Although recent research shows that ant colony optimization belongs to the most promising heuristic techniques, the question remains whether such complex methods are justified in practice or if – at least in some cases – simpler approaches can achieve satisfying results in real-world settings. Hence, in chapter 5, a different perspective is taken. With a stronger focus on practicality, the study presented in this chapter investigates some less complex heuristic techniques for production scheduling on unrelated parallel machines such as, e.g., the *earliest due date* or the *shortest processing time* heuristic. Since short-term planning typically involves the use of rolling planning horizons in practice (de Araujo et al., 2007), the number of problem components of respective decision problems such as, e.g., the optimization of the production sequence, remains comparably small. Therefore, even simple heuristic approaches might be a feasible alternative to more sophisticated metaheuristics such as, for instance, ant colony optimization, genetic algorithms, simulated annealing, scatter search, or tabu search. The specific scheduling problem considered in chapter 5 comprises two sub problems, i.e. first, the assignment of production orders to machines and, second, the determination of the production sequence. Accordingly, combinations of heuristic algorithms for both sub problems are investigated by simulating different randomized real-world scenarios generated based on genuine data taken from an international pharmaceutical company. The analysis of overall 108 unique strategy-scenario combinations provides a robust view of the interaction of loading and sequencing heuristics and enables an assessment of the potential of simpler heuristic techniques in unrelated parallel machine settings under different operational conditions.

Chapter 6 summarizes and concludes the thesis.

3

A NOVEL PHEROMONE MODEL FOR ANT COLONY OPTIMIZATION – NORMALIZING PHEROMONE TRAILS FOR EFFECTIVE SEARCH BEHAVIOUR

Michael Hamann

Abstract

This paper introduces a novel generic pheromone model for algorithms of the ant colony optimization metaheuristic, which improves the searching behaviour of the artificial ants. The algorithm, NormANTS, is based on a normalization of pheromone trails between zero and one and a coupling of the iterative pheromone deposit with the total number of iterations. In order to investigate the impact of the pheromone model, a detailed analysis of the search behaviour of the artificial ants in terms of intensification and diversification is conducted. Moreover, NormANTS is applied to the multidimensional knapsack problem and tested on 33 different knapsack instances from the literature. In order to evaluate performance, the algorithm is compared to various ant-based approaches that differ regarding their level of problem specificity and complexity. In the computational study, NormANTS exhibits a high degree of effectiveness regarding the sampling of the search space. Overall, the algorithm consistently generates optimal solutions and achieves a particularly high average solution quality. While the algorithm is applied to the multidimensional knapsack problem in this paper, the pheromone model can be easily transferred to other combinatorial optimization problems due to its generic nature.

3.1 Introduction

The ant colony optimization metaheuristic comprises a wide range of search heuristics inspired by the foraging behaviour of real ants (Dorigo et al., 1996). Indirect communication between individual ants enables ant colonies to find shortest paths between their nest and food sources (Deneubourg et al., 1990). The same principle, i.e. communication via chemical substances called pheromones, is used in artificial ant colonies in the field of combinatorial optimization (Dorigo and Stützle, 2004). Since the early nineties, numerous applications for well-known problems such as the traveling salesman problem (TSP) (Cordón et al., 2000; López-Ibáñez et al., 2013), the knapsack problem (Boryczka, 2007; Ke et al., 2010; Leguizamón and Michalewicz, 1999), or the quadratic assignment problem (Gambardella et al., 1999; Stützle and Hoos, 2000) have been presented.

In this study, a novel pheromone model for ACO is proposed which aims at an improved effectiveness in terms of search space sampling. It is based on a normalization of pheromone trails between zero and one and a coupling of the iterative pheromone deposit with the total number of iterations. Correspondingly, the algorithm is named *NormANTS*.

In order to analyse the performance of the algorithm, a computational study is carried out in which NormANTS is applied to the multidimensional knapsack problem. Note that the algorithm and the pheromone concept introduced in this paper are highly generic and thus could have been applied to other combinatorial optimization problems as well. The MKP has been chosen in the computational study since there are numerous test problems in the literature for which results of different ant algorithms with traditional pheromone concepts are available. Overall, 33 different knapsack instances ranging from small to large size are solved in the experimental study. The solutions obtained by different alternative algorithms based on the ACO metaheuristic serve as benchmarks and enable an evaluation of the results of NormANTS.

The contribution of this study with respect to past research in this field primarily lies in (i) the introduction of a novel pheromone model that differs regarding the quantification of pheromone trails and the definition of the pheromone delta for the update procedure, (ii) a detailed analysis of the resulting search behaviour of the artificial ants, and (iii) an experimental comparison with state-of-the-art ACO-based approaches from the literature for the multidimensional knapsack problem.

The remainder of this paper is organized as follows. Section 3.2 provides a brief overview of the pheromone models of the most successful ACO variants and discusses implications for the search behaviour of the artificial ants. Section 3.3 reviews the relevant literature in the field of ant colony optimization and multidimensional knapsack problem. The characteristics and constituent properties of NormANTS are presented in section 3.4. In section 3.5, the design of the experimental study, the analysis of the search behaviour, and the computational results are reported. In the last section, the paper concludes with a brief discussion of the results and future research topics.

3.2 Quantification and update of pheromone trails in the ant colony optimization metaheuristic

Traditionally, in applications of the ACO metaheuristic the increase of the pheromone trails during the pheromone update procedure is defined as some function of the quality of the solutions found. More precisely, the pheromone delta either depends on the objective function value obtained by the ant that modifies the trails (see, e.g., Dorigo et al., 2006), or the number of ants that generated the respective solution (see, e.g., Seçkiner et al., 2013).

In the following, a brief comparison of the most successful variants of ACO (Dorigo and Socha, 2007), namely *Ant System (AS)* (Dorigo et al., 1991b, 1996), *Ant Colony System (ACS)* (Dorigo and Gambardella, 1997; Gambardella and Dorigo, 1996), and *MAX-MIN Ant System (MMAS)* (Stützle and Hoos, 2000) illustrates the typical logic of quantifying the iterative pheromone delta. Note that originally all these algorithms have been applied to the traveling salesman problem so that the original formulas make use of problem-specific terminology. The definitions provided in the remainder of this section abstract from the TSP but are otherwise identical.

The first algorithm, AS, computes the amount of pheromone deposited on a solution component i by ant k as

$$\Delta\tau_i^k = \begin{cases} Q/L_k & if\ ant\ k\ used\ component\ i\ in\ its\ solution, \\ 0 & otherwise \end{cases} \tag{3.1}$$

where Q is a constant and L_k is the objective function value of the solution constructed by ant k. The pheromone delta used by MMAS is defined in a similar way with

$$\Delta\tau_i^{best} = \begin{cases} 1/L_k & if\ the\ best\ ant\ used\ i\ in\ its\ solution, \\ 0 & otherwise \end{cases} \tag{3.2}$$

whereby only the best ant (either of the iteration or globally) is allowed to lay pheromone on its solution components. The third algorithm, ACS, uses the same definition of $\Delta\tau_i^{best}$ as MMAS but additionally conducts a local update in which each ant deposits pheromone only to the last solution component chosen.

As can be seen from equations (3.1) and (3.2), the pheromone delta depends on specifics of the current problem such as actual objective function values and the corresponding distance between good and bad solutions. Hence, the amount of pheromone that is deposited at the end of each iteration varies for different problems and even for different ants of the same colony optimizing one specific problem (given that more than one ant is allowed to carry out an update and the ants found solutions of different quality). Although this inherent dynamic in the pheromone update and trail development is intended in order to favour convergence towards good solutions, there remains a certain pitfall to this concept: It is difficult to understand how much pheromone is deposited by individual ants in each iteration, and, as a result, how quickly pheromone trails develop and how strong the pheromone trails become on individual solution components in the course of the search. Consequently, without further knowledge about the particular problem instance at hand, it is not fully clear in what way the solution space is actually browsed by the ants and to what extent the search is explorative or exploitative.

Even if good values are found for the set of parameters that controls an ant algorithm, the problem persists that to a certain extent the evolution of the pheromone trails remains a black box. This makes it challenging to guarantee for effective search behaviour without premature convergence or stagnation situations.

The pheromone model introduced in this paper intents to remove this black box, i.e. it aims at obtaining a higher level of control over the pheromone trail development independently from the specific optimization problem at hand.

3.3 Ant colony optimization for the multidimensional knapsack problem

Since the MKP is used in the experimental study of this paper, this section provides a brief overview of this combinatorial optimization problem and discusses the most relevant aspects when applying ACO to the MKP.

The objective of the multidimensional knapsack problem is to find a subset of objects that maximizes the total profit while satisfying several resource constraints. As the resources are typically denoted with index m, the MKP is also referred to as the m-dimensional knapsack problem. In line with the definition given by Chu and Beasley (1998), the MKP can be formulated as follows:

$$maximize \quad \sum_{j=1}^{n} p_j x_j \tag{3.3}$$

$$subject\ to \quad \sum_{j=1}^{n} r_{ij} x_j \leq b_i \qquad \forall\ i = 1, \dots, m \tag{3.4}$$

$$x_j \in \{0,1\} \tag{3.5}$$

The objective function, i.e. the maximization of the total profit, is given in equation (3.3) with p_j being the profit of an object j and x_j the corresponding binary decision variable (equation (3.5)). If x_j is equal to one, object j is selected leading to the realization of p_j. Equation (3.4) reflects the m resource constraints whereby r_{ij} is the quantity that is consumed of resource i if object j is selected. The overall available amount of resource i is given by b_i. Note that a MKP is considered as well-stated if the assumptions $p_j > 0$ and $r_{ij} \leq b_i < \sum_{j=1}^{n} r_{ij}$ hold.

In the literature, there are several papers that solve this well-known $\mathcal{NP}$-hard problem using ant colony optimization. Leguizamón and Michalewicz (1999) were the first to present an ant algorithm for the multidimensional knapsack problem. In their paper, they suggest an adapted version of AS which was introduced by Dorigo et al. (1991b). Boryczka (2007) also proposes a modified version of AS making use of a different transition rule. Alaya et al. (2004) introduce an adapted version of MMAS as presented by Stützle and Hoos (2000), which successfully implements pheromone trails on pairs of objects. The MMAS is also the basis for the algorithm presented by Ke et al. (2010) who achieve an improved search behaviour by dynamically adjusting the lower pheromone boundary. The algorithm of Kong et al. (2008) applies a pheromone laying method specifically designed for the binary solution structure, which allows the generation of infeasible solutions during solution construction.

The majority of ACO-based applications for the MKP use heuristic factors in form of a pseudo-utility value in order to provide some local information about solution components and thus enable a more directed search. Although static definitions were tried (see, e.g., Fidanova, 2002), the majority of algorithms uses the dynamic approach suggested by Leguizamón and Michalewicz (1999), who define the heuristic factor of an object (analogously to the efficiency ratio of Dobson (1982)) as the ratio between its value and its average requirements regarding the remaining capacities (equation (3.6)). Reflecting the current state of the solution construction, $\tilde{S}_t$ is defined as the set of selected objects at the t^{th} construction step. For each of the remaining objects j, the corresponding heuristic value $\eta_j(\tilde{S}_t)$ is denoted with

$$\eta_j(\tilde{S}_t) = \frac{p_j}{\bar{\delta}_j(t)} \tag{3.6}$$

The average tightness $\bar{\delta}_j(t)$ of an object j, i.e. its average demand of the remaining resources, is given as

$$\bar{\delta}_j(t) = \frac{\sum_{i=1}^{m} \delta_{ij}(t)}{m} \tag{3.7}$$

with

$$\delta_{ij}(t) = \frac{r_{ij}}{b_i - \sum_{l \in \mathfrak{S}_t} r_{il}} \tag{3.8}$$

Besides the definition of heuristic values, the choice of an appropriate pheromone representation is considered as essential. In recent years, different options for laying pheromone trails have been described in the literature, i.e. associating pheromone trails directly with each object (Leguizamón and Michalewicz, 1999), depositing pheromone on each pair of successively selected objects (Fidanova, 2002), or laying pheromone trails on all pairs of objects (Alaya et al., 2004).

Following the intuition to use the least complex and most obvious pheromone representation for a specific problem, the algorithm developed in this study applies the simplest type, i.e. vertex-oriented pheromone trails.

3.4 NormANTS – Combining normalized pheromone trails with an iteration-dependent pheromone delta

NormANTS is based on the MAX-MIN Ant System introduced by Stützle and Hoos (2000). More precisely, it applies two key features of MMAS, i.e. it sets a minimum and a maximum boundary for pheromone values, τ_{min} and τ_{max}, and it uses the same tye of transition probability. However, NormANTS incorporates a novel pheromone model, which substantially differs in terms of quantification of pheromone trails and update procedure.

Solutions are constructed as follows: In each iteration, every ant k subsequently selects objects from the set of remaining feasible objects, $candidates_k$, that do not violate any of the knapsack constraints. When no object satisfying the capacity constraints is left, the ant stops its search for the current iteration.

The transition probability with which an ant decides to select an object j is defined as

$$p_j = \begin{cases} \frac{[\tau_j]^\alpha [\eta_j]^\beta}{\sum_{l \in candidates_k} [\tau_l]^\alpha [\eta_l]^\beta} & if\ j \in candidates_k, \\ 0 & else, \end{cases} \tag{3.9}$$

The intensity of the pheromone trails on an object j is denoted with τ_j, whereas η_j is the heuristic value. For the MKP instances in the experimental study, NormANTS applies the dynamic computation of pseudo-utility values as described in section 3.3, i.e. the local attractiveness η_j of an object j is defined as its value divided by its average tightness. The parameters α and β balance the relative importance of pheromone and heuristic values.

As the designation of the set $candidates_k$ implies, candidate lists are used, i.e. only a subset containing a predefined number of the most attractive objects is considered during the current step of the solution construction (Dorigo and Stützle, 2003; Gambardella and Dorigo, 1996). Hence, an ant k does not choose from all remaining feasible objects that have not been selected yet, but from the subset $candidates_k$, which contains the orders with the highest heuristic values.

Regarding the pheromone update, an elitist strategy is used (Bullnheimer et al., 1997; Dorigo et al., 1996; Stützle and Hoos, 2000). At the end of each iteration the iteration-best ant, $k^{+^{iter}}$, and the best ant so far, $k^{+^{global}}$, are allowed to deposit pheromone on their respective solution components. Combined with the iterative pheromone reduction through evaporation with rate ρ, the trails are updated with

$$\tau_j = (1 - \rho)\, \tau_j + \Delta\tau_j^{k^{+^{iter}}} + \Delta\tau_j^{k^{+^{global}}} \tag{3.10}$$

As discussed in section 3.2, traditional ways to quantify pheromone trails and the iterative pheromone delta create certain dependence from specifics of the current problem instance such as actual objective function values and the corresponding distance between good and bad solutions. Hence, the pheromone limits are typically chosen problem-dependent so that the distance between τ_{min} and τ_{max} somehow fits to the expected rate of growth of the pheromone trails. Stützle and Hoos (2000) define τ_{min} as a function of τ_{max} (with $\tau_{min} = \varepsilon\tau_{max}$) and suggest computing τ_{max} as an estimate of the asymptotically maximum pheromone trail value. However, this value depends on the amount of pheromone that is deposited during the search, which again varies from one problem to another (Solnon and Fenet, 2006).

In order to achieve a higher level of control in this regard, NormANTS introduces the following two instruments. First, heuristic values and pheromone values are normalized between zero and one. Accordingly, the pheromone deltas for the iteration-best and the globally best ant so far, $\Delta\tau_j^{k^{+iter}}$ and $\Delta\tau_j^{k^{+global}}$, are defined as

$$\Delta\tau_j^{k^{+iter}} = \vartheta^{\Delta\tau^{iter}} \cdot \rho \left(\pi^{k^{+iter}} - \pi^{k^{-global}}\right) / \left(\pi^{k^{+global}} - \pi^{k^{-global}}\right) \tag{3.11}$$

and

$$\Delta\tau_j^{k^{+global}} = \vartheta^{\Delta\tau^{global}} \cdot \rho \left(\pi^{k^{+global}} - \pi^{k^{-global}}\right) / \left(\pi^{k^{+global}} - \pi^{k^{-global}}\right) \tag{3.12}$$

, respectively. The variable π denotes the objective function value of the solution generated by some ant k, with k^+ (k^-) being the best (worst) ant either of the current iteration (k^{+iter}) or so far ($k^{+global}$). $\vartheta^{\Delta\tau^{global}}$ and $\vartheta^{\Delta\tau^{iter}}$ are control parameters that can be used to adjust the weight of the different elitist ants and to increase or decrease the corresponding pheromone delta. In this regard, emphasizing the iteration-best ant supports exploration while focusing on the globally best ant intensifies exploitation (Levine and Ducatelle, 2004). Besides the pheromone also the heuristic values are normalized following the same principle, i.e.

$$\eta_j^{normalized} = \left(\eta_j - \eta^{worst}\right) / \left(\eta^{best} - \eta^{worst}\right) \tag{3.13}$$

This prevents distortions regarding the relative importance of pheromone and heuristic values in the calculation of transition probabilities.

Second, the evaporation rate ρ is defined as a function of the total number of iterations of the search procedure with

$$\rho = 1/\ total\ number\ of\ iterations \tag{3.14}$$

Combined, these two instruments work as follows. Using normalized values decouples the pheromone delta from individual characteristics of problem instances while information about the relative distance between good and bad solutions remains available.

In combination with the limited range of the normalized pheromone values, the direct link between ρ and the number of iterations ensures that the pheromone delta (which is also a function of ρ) is completely under control. More precisely, the number of iterations directly determines the size of the delta so that the amount of pheromone deposited at the end of each iteration is on the one hand large enough to enable the evolution of an adequately differentiated pheromone map throughout the optimization run and on the other hand small enough to keep the difference between objects under control.

3.5 Computational study

3.5.1 Experimental design

In the computational study, NormANTS is applied to the multidimensional knapsack problem. It is tested on 33 different MKP instances from the literature. In order to evaluate both the impact of the novel pheromone concept as well as the overall performance, the results of NormANTS are compared with different benchmarks.

First, in order to be able to associate observed differences in performance with the pheromone concept, ant algorithms that use a vertex-oriented pheromone model such as Leguizamón and Michalewicz (1999), Boryczka (2007), and Ke et al. (2010) are selected. Besides the similarity regarding pheromone representation, none of these algorithms includes any kind of additional knapsack specific local search procedure that might significantly improve weak results generated during the artificial ants search and thus distort the comparison. Note that with its dynamic adjustment of the pheromone limits *DMMAS* of Ke et al. (2010) is more complex than the algorithms of Leguizamón and Michalewicz (1999) and Boryczka (2007), who follow the traditional pheromone update models discussed in section 3.2. Accordingly, the most appropriate benchmarks for an evaluation of the impact of the novel pheromone model with normalized pheromone trails are the ant algorithms of Leguizamón and Michalewicz (1999) and Boryczka (2007).

Second, to analyse the overall performance independently from the pheromone model, the results of Alaya et al. (2004), *Ant-knapsack*, are considered additionally. This algorithm deposits pheromone trails on all pairs of objects (instead of individual vertices).

MKP test instances were selected in line with the above so that a satisfying number of suitable results were available for the respective benchmark algorithms. In order to analyse the overall performance of NormANTS, problems of different size were chosen including instances that used to be particularly hard to solve. All test problems used have been downloaded from OR-Library (Beasley, 1990).

The algorithm described in this study has been implemented in MatLab R2012a. All test runs have been conducted on a Macintosh computer with an Intel Core i5 processor (1.7 GHz) and 4 GB RAM.

3.5.2 Parameter settings and evolution of pheromone trails

For all problem sets, the number of iterations and ants were set to 1000 and 100, respectively. The parameters that balance the relative importance of pheromone trails and heuristic values were set to $\alpha = 2$ and $\beta = 3$. Compared with the parameter configurations reported by Leguizamón and Michalewicz (1999) or Boryczka (2007), who both suggest lower values for α and higher values for β, the chosen setting results in a relatively strong influence of the pheromone. This supports a rather explorative search in the early phase (when all trails are still undeveloped and close to their initial values) and a rather exploitative search towards the end of an optimization run (when the pheromone map is differentiated and potentially desirable objects are intensely marked). The initial pheromone values were defined as $\tau^{ini} = 0.5$. The upper and lower bound for the pheromone trails were set to $\tau^{min} = 0.1$ and $\tau^{max} = 1$, respectively. The evaporation rate was defined as $\rho = 1/total\ number\ of\ iterations = 0.001$. The control factors $\vartheta^{\Delta\tau^{iter}}$ and $\vartheta^{\Delta\tau^{global}}$ for the update of the iteration-best and globally best ant were set to $\vartheta^{\Delta\tau^{iter}} = \vartheta^{\Delta\tau^{global}} = 0.5$.

Figure 3.1 exemplarily shows the iterative pheromone deposit and evaporation for problem 5.100-00, one of the knapsack instances taken from Chu and Beasley (1998). The chosen values for τ^{ini}, τ^{min}, τ^{max}, ρ, $\vartheta^{\Delta\tau^{iter}}$, and $\vartheta^{\Delta\tau^{global}}$ resulted in an average pheromone deposit of 1.4 and an average evaporation of 0.37 per solution component over 1000 iterations. On the one hand there is a clear differentiation between good and bad objects, which prevents the ant's search from becoming too diversified. On the other hand, the absolute differences in terms of pheromone values do not grow too large. Accordingly, many objects remain actually (not only theoretically) selectable throughout the whole optimization run. This minimizes the risk of stagnation.

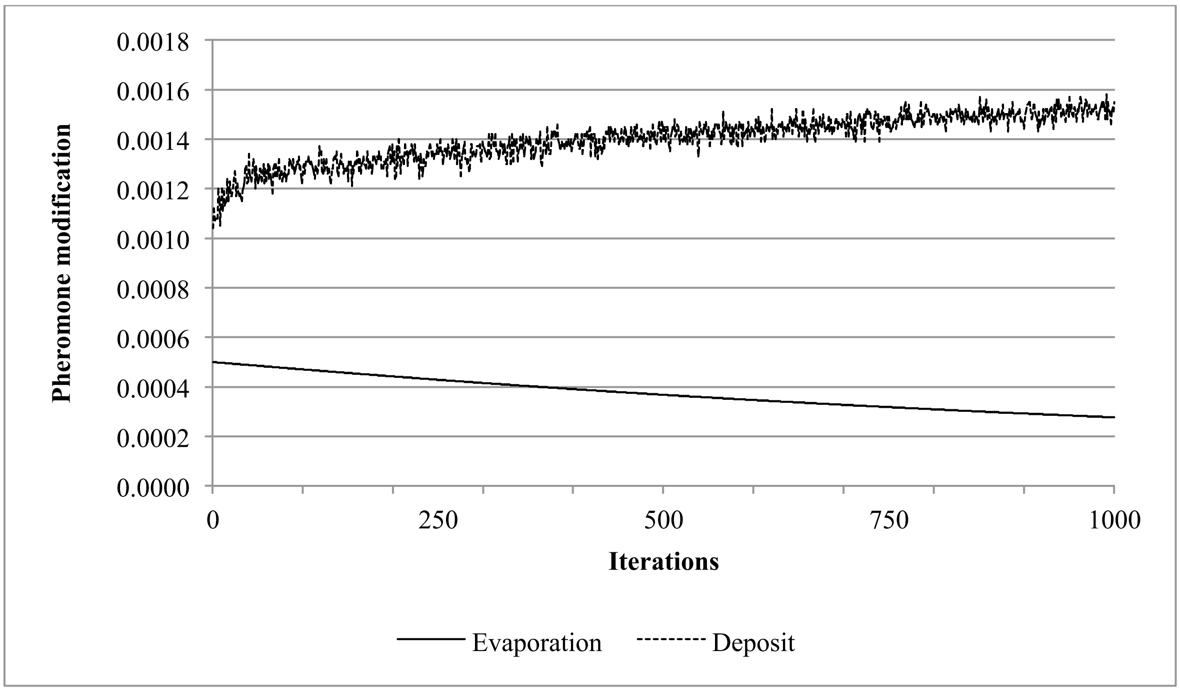

Figure 3.1: Iterative pheromone deposit and evaporation for 5.100-00 over 1000 iterations (average over ten runs)

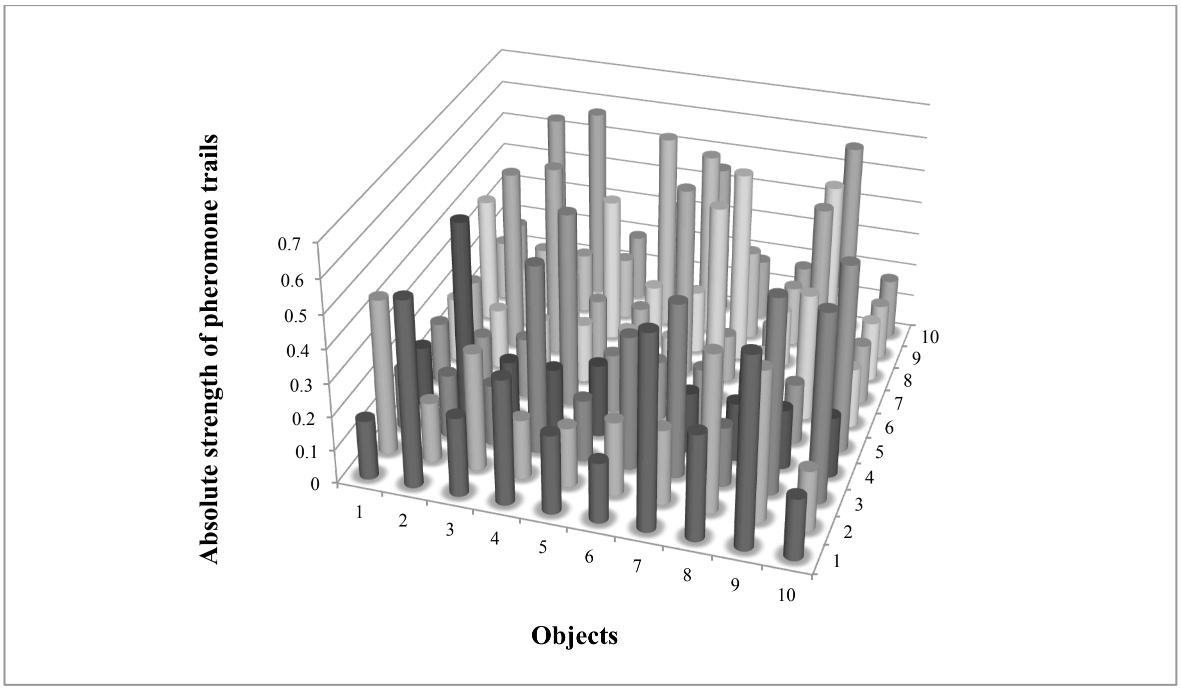

Figure 3.2: Differentiated pheromone map for 5.100-00 after 1000 iterations (average over ten runs)

The fact that the iterative pheromone delta is independent from absolute objective function values of the problem instance and linked to the number of iterations enables a slow and coordinated evolution of the pheromone trails. As can be seen from Figure 3.1, both pheromone deposit and evaporation remain on a relatively constant level. This is essential for maintaining a high degree of control over the incremental modification of the trails. Note that without further adjustments of any of the parameters, the average pheromone deposit and evaporation are highly similar for all other instances from the computational study. This reduces the necessity for problem instance-specific adjustments of the parameter settings of the ant algorithm.

As a consequence of the regulated pheromone evolution, the solution quality is gradually improved in the course of the ants' search. A stronger convergence to solutions occurs only towards the end of the optimization run. However, if need be for a more aggressive search, a reduction of the number of iterations would automatically trigger an adjustment of the cumulated pheromone delta per iteration. For example, setting the number of iterations to 100 resulted in an average evaporation of 0.004 and an average deposit of 0.0133 per object and iteration for problem instance 5.100-00 (i.e. roughly ten times the value than for 1000 iterations). Accordingly, the pheromone trails developed significantly faster so that a differentiation similar to the one depicted in Figure 3.2 was reached after 100 iterations already. In addition, one could also increase the pheromone deposit by setting higher values for $\vartheta^{\Delta\tau^{iter}}$ and $\vartheta^{\Delta\tau^{iter}}$. This would also support a stronger intensification leading to larger differences between "good" and "bad" solution components.

3.5.3 Search behaviour analysis: diversification and intensification

In order to further analyse the ability of NormANTS to effectively explore the search space two measures, namely resampling ratio and similarity ratio, are considered. Both metrics have been used in the field of ACO to evaluate the diversification and intensification of ant algorithms (Ke et al., 2010; Solnon and Fenet, 2006). After a brief definition of the metrics, the results are given in Figure 3.3 and 3.4. Note that in both figures, the plotted values for DMMAS and MMAS are only rough estimates as they have been derived from the corresponding graphs reported by Ke et al. (2010). However, the fitted curves are sufficiently accurate to reflect the general behaviour of DMMAS and MMAS and thus enable a schematic comparison with NormANTS.

Resampling ratio

Analogous to the definitions provided by Van Hemert and Bäck (2002) and Solnon and Fenet (2006) the resampling ratio in the t^{th} iteration is computed as

$$Resampling\ Ratio = \frac{(NumberSolutionsTotal - NumberSolutionsUnique)}{NumberSolutionsTotal} \tag{3.15}$$

where $NumberSolutionsTotal$ is the total number of solutions generated so far (i.e. until the t^{th} iteration) and $NumberSolutionsUnique$ is the number of unique solutions found in iteration t. Note that "unique" refers to all solutions generated during the iterations 1 to $(t-1)$. A resampling ratio close to zero implies an effective search with only few duplicate solutions whereas values close to one indicate stagnation, i.e. hardly any new solutions are generated in the course of the search.

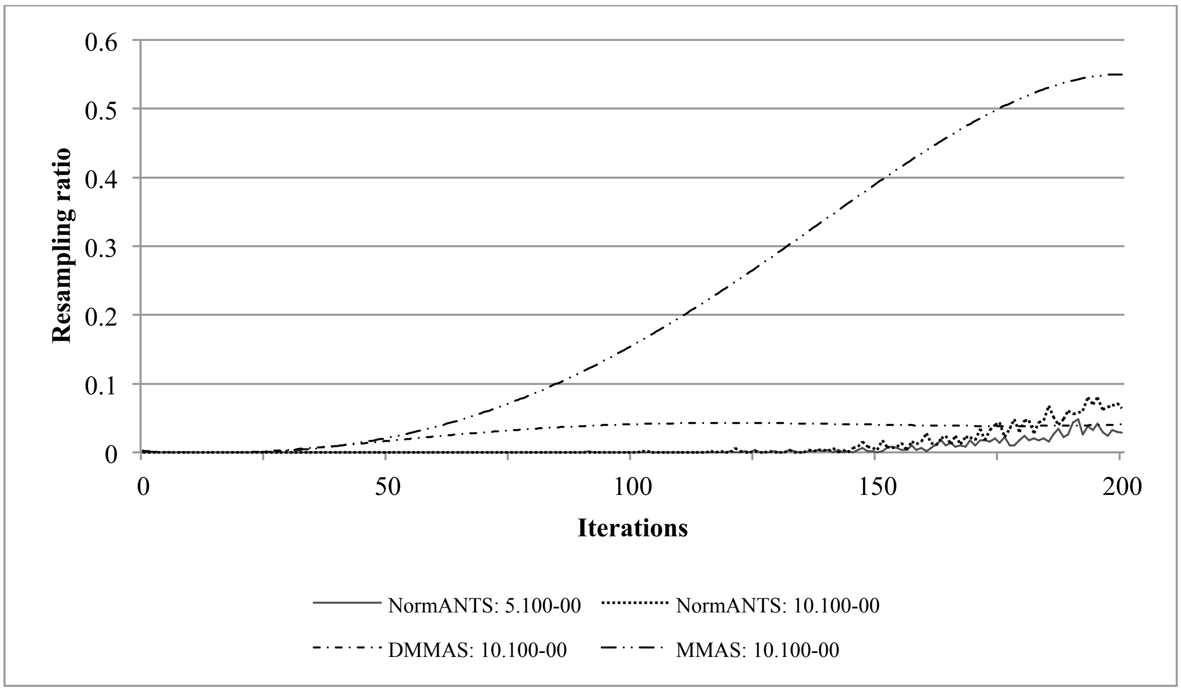

Figure 3.3: Evolution of the resampling ratio for 5.100-00 and 10.00-00 over 200 iterations (average over ten runs)

Figure 3.3 shows the average resampling ratio for the instances 5.100-00 and 10.100-00 over ten runs. The number of iterations was set to 200 (accordingly, $\rho = 1/200$). All other parameter values were chosen as reported above. For both instances, the resampling ratio of NormANTS remains below 0.05 until three quarters of the iterations are completed and clearly below 0.1 at all times. Only towards the end, a slight increase can be observed.

However, this is actually a good sign as it indicates a certain degree of intensification during the final phase of the search. For MMAS with a traditional pheromone model the resampling ratio develops in a substantially different way. Already in an early phase, i.e. after about 50 iterations, a quite steep increase can be observed, ultimately leading to a resampling ratio of around 0.55 in iteration 200. Hence, it can be stated that NormANTS is significantly more effective in sampling the search space than the original MMAS. Note that the only difference between these two algorithms is the novel pheromone model of NormANTS, which appears to have a strong positive impact on the search behaviour. The comparison with DMMAS shows that both algorithms maintain good resampling ratios throughout the optimization. However, for NormANTS the ratio increases considerably later, indicating that during the first three quarters of the search it is less prone to duplicates than DMMAS.

Similarity ratio

The formulation of the similarity ratio used in this paper follows the definition given by Solnon and Fenet (2006) and its adaption provided in Ke et al. (2010). Hence, it also corresponds to the pair-wise population diversity measures suggested by Sidaner et al. (2002) and Morrison and De Jong (2002). Let x_j^k be the manifestation of the binary variable for the j^{th} object from the solution of the k^{th} ant, and n^a the number of ants, then

$$Similarity\ Ratio = \frac{\sum_{j=1}^{n}\left(\sum_{k=1}^{n^a} x_j^k \cdot \left(\sum_{k=1}^{n^a} x_j^k - 1\right)\right)}{(n^a - 1) \cdot \sum_{j=1}^{n}\sum_{k=1}^{n^a} x_j^k} \tag{3.16}$$

If the solutions of all ants are identical, this ratio is equal to one. If all ants select completely different objects, it is equal to zero.

Figure 3.4 depicts the development of the similarity ratio for the instances 5.100-00 and 10.100-00. For NormANTS, the ratio rises for both problems steadily from around 0.55 to 0.8. In the first half of the search it is considerably lower than for MMAS and DMMAS which both start with values around 0.72. While in the long run, DMMAS and NormANTS reach similar final values between 0.75 and 0.8, the similarity ratio of MMAS asymptotically approaches one after three quarters of the iterations. Generally, the results for the similarity ratio support the findings based on the analysis of the resampling ratio, i.e. that NormANTS with the novel pheromone model is able to effectively search the solution space, especially in comparison to the MMAS with the traditional pheromone concept described in section 3.2.

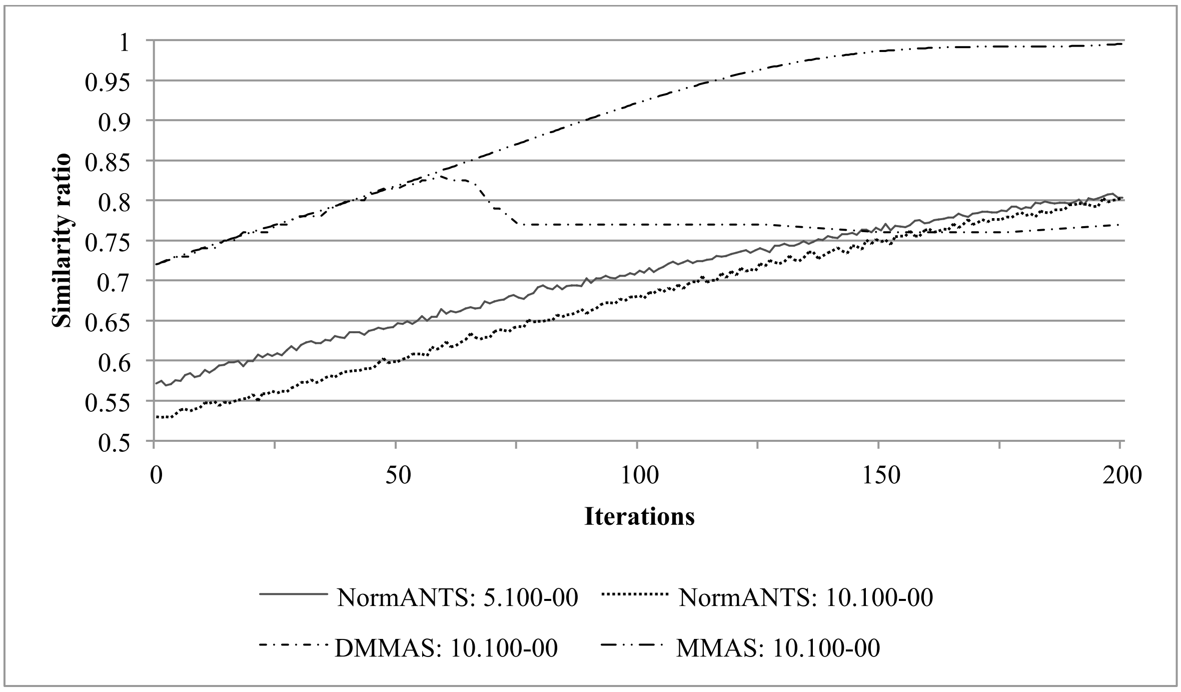

Figure 3.4: Evolution of the similarity ratio for 5.100-00 and 10.00-00 over 200 iterations (average over ten runs)

Note that a good balance of the trade-off between diversification and intensification is exactly what also Ke et al. (2010) aim for. While NormANTS uses normalized values and a coordinated pheromone delta to keep the relative differences within the pheromone map under control, DMMAS dynamically adjusts the lower pheromone limit in order to achieve just the same.

3.5.4 Results

The first test set contains instances from Weingartner and Ness (1967) and Shih (1979). The size of these problems ranges from two constraints and 28 to 105 objects (weing01-08) to five constraints and 30 objects (weish01-05). The results are given in Table 3.1. If available, the best solution, the average objective function value as well as the standard deviation are reported. For all problems of the first set, NormANTS finds the best known solution consistently. Only for weing07, one of the two largest instances, the standard deviation is larger than zero and the average solution quality is unequal to the best known solution. Overall, NormANTS outperforms the algorithm of Leguizamón and Michalewicz (1999) and matches the results of Boryczka (2007).

Table 3.1: Results of NormANTS for instances from Weingartner & Ness (1967) and Shih (1979)

Instance		L. & M. (1999)		Boryczka (2007)	NormANTS		
	Best known	Best	Avg.	Best	Best	Avg.	Std.Dev.
Weing01	141278	**141278**	141078	**141278**	**141278**	**141278**	0.0
Weing02	130883	**130883**	**130883**	**130883**	**130883**	**130883**	0.0
Weing03	95677	**95677**	95667	**95677**	**95677**	**95677**	0.0
Weing04	119337	**119337**	**119337**	**119337**	**119337**	**119337**	0.0
Weing05	98796	**98796**	**98796**	**98796**	**98796**	**98796**	0.0
Weing06	130623	**130623**	**130623**	**130623**	**130623**	**130623**	0.0
Weing07	1095445	1095382	1095382	**1095445**	**1095445**	**1095420**	34.5
Weing08	624319	**624319**	**624319**	**624319**	**624319**	**624319**	0.0
Weish01	4554	**4554**	**4554**	**4554**	**4554**	**4554**	0.0
Weish02	4536	n/a	n/a	**4536**	**4536**	4536	0.0
Weish03	4115	n/a	n/a	**4115**	**4115**	4115	0.0
Weish04	4561	n/a	n/a	**4561**	**4561**	4561	0.0
Weish05	4514	n/a	n/a	**4514**	**4514**	4514	0.0

Best found and average objective function values as well as standard deviation of ten runs per instance.
The best results are in boldface.
n/a: not available

The second test bed encompasses problems taken from Chu and Beasley (1998). All instances of this set comprise 100 objects and five constraints. The results are summarized in Table 3.2.

Table 3.2: Results of NormANTS for instances from Chu & Beasley (1998) with five constraints

Instance	Best known	L. & M. (1999)		Alaya et al. (2004)			Ke et al. (2010)			NormANTS		
		Best	Avg.	Best	Avg.	Std.Dev.	Best	Avg.	Std.Dev.	Best	Avg.	Std.Dev.
5.100-00	24381	**24381**	24331	**24381**	24342	29.3	**24381**	24362	23.8	**24381**	**24381**	0.0
5.100-01	24274	**24274**	24245	**24274**	24247	38.5	**24274**	24273	6.2	**24274**	**24274**	0.0
5.100-02	23551	**23551**	23527	**23551**	23529	8	**23551**	23540	7.2	**23551**	**23542**	7.5
5.100-03	23534	23527	23463	**23534**	23462	32.6	**23534**	23482	14.9	**23534**	**23512**	16.8
5.100-04	23991	**23991**	23949	**23991**	23946	31.8	**23991**	23954	10.8	**23991**	**23972**	16.8
5.100-05	24613	**24613**	24563	**24613**	24587	31.3	**24613**	**24608**	6.3	**24613**	**24608**	6.6
5.100-06	25591	**25591**	25504	**25591**	25512	43.8	**25591**	**25591**	0	**25591**	**25591**	0.0
5.100-07	23410	**23410**	23361	**23410**	23371	30.3	**23410**	23404	13.3	**23410**	**23410**	0.0
5.100-08	24216	24204	24173	**24216**	24172	32.9	**24216**	**24211**	5.9	**24216**	**24211**	6.6
5.100-09	24411	**24411**	24326	**24411**	24356	44.3	**24411**	24406	13.8	**24411**	**24411**	0.0

Best found and average objective function values as well as standard deviation of ten runs per instance.
The best results are in boldface.

Again, NormANTS is able to identify the best known solution for all instances. In terms of average solution quality, NormANTS considerably outperforms Alaya et al. (2004). For 5.100-05, 5.100-06, and 5.100-08, NormANTS and DMMAS achieve the same averages. Still, for seven out of ten instances NormANTS outperforms DMMAS in this regard and obtains the best average solution quality of all algorithms.

Table 3.3 displays the results for large sized problems also taken from Chu and Beasley (1998). Each instance of the third set consists of ten constraints and 100 objects. For nine out of ten instances, the best known solution was found. In this regard only the algorithm of Ke et al. (2010) performed equally well. Interestingly, for problem 10.100-05 NormANTS obtained a higher average solution quality than DMMAS even though the latter was able to identify the optimum while NormANTS was not.

Table 3.3: Results of NormANTS for instances from Chu & Beasley (1998) with ten constraints

Instance	Best known	L. & M. (1999)		Alaya et al. (2004)			Ke et al. (2010)			NormANTS		
		Best	Avg.	Best	Avg.	Std.Dev.	Best	Avg.	Std.Dev.	Best	Avg.	Std.Dev.
10.100-00	23064	23057	22996	**23064**	23016	42.2	**23064**	23045	19.6	**23064**	**23055**	5.8
10.100-01	22801	**22801**	22672	**22801**	22714	67.2	**22801**	22742	40.8	**22801**	**22753**	47.0
10.100-02	22131	**22131**	21980	**22131**	22034	66.9	**22131**	22091	29.8	**22131**	**22110**	30.7
10.100-03	22772	22772	22631	22717	22634	60.6	**22772**	**22710**	37.6	**22772**	22689	50.3
10.100-04	22751	22654	22578	22654	22547	66.3	**22751**	22617	43.9	**22751**	**22643**	45.5
10.100-05	22777	22652	22565	22716	22602	63.3	**22777**	22663	40.3	22716	**22666**	36.5
10.100-06	21875	**21875**	21758	**21875**	21777	44.9	**21875**	21826	28.4	**21875**	**21833**	46.4
10.100-07	22635	22551	22519	22551	22453	89.2	**22635**	22557	31	**22635**	**22567**	36.0
10.100-08	22511	22418	22292	**22511**	22351	69.4	22438	22409	17.3	**22511**	**22431**	26.3
10.100-09	22702	**22702**	22588	**22702**	22591	88.5	**22702**	22696	32.7	**22702**	**22702**	0.0

Best found and average objective function value as well as standard deviation of ten runs per instance.
The best results are in boldface.

Overall, NormANTS achieves the best average solution quality for nine out of ten instances and outperforms DMMAS in this regard. Compared with Leguizamón and Michalewicz (1999) and Alaya et al. (2004), NormANTS consistently obtains better results regarding both, the best solution found as well as the solution average.

3.6 Conclusion

In this paper, a novel pheromone concept for ACO is introduced. The two constituent characteristics of this pheromone model are (i) the normalization of pheromone values between zero and one and (ii) the coupling of the iterative pheromone delta with the total number of iterations. These properties ensure a high degree of control of the iterative modification of pheromone trails. The balance of pheromone deposit and removal leads to a coordinated evolution of the pheromone map so that after the chosen number of iterations an adequate differentiation between good and bad objects is achieved. Accordingly, diversification and intensification are well balanced throughout the search of the artificial ants.

In order to analyse the performance of the proposed algorithm, an experimental study was conducted. Without any further problem specific adjustments, NormANTS was applied to the multidimensional knapsack problem and tested on 33 different instances taken from the literature. The solutions were compared with various benchmarks, namely the ant algorithms of Leguizamón and Michalewicz (1999), Alaya et al. (2004), Boryczka (2007), and Ke et al. (2010). The results indicate that the pheromone model with normalized pheromone trails and an iteration-dependent pheromone delta allows for an effective search behaviour. For 32 out of 33 instances, the optimum was found. Moreover, the average solution quality was consistently close to the best known solution. Interestingly, this was also the case for the single instance that was not solved to optimality. Overall, NormANTS clearly outperformed the algorithms of Leguizamón and Michalewicz (1999) and Alaya et al. (2004) in terms of average solution quality and best solution found. A comparison with the results of Ke et al. (2010) revealed that in terms of average solution quality NormANTS performed better.

An analysis of the similarity and resampling ratio further supported the high search capability. The effectiveness in exploring the search space can be considered the major reason for the good results in the experimental study. Note that in this regard, NormANTS and DMMAS exhibit similar behaviour with comparable similarity and resampling ratios, which explains why both algorithms obtain such good results.

Furthermore, the results imply that not necessarily the choice of the right type of pheromone trails (in this case vertex-oriented or pair-wise) determines the search capabilities. Mechanisms to control the evolution of these trails in the course of the search seem to be at least as important.

Although NormANTS is applied to the multidimensional knapsack problem in this study, the suggested pheromone model is of generic nature and not problem specific. Hence, it can be easily transferred to other optimization problems. Accordingly, the application of the algorithm presented in this paper (or its key properties, respectively) to other optimization problems is an interesting topic for further research.

4

ANT COLONY OPTIMIZATION FOR THE MULTI-PERIOD MIXED-MODEL ASSEMBLY LINE SEQUENCING PROBLEM

Michael Hamann, Jörn-Henrik Thun, and Andrej Saweljew

Abstract

This paper introduces a model for a multi-period variant of the assembly line sequencing problem, expanding the objective of the original mixed-model sequencing problem by a labour dimension. The production sequence has to be chosen such that work overload does not occur and the total labour requirement over multiple work shifts is minimized. An ant algorithm with a novel pheromone concept based on normalized pheromone values and a directly controlled pheromone map evolution is presented and tested on 48 different randomly generated problem instances of various sizes. Optimal solutions generated with Gurobi as well as the solutions of probabilistic and deterministic greedy algorithms serve as benchmarks. The results of the computational study show that the ant colony optimization approach provides optimal or near-optimal solutions within short computation times and outperforms the other algorithms.

4.1 Introduction

Today, automotive manufacturers face a demanding and highly competitive market environment, in which it is indispensable to satisfy individual customer demands in order to gain sustainable competitive advantage (Holweg and Pil, 2001; Pil and Holweg, 2004). Accordingly, they have to offer a great variety of different models (Holweg and Greenwood, 2001; Röder and Tibken, 2006), which implies the production of many variants of the same base product on the same assembly line. One of the most important models for sequencing products on such a mixed model assembly line is the mixed-model sequencing model (MMS) (Wester and Kilbridge, 1963). It aims at finding a production sequence that meets the overall demand and at the same time minimizes the total amount of work overload. Work overload occurs when the processing of a model cannot be finished within the boundaries of the respective workstation. Various characteristics of the assembly line such as cycle time, processing times and station borders are explicitly considered.

In this paper, we present a multi-period variant of such a mixed-model sequencing problem (MPMMS), which focuses on the minimization of the overall labour requirement over multiple working shifts. As for the MMS, a solution of the MPMMS consists of a sequence of different models that are to be processed by workstations along the assembly line. The production sequence has to be chosen so that work overload does not occur and the total labour requirement is minimal. This requires solution procedures to provide some flexibility and problem-specific intelligence, since we need to smoothen labour requirements within shifts while we need to allow for a different staffing of workers between shifts in order to realize minimal overall labour requirements. Furthermore, the multi-period character of the model increases the complexity of the problem, as every time a solution is evaluated in terms of feasibility and quality, multiple shifts have to be evaluated individually.

As traditional exact approaches such as branch-and-bound or branch-and-cut often require inadequately high computation times for $\mathcal{NP}$-hard problems such as the MMS, we use ant colony optimization to solve the MPMMS. Besides some problem-specific features regarding the use of local information during the search, we apply a novel pheromone update concept, which is based on normalized pheromone values and enables a direct control of the speed with which the pheromone map evolves. In addition to the ant algorithms, we propose two greedy algorithms, which utilize information about the similarity of models in terms of labour requirement for the solution construction. To evaluate the algorithms, we test them on 48 different MPMMS instances ranging from small to large size. Solutions generated with Gurobi serve as a benchmark for the results obtained by the heuristic approaches.

The remainder of this paper is organized as follows. In section 4.2, the relevant literature in the field of mixed-model assembly line sequencing is reviewed. In section 4.3, we introduce a model for the multi-period mixed-model sequencing problem. The characteristics of the proposed greedy and ant algorithms are summarized in section 4.4. In section 4.5, we present an extensive experimental study and report the computational results. In the last section, we conclude with a discussion of our results.

4.2 Literature review

There exists a massive body of literature on a broad variety of assembly line sequencing problems, assembly line balancing problems, and combinations of both. Hence, the literature review presented in this section is restricted to research with a high relevance for this study, i.e. a) research that focuses on the core problem of this paper, the mixed-model sequencing problem, and b) research that solves the MMS or closely related problems using ant colony optimization. For a more comprehensive overview, the reader is referred to Boysen et al. (2009), who present a hierarchical classification scheme for the three alternative modelling approaches for sequencing problems discussed in the literature, namely mixed-model sequencing, car sequencing and level scheduling.

Moradi and Zandieh (2013) apply a novel imperialist competitive algorithm to a just-in-time mixed-model sequencing problem where variations of production rates are to be minimized. To evaluate performance, the proposed algorithm is compared against a genetic algorithm. Boysen et al. (2011) investigate a variant of the mixed-model sequencing problem in which a utility worker takes over whenever it is foreseeable that work overload will occur in a production cycle. They present a binary linear program along with a complexity proof and test different exact as well as heuristic solution approaches. Work overload minimization criteria are also investigated by Cano-Belmán et al. (2010). In their study, they address a mixed-model assembly-line sequencing problem with a scatter search based hyper-heuristic. The results of their experimental study show the effectiveness of the proposed hyper-heuristic compared to existing heuristics. Bautista and Cano (2011) present a formulation for the mixed-model sequencing problem with workload minimization for production lines with serial workstations. To solve this variant of MMS, they suggest a procedure through bounded dynamic programming. Akgündüz and Tunalı (2010) propose an adaptive genetic algorithm to solve a mixed-model assembly line sequencing problem with multiple objectives such as variation in part consumption rates, total utility work and setup costs. The results of their computational study show that the adaptive GA-based approach outperforms the non-adaptive algorithm in terms of solution quantity and quality. Multiple objectives, i.e. the minimization of number of setups and variation of production rates, are also considered by Moradi et al. (2011), who combine a ranked-based roulette wheel selection algorithm with a pareto-based population ranking algorithm. The results of the hybrid algorithm are compared against solutions obtained via total enumeration in small problems and also against other search heuristics in small, medium and large problems.

Regarding the optimization technique, there are a couple of publications that apply ACO to the car sequencing problem, which is closely related to the MMS. Gottlieb et al. (2003) provide one of the first studies on ant colony optimization for the car sequencing problem. A car sequencing problem with three production stages, i.e. construction, painting, and assembly, is solved by Gagné et al. (2006) using ant colony optimization. Solnon (2008) introduces an ant colony optimization approach for solving the car sequencing problem with two different pheromone structures, i.e. one learning for "good" car sequences and one learning for "critical" car sequences. The paper shows that the combination of the two pheromones leads to promising results. Morin et al. (2009) propose a specialized pheromone trail structure, which is specifically adapted to the type of constraints in the car sequencing problem. Again, the ant colony optimization approach shows good results.

Only few publications presenting applications of ACO to mixed-model sequencing problems can be found. McMullen (2001) propose an ant algorithm for the production-sequencing problem with two objectives, namely minimization of setups and optimization of stability of material usage rates. The solutions obtained with ACO are compared against solutions generated with simulated annealing, tabu search, genetic algorithms and neural network approaches. The experimental results show that the ACO approach is competitive in terms of solution quality and CPU requirements. Zhu and Zhang (2011) consider a similar objective. They propose an ant algorithm with an elitist ant strategy for identifying optimal sequence of multi-product models, such that the deviation between the ideal material usage rate and the practical material usage rate is minimized.

4.3 The multi-period mixed-model sequencing problem

4.3.1 Problem description

In a just-in-time environment, the final production sequences for assembly lines are typically defined for a fixed planning horizon in order to enable an exact coordination of deliveries by suppliers. Such a fixed horizon typically ranges from one or two days up to two weeks (see, e.g., Ervolina et al., 2009). As a consequence, a production sequence incorporates multiple working shifts, e.g. ten shifts, if we assume a five-day planning horizon and two working shifts per day. Depending on the economic objectives of the sequencing process, it is – as in this paper – necessary to explicitly take this multi-period character into consideration.

According to Boysen et al. (2009), sequencing approaches in literature mainly focus on two basic objectives, namely minimizing work overload and levelling part usage. The extended mixed-model sequencing problem in this paper considers a third objective in addition to the work overload criterion: It aims at minimizing the labour requirements along the assembly line over multiple working shifts. Labour requirement here is defined as the required number of operators per work station that need to be staffed to process the scheduled production orders. This objective was directly motivated by a real-world assembly line of a large German manufacturing plant from the automotive industry. At the respective company, fixed cost determined by the workforce play a major role when it comes to minimizing production costs. Since the required number of operators depends on the specific models that are scheduled for production, an efficient assignment of operators to stations is one major objective of the detailed scheduling process.

It is important to note that the amount of labour (or its cost, respectively) is not determined by work overload as in the original MMS but by the staffing of personnel at the workstations during a work shift. Since every model has individual labour requirements on each station of the assembly line, the number of operators depends on the models that are processed during a shift. The differences regarding the labour requirements are driven by the overall cycle time or production tact of the assembly line. By parallelizing time-consuming tasks and assigning them to a higher number of operators, processing times are adjusted in order to match the given cycle time of the production system. The cycle time is not adjusted on a frequent basis and hence not subject to short-term planning but output of the simple assembly line balancing problem (SALBP) that focuses on minimizing the cycle time for a fixed number of workstations (or minimizing the number of work stations for a fixed cycle time) and typically precedes the MMS.

In each shift, the overall labour requirement per station is determined by the maximum labour requirement of the different orders that pass the station within a shift. For example, assume that ten models are processed at a station during a shift. If five of them require three operators while five require two, three operators need to be scheduled at the station for the whole shift to cover the labour requirements of the first five models. This example raises the question why such differences in terms of labour are not covered using utility workers or floaters, which are not assigned to a certain workstation but can flexibly move along the assembly line to support stations where labour-intensive models are processed.

The answer to this question is twofold. First, in practice, there are organizational limits to the use of such utility workers. The higher their number and the higher the number of stations, the higher will be the complexity of coordination and the risk of diversion and nervousness along the assembly line. Hence, in the case of several dozens up to several hundred stations, it does not seem convincing to argue that utility workers should be the only means to cover work overload situations. Second, the differences between models regarding their labour requirements are known a-priori, i.e. before the final production sequence is determined. It is reasonable to utilize this information in the short term planning process if possible.

Since we focus on labour as the major objective dimension, we formulate the minimization of work overload, which is the major objective of the original MMS, as an additional hard constraint. More precisely, we drop the assumption that work overload does not impact succeeding workstations and consider solutions that lead to violations of station boundaries as infeasible.

Note the characteristics of a "good" production sequence and the implication for the optimization algorithms: A good sequence smoothens the labour requirement within individual working shifts. The more alike the labour requirements of orders being processed within one shift, the better is typically the utilization of the operators and the lower will the overall labour amount be. Between shifts, however, the number of operators per station differs. As a consequence, the optimization algorithms need to be able to smoothen labour requirements within shifts while allowing for differences between shifts in order to minimize the overall labour demand. Furthermore, it is important to understand that labour differences between shifts do not imply that the total workforce strongly fluctuates over time. It is the assignment of operators to stations that is different in different shifts within the short term planning horizon but not necessarily the average size of the overall workforce.

In line with the explanations above and following the basic assumptions of the MMS as stated by Scholl and Klein (1999), Boysen et al. (2009), and Golle et al. (2014), the characteristics of the MPMMS can be summarized as follows:

- The assembly line consists of a given number of workstations of individual station length.
- The stations are closed, i.e. an operator can only process a model within his station boundaries and must not move beyond.

- Individual deterministic processing times (station times) are given for all models at every station.
- Assigning more personnel to a workstation cannot reduce processing times any further. It is assumed that this has already been done as far as possible during the process of setting the cycle time for the assembly line.
- The conveyor moves at constant speed from left to right.
- Fixed rate launching is applied, i.e. models are launched according to a fixed cycle time.
- The demand for each model is given and non-negative in the actual period. Rush orders are not allowed.
- The operators have zero return times, i.e. we neglect the time an operator needs to move from where he finished working on one model to the point where her starts working on the next model.
- Times and distances are standardized, i.e. one time unit is needed to cover one distance unit.
- Work overload is not allowed.
- The labour demand is determined by the maximum requirement of all orders that pass a station within a shift.

4.3.2 Model

To describe the problem, we use the following notations:

T	number of production cycle, i.e. sequence length (index t)
M	number of models (index m)
K	number of stations (index k)
S	number of work shifts (index s)
c	cycle time
d_m	demand for model m
l_k	length of station k
$\mathfrak{l}_k$	left border of station k
$\mathfrak{r}_k$	right border of station k
p_{mk}	processing time of model m in station k
θ_{kt}	start position of the t^{th} model in station k
o_{kt}	end position of the t^{th} model in station k

ι_s	start time of shift s
ζ_s	end time of shift s
ϕ_{kt}	start time of the t^{th} model in station k
ω_{kt}	end time of the t^{th} model in station k
$\mathcal{L}_{mk}$	labour requirement of model m in station k
x_{mt}	binary variable, 1 if m is produced in slot t, 0 otherwise
y_{mks}	binary variable, 1 if m is processed in station k during shift s, 0 otherwise

With the problem description, notation, and assumptions above, we get the following multi-period MMS model:

$$minimize \quad \pi = \sum_s \sum_k (max_{m\in M}(\mathcal{L}_{mk} \cdot y_{mks})) \tag{4.1}$$

$$subject\ to$$

$$\theta_{kt} = max\left((\theta_{k,t-1} + \textstyle\sum_{m\in M}(p_{mk} \cdot x_{m,t-1}) - c), (\mathfrak{l}_k)\right) \quad \forall\, k \in K; t = 2, \dots, T \tag{4.2}$$

$$o_{kt} = \ \theta_{kt} + \textstyle\sum_{m\in M} p_{mk} \cdot x_{mt} \quad \forall\, k = 1, \dots, K; t = 1, \dots, T \tag{4.3}$$

$$\theta_{kt} + \textstyle\sum_{m\in M} p_{mk} \cdot x_{mt} \ \le \mathfrak{r}_k \quad \forall\, k = 1, \dots, K; t = 1, \dots, T \tag{4.4}$$

$$\phi_{kt} = \mathfrak{l}_k \quad \forall\, k = 1, \dots, K; t = 1 \tag{4.5}$$

$$\phi_{kt} = max\{\omega_{k,t-1};\ c(t-1) + \mathfrak{l}_k\} \quad \forall\, k = 1, \dots, K; t = 2, \dots, T \tag{4.6}$$

$$\omega_{kt} = \ \phi_{kt} + \textstyle\sum_{m\in M} p_{mk} \cdot x_{mt} \quad \forall\, k = 1, \dots, K; t = 1, \dots, T \tag{4.7}$$

$$y_{mks} = \begin{cases} 1 & if\ \phi_{kt} \le \zeta_s \ \wedge \phi_{kt} \ge \iota_s) \vee (\omega_{kt} \le \zeta_s \ \wedge\ \omega_{kt} \ge \iota_s) \\ 0 & else \end{cases} \tag{4.8}$$
$$\forall\, m = 1, \dots, M; k = 1, \dots, K; s = 1, \dots, S;\ t = 1, \dots, T$$

$$x_{mt} \in \{0,1\} \quad \forall\, m \in M, t \in T \tag{4.9}$$

$$\textstyle\sum_{t\in T} x_{mt} = d_m \quad \forall\, m \in M \tag{4.10}$$

$$\sum_{m\in M} x_{mt} = 1 \quad \forall\, t \in T \tag{4.11}$$

$$\phi_{kt} \geq 0 \quad \forall\, k \in K; t = 2, \dots, T \tag{4.12}$$

$$\omega_{kt} \geq 0 \quad \forall\, k \in K; t \in T \tag{4.13}$$

The objective function given in equation (4.1) minimizes the total labour requirement over all shifts and stations. The labour requirement per station and shift depends on the models m that are processed in station k during shift s.

The starting position of the processing of the order in the t^{th} production slot in a station k is equal to the maximum of the left border of station k and the start position of the previous order plus the distance covered during processing minus the distance covered during the cycle time c (equation (4.2)). The end position of an order equals the starting position plus the distance covered during the processing time of this order in station k (equation (4.3)). Equation (4.4) reflects that the processing of orders has to be completed within the station borders. As mentioned before, while this is the objective of the original MMS, it is formulated as an additional hard constraint in this model.

For the first order in the production sequence, the starting time of processing equals the left station border at every station k (equation (4.5)). For the following orders the definition of starting time is considerably more complex as it depends on all previous orders, potential waiting times induced by station boundaries and the production cycle time (equation (4.6)). The end time is defined as the starting time plus the respective processing time at each station (equation (4.7)).

Equations (4.8) and (4.9) describe the binary variables of the model. y_{mks} indicates, if model m is processed in station k within shift s. x_{mt} indicates if model m is scheduled in production slot t. For each model, the demand needs to be fully satisfied (equation (4.10)) and in each production slot, exactly one model has to be scheduled (equation (4.11)). Moreover, starting and end times of orders are non-negative (equations (4.12) and (4.13)).

4.4 Solution approaches for the MPMMS

4.4.1 Greedy algorithms

Basically, any combinatorial optimization problem can be approached with more or less simple greedy algorithms, which iteratively add solution components according to some problem-specific heuristic rule until a complete solution is built. Typically, utility values of potential solution components are set in relation to their resource requirements in order to define priorities. Consider a traveling salesman problem (TSP), for example, where geographically distributed customers have to be served. Here, the utility value of a route between two customers usually is defined as the inverse of its length: the shorter the route, the higher is its utility or "heuristic value". Since a connection between two cities or customers is either short or long (at least in the general case), the calculation of heuristic values is basically straightforward. Naturally, good overall solutions for TSPs make use of shorter connections. In other words, by prioritizing shorter connections during solution construction, convergence towards favourable solutions can be both guided and speeded up.

However, different to a TSP, solution components of the multi-period MMS are not per se advantageous or not – they do not have a certain value or cost upfront. As a consequence, it is more challenging to assign utility values. Good solutions consist of a production sequence that results in a high utilization of the personnel assigned during the different working shifts, i.e. it is generally favourable to have models with rather similar labour requirements within one shift. Hence, we designed the general utility of an order so that it is determined by the previous orders of the production sequence that are processed during the same working shift: An order gets a low utility value if its labour requirements strongly deviate from the requirements of its predecessors and vice versa. By preferring orders with high utility values during the solution construction, it is ensured that partial sequences are generated such that the labour requirements of the respective orders are quite constant. This leads to a high utilization of the core personnel and a low overall labour demand over all shifts.

We designed two different greedy algorithms that both follow this logic in general but differ slightly regarding their definitions of "labour deviation". The first greedy logic considers only positive deviations, i.e. orders that have lower labour requirements than their predecessors at each work station still get the maximum heuristic value assigned. The second variant considers the overall deviation, i.e. both a labour increase as well as a decrease result in a lower utility value of the respective order.

- "Greedy logic I": Choose first the order that requires the lowest increase in term of labour over all workstations.
- "Greedy logic II": Choose first the order that has the smallest overall deviation in terms of labour requirement.

For greedy logic I and greedy logic II, both, a deterministic variant as well as an iterative probabilistic variant have been implemented. While the deterministic variants, *gre-1-det* and *gre-2-det*, always select the model with the highest heuristic value, the probabilistic algorithms, *gre-1* and *gre-2*, define transition probabilities according to the heuristic values and iteratively construct solutions based on these probabilities.

4.4.2 Ant colony optimization

The ant colony optimization metaheuristic has been inspired by the behaviour of foraging ants (Dorigo et al., 1996). Indirect communication between individual ants via chemical substances called pheromones enables ant colonies to identify the shortest paths between their nest and food sources (Deneubourg et al., 1990). The same principle is exploited in artificial ant colonies in order to solve combinatorial optimization problems (Dorigo and Stützle, 2004).

Since the early nineties, various ant algorithms have been developed and used to solve to different combinatorial optimization problems. Successful applications for extensively studied problems such as the traveling salesman problem (Cordón et al., 2000; López-Ibáñez et al., 2013), the vehicle routing problem (Bell and McMullen, 2004; Fuellerer et al., 2009), the knapsack problem (Boryczka, 2007; Leguizamón and Michalewicz, 1999), or the quadratic assignment problem (Gambardella et al., 1999; Stützle and Hoos, 2000), demonstrate that ACO is capable to deliver competitive results that match or are close to state-of-the-art algorithms for the respective problems.

Also for problems that are more closely related to the MMS such as, e.g., the car sequencing problem, ant algorithms have been applied with promising results (Morin et al., 2009; Solnon, 2008; Zhu and Zhang, 2011). Hence, ACO has been selected in this study to solve the multi-period MMS.

The ant algorithm presented in this paper is an advancement of the MAX-MIN Ant System as presented by Stützle and Hoos (2000). It uses the different types of greedy logic explained in section 4.1 as heuristic functions, which add some local problem-specific information during the search process to complement the artificial learning. Moreover, the algorithm combines different techniques, which aim at improving the overall performance in terms of solution quality and computation times. Besides proven methods such as candidate lists, it applies the novel pheromone model presented in chapter 3.

In the remainder of this section, the solution construction of the ant algorithm, the search procedure, and the pheromone concept are explained in detail.

Solution construction

Similar to other combinatorial optimization problems, the MPMMS can be depicted as a fully connected directed graph where each node i resembles an order that needs to be scheduled for production. A feasible solution is represented by a complete tour through the graph that doesn't violate any of the given restrictions. To generate such a solution, an ant k moves stepwise through the construction graph, i.e. it selects order after order from the set of remaining orders and schedules them for production. When all orders have been assigned to a slot in the production schedule (or if there is no order left satisfying the capacity constraint), the ant stops its search for the current iteration. The transition probability with which ant k chooses the next node j is given with

$$p_{ij} = \begin{cases} \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in candidates_k}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}} & if\ j \in candidates_k, \\ 0 & else, \end{cases} \tag{4.14}$$

with τ_{ij} being the pheromone intensity of the trail, i.e. the learnt desirability to process order j after order i, and η_{ij} being the heuristic value, i.e. the local attractiveness of scheduling order j after order i given by a heuristic function. As mentioned above, we implemented two types of greedy logic as heuristic functions to generate local information about the similarity of solution components.

Moreover, we applied candidate short lists as described by Gambardella and Dorigo (1996) and Dorigo and Stützle (2003). Generally, each individual solution component is assigned a candidate list based on the local attractiveness of all other (feasible) solution components. Only the most attractive components are included in the candidate list for the current solution component and are made available for selection in the next step of the solution construction process. In this case this means that during solution construction an ant k does not need to choose from all remaining feasible orders that have not been assigned to a slot in the production sequence yet, but only considers a subset containing the orders with the highest heuristic values, namely $candidates_k$.

Focusing on a reduced number of potentially desirable solution components during each step of the solution construction has both positive and negative implications. On the one hand, the downside of this technique lies in the risk of information loss during the search procedure. It is possible that solution components are temporarily neglected, as they don't seem desirable from a local point of view, although they might be required from a global perspective in order to achieve the global optimum. On the other hand, however, using a limited number of candidates during the solution construction can drastically reduce computation times. Regarding the complexity of MPMMS the gain in solution speed outweighs the potential loss of solution quality.

Pheromone update

As in the original MAX-MIN Ant System we set a minimum and a maximum boundary for the pheromone values, τ_{min} and τ_{max}, in order to ensure exploitation of the solution space at any time and prevent the artificial ant colony from entering a stagnation situation. Furthermore, an elitist strategy similar to the suggestions of Dorigo et al. (1996) and Bullnheimer et al. (1997) is used for the pheromone update. The iteration best ant, k^{+iter}, as well as the best ant so far, k^{+all}, are allowed to modify the pheromone trail on their solution components. In order to favor exploration, not only a positive update is conducted as in the majority of ant algorithms. A negative update, i.e. pheromone subtraction, as suggested by Cordón et al. (2000) and Montgomery and Randall (2002) is additionally carried out by the worst ant of the iteration, k^{-iter}.

Note that a purely positive update by only the best ant(s) as it is commonly used in applications of ACO would work as well in this case. However, considering the computational complexity and the potential size of real-world instances of the MPMMS with high numbers of work stations and orders, enhancing convergence by adding information about inferior solutions during the search seems desirable.

Together with the pheromone evaporation, the trails are updated at the end of each iteration with

$$\tau_{ij} = (1-\rho)\,\tau_{ij} + \Delta\tau_{ij}^{k^{+iter}} + \Delta\tau_{ij}^{k^{+all}} - \Delta\tau_{ij}^{k^{-iter}} \tag{4.15}$$

Note that since both, the iteration-best as well as the globally best ant so far, carry out a pheromone update, the positive update outweighs the negative update at any time. As partial sequences are often simultaneously part of both, the best and the worst solution found by the ants, this is essential in order to prevent a neutralization of positive and negative update, which would negatively influence the evolution of differentiated pheromone trails.

Normalizing pheromone and heuristic values and controlling search speed

Many ant algorithms compute the increase of the pheromone trails as a function of the quality of the solution of the respective ant that modifies the trails (Alaykỳran et al., 2007; Boryczka, 2007; Stützle and Hoos, 2000) or as a function of the number of ants that generated the respective solution (Seçkiner et al., 2013). In other words, either the absolute value of the respective solution or the temporary strength of the convergence of the artificial colony influence the amount of pheromone deposited on the trails.

By linking the iterative pheromone delta of some ant k to the solution quality with, e.g., $\Delta\tau_{ij}^{k} = 1/ObjectiveFunctionValue$ (if (i,j) is used by ant k), the search behaviour of the algorithm directly depends on the numerical values of the problem instance and temporary solutions found. As a consequence, even with the same parameter settings (number of iterations and ants, relative importance of pheromone and heuristic values, etc.) the artificial ants will behave differently for problem instances with relatively small differences between very good and very bad solutions (in terms of objective function value) and problem instances with relatively large differences.

In this paper, an approach is used which allows for a more direct control of the speed with which the pheromone map evolves and, hence, a more direct control of the search behaviour. First, heuristic values and pheromone values are normalized between zero and one. Using such normalized values has several advantages. High absolute objective function values do not lead to a bias when the transition probabilities are calculated during solution construction, i.e. the potential risk of negative search bias towards non-optimal areas of the solution space because of good temporary solutions is mitigated. Furthermore, normalization provides the basis for a direct control of the pheromone delta by which the pheromone trails are updated iteratively during the search process, i.e. the speed with which the artificial ants explore the solution space and converge to certain regions is independent from individual characteristics of problem instances.

Second, the evaporation rate ρ (and as a consequence an ant's pheromone delta) is defined as a function of the total number of iterations of the search procedure with

$$\rho = 1/\,total\ number\ of\ iterations \tag{4.16}$$

Especially when applying different types of pheromone modification such as increase, decrease and evaporation simultaneously, it is important to carefully regulate their interaction so that they form a balanced system that iteratively leads to a differentiated evaluation of solution components. Ideally, the overall pheromone delta is designed so that the pheromone trails develop reasonably within the chosen number of iterations, i.e. the trails on desirable solution components slowly grow stronger while the trails on undesirable solution components slowly vanish. In order to achieve this, the number of iterations and the modification of the pheromone trails are directly linked. The normalized pheromone deltas for the iteration-best and -worst ant, k^{+iter} and k^{-iter}, and the globally best ant so far, k^{+all}, are defined as

$$\Delta\tau_{ij}^{k^{+iter}} = \rho\left(\pi^{k^{+iter}} - \pi^{k^{-all}}\right)/\left(\pi^{k^{+all}} - \pi^{k^{-all}}\right) \tag{4.17}$$

$$\Delta\tau_{ij}^{k^{-iter}} = \rho\left(\pi^{k^{-iter}} - \pi^{k^{-all}}\right)/\left(\pi^{k^{+all}} - \pi^{k^{-all}}\right) \tag{4.18}$$

$$\Delta\tau_{ij}^{k^{+all}} = \rho\left(\pi^{k^{+all}} - \pi^{k^{-all}}\right)/\left(\pi^{k^{+all}} - \pi^{k^{-all}}\right) \tag{4.19}$$

Combining normalized pheromone and heuristic values with the iteration-dependent pheromone delta results in a controlled evolution of the pheromone map within the given number of iterations (or computation time, respectively). In other words, without changing the general parameterization of the algorithm, learning rate and search speed vary depending on the number of iterations that is chosen based on problem size or complexity. For a smaller (larger) number of iterations, the pheromone trails will develop quicker (slower), i.e. the ants will automatically converge faster (slower) towards certain solutions.

Figure 4.1 exemplarily shows the development of the pheromone trails for different settings regarding the number of iterations (the test runs have been conducted for a random MPMMS instance with 15 orders and ten stations).

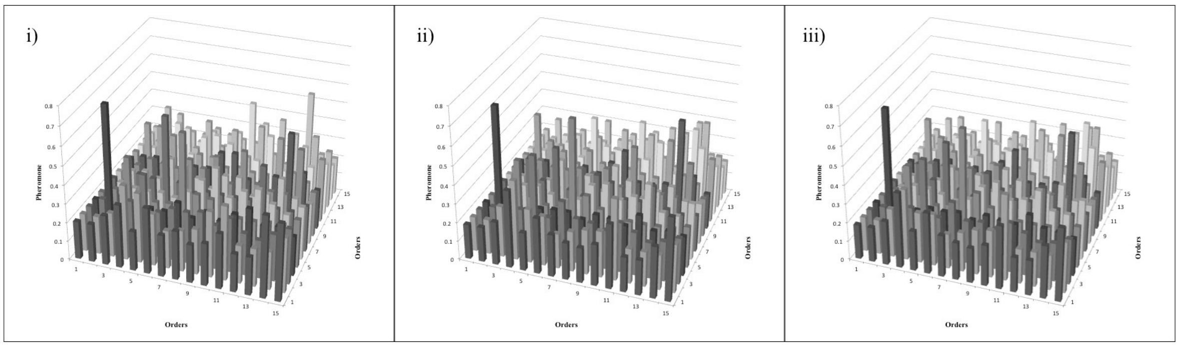

Figure 4.1: Evolution of pheromone map for 100 iterations (i)), 1000 iterations (ii)), and 10.000 iterations (iii))

The orders that need to be scheduled are depicted on the horizontal axes, while the absolute strength of the pheromone trails is shown on the vertical axes. Despite the different numbers of iterations, desirable combinations of orders (that are in fact part of the optimal solution) are clearly marked in all three cases. Overall the pheromone maps show the differentiated structure that is wanted.

4.5 Computational study

4.5.1 Experimental design

In the experimental study, three different versions of the ant algorithm as well as two stochastic and two deterministic greedy algorithms are tested on overall 48 different problem instances. The algorithms *ant-1* and *ant-2* use the greedy logic I and II as heuristic functions. The heuristic *learning only (lo)* works with pheromone-based learning only and doesn't consider any heuristic values. It serves as a benchmark to evaluate the impact and importance of local information when solving the MPMMS with ACO. Besides the ant algorithms, the greedy algorithms described in section 4.4, *gre-1* and *gre-2* as well as *gre-1-det* and *gre-2-det*, are tested. Table 4.1 provides an overview of the heuristic approaches.

Table 4.1: Overview of heuristic algorithms

Algorithm	**Description**
lo	ACO without heuristic function
ant-1	ACO with heuristic function following greedy logic I
ant-2	ACO with heuristic function following greedy logic II
gre-1	Probabilistic greedy algorithm following greedy logic I
gre-2	Probabilistic greedy algorithm following greedy logic II
gre-1-det	Deterministic greedy algorithm following greedy logic I
gre-2-det	Deterministic greedy algorithm following greedy logic II

In order to evaluate the performance of the different algorithms, we generated random test instances adapting the principles of the general instance generator of Scholl and Klein (1999) and the suggestions of Golle et al. (2014). This approach is also used by Cano-Belmán et al. (2010) and Emde et al. (2010) and is well suited for the generation of multi-period instances of MMS in this paper. It provides a general framework for generating problem instances that have the same structure and are similar to a certain extent in a realistic manner but still differ in terms of parameter values that are chosen randomly based on the following limits and distributions.

The length of the workstations is randomly distributed in the interval $[100;\ 140]$, and, for all instances, the cycle time is set to $c = 90$. Furthermore, each station has two possible processing times, a long one and a short one, namely $p_k^- < c$ and $p_k^+ > c$, that are randomly chosen from the intervals $[0.75c, c-1]$ and $[c+1, min(1.15c,\ l_k)]$, respectively. The processing times per model and station, p_{mk}, are then randomly chosen from $\{p_k^-; p_k^+\}$ so that each model is unique and the average processing time for each station k over all models and for each model m over all stations are not larger than c. The labor requirement per station and model, lab_{mk}, is randomly chosen from the intervals $[1; 3]$ and $[2; 4]$. Overall, each randomly generated model resembles one production order that needs to be scheduled and fulfilled.

The number of orders as well as the number of workstations have been varied in order to generate problem sets that differ in size and complexity. An overview is given in Table 4.2.

All heuristic algorithms have been implemented in MatLab R2012a; the mixed integer program has been modelled in GAMS 23.73 with Gurobi 4.5.1. The test runs have been conducted on a Windows 7 Professional computer with an Intel Core i7 processor with six cores (3.20 GHz each), twelve threads, and 32 GB RAM in total.

Table 4.2: Overview of problem sets

Problem Set		Orders	Stations	Instances	Problem Set		Orders	Stations	Instances
1	1_1	10	10	1_1_1	5	5_1	40	20	5_1_!
				1_1_2					5_1_2
				1_1_3					5_1_3
	1_2		15	1_2_1		5_2		25	5_2_1
				1_2_2					5_2_2
				1_2_3					5_2_3
2	2_1	15	10	2_1_1	6	6_1	50	20	6_1_1
				2_1_2					6_1_2
				2_1_3					6_1_3
	2_2		15	2_2_1		6_2		25	6_2_1
				2_2_2					6_2_2
				2_2_3					6_2_3
3	3_1	20	10	3_1_1	7	7_1	75	30	7_1_1
				3_1_2					7_1_2
				3_1_3					7_1_3
	3_2		15	3_2_1		7_2		40	7_2_1
				3_2_2					7_2_2
				3_2_3					7_2_3
4	4_1	30	20	4_1_1	8	8_1	100	30	8_1_1
				4_1_2					8_1_2
				4_1_3					8_1_3
	4_2		25	4_2_1		8_2		40	8_2_1
				4_2_2					8_2_2
				4_2_3					8_2_3

4.5.2 Parameter settings for ACO

During pre-tests, we examined the learning and convergence behaviour of the ant algorithm for different parameter settings. We varied α and β, the number of iterations and ants, the number of candidates, and the evaporation rate. Of all tested settings, the one described in the following performed best, and has thus been used for the experimental study.

For the smaller problem sets one to three, the number of iterations and ants were set to 500 and 50 for all ant algorithms and probabilistic greedy algorithms. For the larger problem sets four to eight, these numbers were increased to 1000 iterations and 100 ants. For all runs, we set $\alpha = 2$ and $\beta = 3$ in order to enable a more explorative search in the beginning and a more exploitative search towards the end of an optimization run. The pheromone values were initialized with $\tau^{ini} = 0.5$; the limits were set to $\tau^{min} = 0.1$ and $\tau^{max} = 1$. The pheromone evaporation was set to $\rho = 1/total\ number\ of\ iterations$. As explained above, this resulted in an adequate pheromone trail development within the chosen number of iterations. The number of candidates was set to 20 (or the maximum number of orders in case the problem instance consisted of less than 20 orders). This number is higher than for example reported by Gambardella and Dorigo (1996), since we found already for smaller problems that the solution quality decreased for lower numbers of candidates.

For this parameterization, the artificial ants iteratively improve solution quality and converge after a reasonable number of iterations as intended. Figure 4.2 exemplarily shows the solution quality development of the algorithm *ant-2* for the test problem *1_1_1* over 500 iterations. The graph shows the best solution found per iteration.

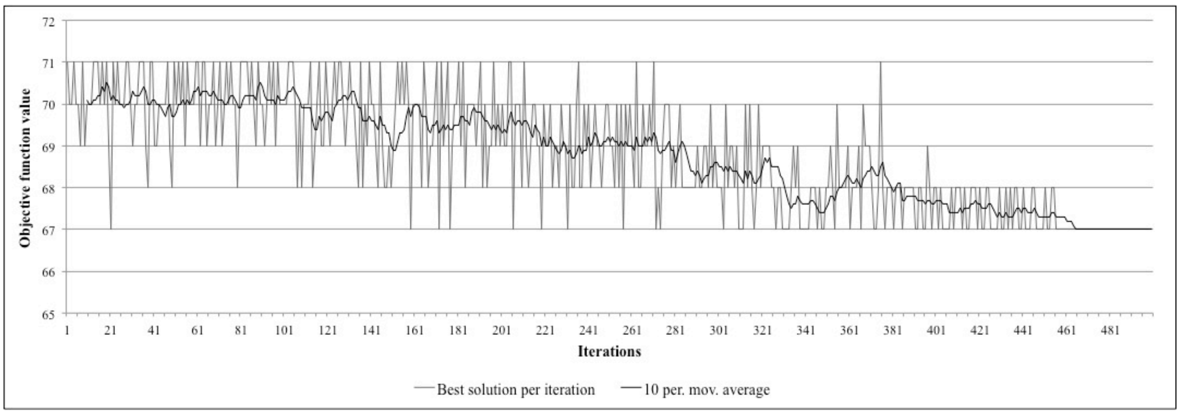

Figure 4.2: Solution quality development and convergence of ant-2 for problem 1_1_1

It is clearly visible that already in the early explorative phase of the search, good solutions and even the global optimum (with an objective function value of 67) are identified (due to the fact that *1_1_1* is quite a small problem with ten orders and ten stations only). Still, the colony doesn't "get stuck" on these early solutions; there is no indication for stagnation or premature convergence. Moreover, the graph reflects the explorative component of the search: In the different phases of the optimization run, the solution quality oscillates in certain ranges, while overall it continuously improves until the ants finally converge to the global optimum.

4.5.3 Results

The results for the smaller problem sets one to three are presented in Table 4.3. For all these problems, the probabilistic greedy and especially the ant algorithms achieved very good results. With *ant-1* and *ant-2,* the global optimum was identified for nine out of 18 instances. For the remaining nine instances, ACO still found the best solution of all heuristics. Moreover, including the already relatively complex instances from problem set three, the deviation from the optima was only 1.3% on average. For *gre-1* and *gre-2*, the average gap was considerably larger with 2.1%.

Table 4.3: Results for small problem sets 1 to 3

	1_1_1			**1_1_2**			**1_1_3**			**1_2_1**			**1_2_2**			**1_2_3**		
	Best	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**
Opt	67	-	2.0	71	-	< 1	79	-	1.0	111	-	4.0	121	-	7.0	117	-	8.0
lo	**67***	0.0%	24.7	**71***	0.0%	25.0	**79***	0.0%	25.0	**111***	0.0%	25.8	**121***	0.0%	25.9	**117***	0.0%	26.1
ant-1	**67***	0.0%	24.7	**71***	0.0%	25.0	**79***	0.0%	25.0	**111***	0.0%	25.9	**121***	0.0%	25.9	**117***	0.0%	26.0
ant-2	**67***	0.0%	24.6	**71***	0.0%	25.0	**79***	0.0%	25.0	**111***	0.0%	25.8	**121***	0.0%	25.9	**117***	0.0%	25.9
gre-1	**67***	0.0%	24.3	**71***	0.0%	24.7	**79***	0.0%	24.7	**111***	0.0%	25.6	**121***	0.0%	25.7	**117***	0.0%	25.6
gre-2	**67***	0.0%	24.3	**71***	0.0%	24.7	**79***	0.0%	24.7	**111***	0.0%	25.5	**121***	0.0%	25.7	**117***	0.0%	25.6
gre-1-det	71	6.0%	< 1	77	8.5%	< 1	91	15.2%	< 1	120	8.1%	< 1	128	5.8%	< 1	124	6.0%	< 1
gre-2-det	73	9.0%	< 1	81	14.1%	< 1	88	11.4%	< 1	123	10.8%	< 1	132	9.1%	< 1	125	6.8%	< 1
	2_1_1			**2_1_2**			**2_1_3**			**2_2_1**			**2_2_2**			**2_2_3**		
	Best	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**
Opt	75	-	18.0	80	-	43.0	73	-	27.0	126	-	93.0	118	-	321.0	121	-	538.0
lo	**75***	0.0%	42.3	82	2.5%	42.4	76	4.1%	42.6	129	2.4%	44.1	120	1.7%	44.1	**123**	1.7%	44.4
ant-1	**75***	0.0%	42.5	82	2.5%	42.4	75	2.7%	42.5	**126***	0.0%	44.2	**119**	0.8%	44.2	**123**	1.7%	44.2
ant-2	**75***	0.0%	42.5	**81**	1.3%	42.4	**73***	0.0%	42.5	**126***	0.0%	44.2	**119**	0.8%	44.2	**123**	1.7%	44.4
gre-1	**75***	0.0%	41.7	82	2.5%	41.8	76	4.1%	41.9	**126***	0.0%	43.5	120	1.7%	43.5	124	2.5%	43.7
gre-2	**75***	0.0%	41.5	**81**	1.3%	41.8	76	4.1%	41.8	127	0.8%	43.5	120	1.7%	43.5	**123**	1.7%	43.8
gre-1-det	80	6.7%	< 1	88	10.0%	< 1	79	8.2%	< 1	132	4.8%	< 1	129	9.3%	< 1	nfsf	-	< 1
gre-2-det	92	22.7%	< 1	87	8.7%	< 1	78	6.8%	< 1	131	4.0%	< 1	123	4.2%	< 1	131	8.3%	< 1
	3_1_1			**3_1_2**			**3_1_3**			**3_2_1**			**3_2_2**			**3_2_3**		
	Best	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**
Opt	99	-	60093.3	101	-	13838.5	90	-	2919.9	151	-	23978.0	153	-	3 d	150	-	3 d
lo	106	7.1%	65.1	107	5.9%	62.5	97	7.8%	64.9	157	4.0%	66.6	160	4.6%	65.7	159	6.0%	66.6
ant-1	**100**	1.0%	65.0	**104**	3.0%	63.5	**95**	5.6%	65.1	153	1.3%	66.7	**158**	3.3%	65.8	**155**	3.3%	66.6
ant-2	**100**	1.0%	65.1	105	4.0%	62.5	**95**	5.6%	64.9	**152**	0.7%	66.8	**158**	3.3%	66.0	**155**	3.3%	66.5
gre-1	104	5.1%	63.7	106	5.0%	61.6	**95**	5.6%	63.8	155	2.6%	65.4	160	4.6%	64.6	157	4.7%	65.5
gre-2	103	4.0%	63.9	106	5.0%	59.7	**95**	5.6%	63.6	156	3.3%	65.7	160	4.6%	64.8	156	4.0%	65.4
gre-1-det	106	7.1%	< 1	nfsf	-	< 1	97	7.8%	< 1	160	6.0%	< 1	165	7.8%	< 1	162	8.0%	< 1
gre-2-det	110	11.1%	< 1	nfsf	-	< 1	101	12.2%	< 1	171	13.2%	< 1	166	8.5%	< 1	164	9.3%	< 1

Best found results and average CPU time in seconds of three runs per instance.
* indicates that the global optimum is found.
nfsf = no feasible solution found.

For the medium and large sized problems (see Tables 4.4 and 4.5, respectively), ACO outperformed all other heuristic algorithms. For all instances of problem sets four to eight, either *ant-1* or *ant-2* generated the best solution found so far. For ACO, the average gap from the best known solution was 0.6% for the medium sized problems and 0.5% for the large sized problems. The average deviation of the greedy algorithms was again larger with 1.3% and 1.0%, respectively.

The performance of the deterministic greedy algorithms was not competitive. There are a few exceptions (due to the randomly generated problem instances) where good results were obtained. However, for many of the larger instances and even some of the small test problems, no feasible solutions have been found. The deterministic character of these algorithms seems to be unsuitable for the combination of work overload constraint and labour minimization objective and insufficient for the complexity of the MPMMS.

Table 4.4: Results for medium problem sets 4 to 6

	4_1_1			4_1_2			4_1_3			4_2_1			4_2_2			4_2_3		
	Best	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**
Opt	(215)	-	3 d	-	-	-	-	-	-	(290)	-	3 d	-	-	-	-	-	-
lo	229	6.5%	433.3	238	3.0%	424.2	224	2.3%	433.3	300	3.4%	432.6	283	1.8%	448.2	277	2.2%	449.8
ant-1	224	4.2%	434.5	**231**	0.0%	415.1	223	1.8%	433.5	**290**	0.0%	431.8	280	0.7%	443.6	274	1.1%	448.6
ant-2	**222**	3.3%	434.5	**231**	0.0%	422.7	**219**	0.0%	433.7	**290**	0.0%	433.9	**278**	0.0%	453.0	**271**	0.0%	450.5
gre-1	224	4.2%	425.4	232	0.4%	401.5	224	2.3%	424.3	295	1.7%	416.7	284	2.2%	432.4	273	0.7%	438.6
gre-2	224	4.2%	425.4	234	1.3%	409.8	220	0.5%	424.6	295	1.7%	419.0	282	1.4%	441.2	**271**	0.0%	440.5
gre-1-det	228	6.0%	< 1	nfsf	-	< 1	235	7.3%	< 1	nfsf	-	< 1	nfsf	-	< 1	nfsf	-	< 1
gre-2-det	229	6.5%	< 1	nfsf	-	< 1	228	4.1%	< 1	304	4.8%	< 1	nfsf	-	< 1	286	5.5%	< 1
	5_1_1			5_1_2			5_1_3			5_2_1			5_2_2			5_2_3		
	Best	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**
Opt	(268)	-	3 d	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
lo	278	3.7%	612.4	293	2.8%	603.7	289	4.0%	614.8	350	2.3%	661.8	362	2.3%	640.7	356	2.0%	635.6
ant-1	272	1.5%	608.9	287	0.7%	610.0	**278**	0.0%	618.7	**342**	0.0%	657.7	357	0.8%	643.1	350	0.3%	653.2
ant-2	**268**	0.0%	609.9	**285**	0.0%	608.1	**278**	0.0%	618.5	**342**	0.0%	662.4	**354**	0.0%	638.2	**349**	0.0%	642.8
gre-1	275	2.6%	591.6	287	0.7%	594.8	279	0.4%	602.2	345	0.9%	637.6	357	0.8%	624.8	351	0.6%	635.7
gre-2	274	2.2%	593.7	288	1.1%	591.8	281	1.1%	600.8	343	0.3%	640.1	357	0.8%	621.9	353	1.1%	622.9
gre-1-det	280	4.5%	< 1	nfsf	-	< 1	285	2.5%	< 1	nfsf	-	< 1	nfsf	-	< 1	nfsf	-	< 1
gre-2-det	278	3.7%	< 1	nfsf	-	< 1	290	4.3%	< 1	353	3.2%	< 1	nfsf	-	< 1	nfsf	-	< 1
	6_1_1			6_1_2			6_1_3			6_2_1			6_2_2			6_2_3		
	Best	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**
Opt	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
lo	360	4.3%	797.6	361	5.6%	780.5	361	4.3%	809.8	441	4.0%	853.9	427	2.2%	780.1	446	4.4%	850.4
ant-1	**345**	0.0%	801.6	**342**	0.0%	797.8	**346**	0.0%	798.6	**424**	0.0%	852.9	419	0.2%	794.5	430	0.7%	852.1
ant-2	346	0.3%	797.8	351	2.6%	790.0	**346**	0.0%	803.6	431	1.7%	863.7	**418**	0.0%	775.9	**427**	0.0%	849.5
gre-1	346	0.3%	783.6	**342**	0.0%	780.7	349	0.9%	779.5	428	0.9%	845.5	423	1.2%	770.9	429	0.5%	836.1
gre-2	352	2.0%	779.0	354	3.5%	779.9	348	0.6%	780.6	430	1.4%	846.1	422	1.0%	750.4	429	0.5%	830.6
gre-1-det	346	0.3%	< 1	nfsf	-	< 1	nfsf	-	< 1	**424**	0.0%	< 1	nfsf	-	< 1	436	2.1%	< 1
gre-2-det	369	7.0%	< 1	nfsf	-	< 1	nfsf	-	< 1	448	5.7%	< 1	nfsf	-	< 1	431	0.9%	< 1

Best found results and average CPU time in seconds of three runs per instance.
* indicates that the global optimum is found.
nfsf = no feasible solution found.

The algorithm without heuristic information, *learning only*, principally works, but with an average gap of 4.3% the results are inferior to both, the ant and probabilistic greedy algorithms. The heuristic information about the similarity of orders seems to support the search of the artificial ants as intended and should be considered during solution construction.

An analysis of the computation times further underlines the potential of ACO. Already for the third problem set (which is still of relatively small size with 20 orders and up to 15 stations), the computation times of the solver literally went through the roof. For *3_1_1*, it took Gurobi almost 17 hours to identify the optimum, while the ant algorithms generated solutions that were constantly within 99.0% of the global optimum in about one minute. For the larger problem sets, the solver could not find any feasible solution within a time span of three days.

Table 4.5: Results for large problem sets 7 and 8

	7_1_1			7_1_2			7_1_3			7_2_1			7_2_2			7_2_3		
	Best	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**
Opt	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
lo	707	6.5%	1494.4	690	6.3%	1484.3	700	6.5%	1459.4	920	5.6%	1625.1	896	6.3%	1685.3	948	6.6%	1710.6
ant-1	**664**	0.0%	1527.1	655	0.9%	1459.9	**657**	0.0%	1455.0	879	0.9%	1656.9	852	1.1%	1712.2	**889**	0.0%	1744.1
ant-2	671	1.1%	1509.4	**649**	0.0%	1531.1	666	1.4%	1453.6	**871**	0.0%	1638.1	**843**	0.0%	1680.9	899	1.1%	1677.3
gre-1	666	0.3%	1494.0	658	1.4%	1439.7	659	0.3%	1417.1	883	1.4%	1617.3	856	1.5%	1676.9	893	0.4%	1696.6
gre-2	679	2.3%	1473.0	654	0.8%	1486.3	675	2.7%	1413.9	875	0.5%	1596.8	846	0.4%	1638.5	906	1.9%	1633.0
gre-1-det	**664**	0.0%	< 1	nfsf	-	< 1	nfsf	-	< 1	nfsf	-	< 1	857	1.7%	< 1	nfsf	-	< 1
gre-2-det	689	3.8%	< 1	nfsf	-	< 1	nfsf	-	< 1	nfsf	-	< 1	861	2.1%	< 1	nfsf	-	< 1
	8_1_1			**8_1_2**			**8_1_3**			**8_2_1**			**8_2_2**			**8_2_3**		
	Best	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**	**Best**	**Gap**	**Time**
Opt	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
lo	687	8.0%	2209.9	713	8.5%	2214.5	718	8.6%	2242.5	971	8.0%	2628.8	970	7.7%	2413.2	951	8.4%	2663.7
ant-1	638	0.3%	2286.9	**657**	0.0%	2261.4	**661**	0.0%	2258.0	901	0.2%	2640.1	**901**	0.0%	2425.9	881	0.5%	2665.0
ant-2	**636**	0.0%	2264.9	670	2.0%	2210.6	663	0.3%	2246.8	**899**	0.0%	2619.0	928	3.0%	2418.0	**877**	0.0%	2612.3
gre-1	640	0.6%	2241.7	**657**	0.0%	2215.1	663	0.3%	2210.7	906	0.8%	2597.8	903	0.2%	2384.7	885	0.9%	2538.3
gre-2	641	0.8%	2215.1	674	2.6%	2158.9	**661**	0.0%	2196.6	901	0.2%	2572.9	928	3.0%	2375.6	882	0.6%	2551.7
gre-1-det	645	1.4%	< 1	nfsf	-	< 1	nfsf	-	< 1	905	0.7%	< 1	nfsf	-	< 1	nfsf	-	< 1
gre-2-det	648	1.9%	< 1	nfsf	-	< 1	nfsf	-	< 1	904	0.6%	< 1	nfsf	-	< 1	nfsf	-	< 1

Best found results and average CPU time in seconds of three runs per instance.
* indicates that the global optimum is found.
nfsf = no feasible solution found.

Note that the solver results reported in Table 4.4 for the instances *4_1_1, 4_2_1* and *5_1_1* (numbers are in brackets) could only be obtained by using the best ACO solution as a starting solution. Again the runs were aborted after three days. Only for the smallest of these three instances, *4_1_1*, the ACO solution could be slightly improved. For the two larger problems, the solver could not find superior solutions within the given three days, even having the advantage of an initial solution.

Note also that the computation times reported in the tables above are a result of the predefined number of iterations and ants, which are both standardized over several problem sets in order to enable a fair comparison of the heuristic algorithms. However, especially for some of the smaller problems, the ant algorithms are able to identify the optimal solutions in significantly less time. For the problem sets one and two, the minimal runtimes lie between 1 and 45 seconds, i.e. ACO requires similar computation times as the solver to achieve the solution quality reported above.

Regarding the success rate and the stability of the solution quality, all ant and probabilistic greedy algorithms perform quite well with standard deviations between 0.66 and 1.00.

4.6 Conclusion

In this paper, a multi-period variant of the mixed-model assembly line sequencing problem is presented. Moreover, different heuristic solution approaches are investigated. The classical mixed-model assembly line sequencing problem addresses situations where different orders have to be scheduled for production, taking different constraints into account. The model developed in this study goes beyond the existing models. It considers that in practice scheduling problems often comprise multiple periods. As in many other papers on mixed-model assembly line sequencing, the problem of avoiding work overload is considered. Additionally, the objective to minimize the overall amount of labour over multiple production periods is taken into account.

To solve the MPMMS, different greedy heuristics and algorithms based on the ant colony optimization metaheuristic are developed. The ant algorithms feature a novel pheromone model, which normalizes pheromone values and heuristic values and directly controls the evaluation of the pheromone trails by linking the iterative delta with the number of iterations. Naturally, this pheromone concept cannot change the fact that a low number of iterations will probably still not be sufficient to solve large-size problems to (near-) optimality. Nevertheless, the pheromone model ensures that the speed of the pheromone trail development (and thus the convergence of the artificial ants towards specific solutions) is balanced adequately. As a consequence, the ant algorithms identify very good solutions within very short computation times.

To evaluate the performance of ACO, an experimental study has been conducted. The solutions obtained via ACO are compared to different greedy algorithms and solutions generated with the solver Gurobi. The results indicate the potential of this ACO approach for solving the MPMMS. The ant algorithms outperformed the other heuristic algorithms and delivered optimal or near-optimal solutions in short time.

For the smaller test problems, the ant algorithms achieved an average solution quality of almost 99% and identified the global optimum for about half of the instances. For the medium and large sized test problems, the solver was unable to find better solutions than ACO within three days, even with the ACO solution available as a starting solution.

For further research, three different topics come to mind. First, related variants of the MPMMS problem should be researched. Second, it would be interesting to see how other proven heuristic methods, such as for example genetic algorithms or simulated annealing, would perform on the MPMMS problem proposed in this paper. Third, the performance of the ant algorithms presented in this study might still be considerably improved, especially for larger problems where the infeasibility of solution becomes more and more problematic. An additional pheromone developing purely based on the infeasibility of partial sequences instead of solution quality might help to guide and speed up the ants search.

5

LOADING AND SEQUENCING HEURISTICS FOR UNRELATED PARALLEL MACHINE SCHEDULING WITH LONG, SEQUENCE-DEPENDENT SETUP TIMES

Jürgen Strohhecker, Jörn-Henrik Thun, and Michael Hamann

Abstract

The purpose of this paper is to develop robust and well-performing heuristics for the scheduling of production orders on parallel machines with sequence-dependent setup times that can easily be used in practice. In particular, this paper examines the combined effects of assignment and sequencing policies on several commonly used performance indicators in the context of unrelated parallel machine scheduling. Discrete event simulation is used in the analysis, since this approach is capable of capturing the complexity of real-world scheduling problems with production orders that differ in many aspects, more than one machine, and varying and product-specific machine speed and setup times that depend on the (dis)similarity of successive orders. Evaluation of 108 unique strategy-scenario combinations provides a robust view of the interaction of loading and sequencing heuristics and production system variability in an unrelated parallel machine setting. The analysis reveals that a loading heuristic based on order quantity and scheduled capacity in combination with a shortest setup heuristic performs best regarding total busy time and service level in most scenarios. The results presented in this paper represent a valuable contribution for practitioners, as well as scientists, since they show that a relatively simple loading and sequencing heuristic is able to save about 17% of total machine busy time.

5.1 Introduction

Since McNaughton's (1959) seminal work on the scheduling of multiple processors, a great variety of papers has evolved in the literature on different variants of parallel machine scheduling. Typically, parallel machine scheduling problems are divided into three categories. The machines can be identical (all machines have the same processing times for all jobs), uniform (the machines have different processing times that are independent of the jobs being processed), or unrelated (the processing time of a machine is different depending on the machine and on the job being processed (Chuang et al., 2010). Such scheduling tasks become even more demanding if sequence-dependent setup times have to be considered explicitly. A setup time is called sequence dependent if it depends on both the job to be processed and the preceding job (Allahverdi and Soroush, 2008).

Prior research has addressed the unrelated parallel machine scheduling problem mostly by using analytical or heuristic optimization techniques (e.g., Fanjul-Peyro and Ruiz, 2011; Mokotoff and Jimeno, 2002). Typically, a static situation with a fixed number of products, machines and jobs known in advance is assumed (e.g., Kim et al., 2002). However, from a practical perspective, companies have to deal with dynamic settings; that is, those in which production orders of varying quantity are received continuously over time. The assignment of orders to machines as well the production sequence have to be determined in a dynamic manner, since the exact number and type of orders during the entire period are not known a priori. Approaches that optimize the order sequence for a longer period of time become obsolete whenever sequencing has to be redone frequently for smaller numbers of orders within a rolling horizon (de Araujo et al., 2007). In such a scenario, the additional value of highly sophisticated analytical or heuristic optimization techniques compared to simple approaches is debatable. Moreover, complex optimization scheduling techniques often lack acceptance due to their low comprehensibility for practitioners. Simple priority-sequencing rules such as, e.g., *first come, first served* (FCFS) or *earliest due date* (EDD) are commonly preferred in practice (Tay and Ho, 2008). However, these simple rules focus on sequencing only and have to be complemented by assignment rules in order to address the scheduling problem adequately in the case of parallel machines.

This paper focuses on the problem of scheduling unrelated parallel machines with sequence-dependent setup times, which is challenging to solve but often observed in practice and therefore highly relevant for practitioners (MacCarthy and Liu, 1993; Mokotoff, 2001; De Paula et al., 2007). It provides robust heuristics for the assignment and sequencing of production orders on parallel machines, matching the requirements of practitioners as described above. The study analyses to what extent practical heuristics can improve manufacturing performance with regard to setup times, processing times and overall capacity utilization. In particular, the study is motivated by the problem of a pharmaceutical company striving to improve packaging performance with regard to processing time and capacity utilization while maintaining acceptably high service levels. In this company, packaging is done on two automated parallel lines, each consisting of a single machine (note that our analysis also applies to lines with multiple machines if they are integrated with a high degree of automation, as is often observed in practice).

In order to capture adequately the complexity of a production system with two unrelated machines and product-specific production rates, sequence-dependent setup times, several hundred product variants and more than 1000 orders per year, discrete event simulation (Arena) is used.

The remainder of this paper is organized as follows. First, the relevant literature in the respective field is reviewed in order to specify the current research gap precisely. Second, the specifics of the unrelated parallel machine scheduling problem are described in detail and the research methodology is introduced. Third, the simulation model, the data set and the simulation analysis are described. Fourth, the results are presented and discussed. Finally, the paper ends by drawing together the main conclusions, highlighting implications for manufacturing companies and suggesting approaches for future research.

5.2 Literature review

The unrelated parallel machines case with sequence-dependent setup times has been paid less attention than related problems with identical or uniform parallel machines or settings without explicit consideration of setup times. Hence, only a few papers can be found in the literature that address scheduling on unrelated parallel machines including sequence-dependent setup times.

A generic literature review on parallel machine scheduling problems regarding different performance measures is provided by Cheng and Sin (1990). Mokotoff (2001) gives a more specific overview by categorizing the literature on scheduling parallel machines according to identical, uniform and unrelated parallel machines. Unrelated parallel machines can be regarded as the more general case, since some machines are faster while some machines are slower and the jobs being processed also influence the processing time (Fanjul-Peyro and Ruiz, 2011).

Kim et al. (2002) suggest simulated annealing in order to solve a scheduling problem for unrelated parallel machines with sequence-dependent setup times with the aim of minimizing total tardiness. Helal et al. (2006) propose a tabu search algorithm for the unrelated parallel machines scheduling problem with sequence- and machine-dependent setup times, striving for a minimization of the makespan. This tabu search algorithm outperforms the partitioning heuristic proposed by Al-Salem (2004) dealing with the same problem. Another algorithm based on tabu search is presented by Bozorgirad and

Logendran (2012), who address a sequence-dependent group scheduling problem on a set of unrelated parallel machines. Rabadi et al. (2006), who also address this problem, introduce a new meta-heuristic referred to as Meta-RaPS, which is capable of identifying all optimal solutions for the small problems described and outperforms the solutions obtained by the existing heuristic for larger problems, such as the partitioning heuristic by Al-Salem (2004). Logendran et al. (2007) propose a methodology in unrelated parallel machine scheduling with sequence-dependent setups developing six different search algorithms based on tabu search to identify the best schedule that gives the minimum weighted tardiness of jobs. Ravetti et al. (2007) introduce the metaheuristic greedy randomised adaptive search procedure as a solution to minimize the total makespan and the weighted delays in an unrelated parallel machine scheduling problem with sequence-dependent setup times and due dates, based on a real case of a refractory brick company. Senthilkumar et al. (2011) study a variant of the unrelated parallel machine scheduling problem for jobs with the objective of minimizing the deviations of job completion times from their corresponding due dates. They propose a hybrid approach based on particle swarm optimization and ant colony optimization, which outperforms the genetic algorithm techniques suggested by Sivrikaya-Şerifoğlu and Ulusoy (1999) and Raja et al. (2008). Investigating the unrelated parallel machine scheduling problem with sequence- and machine-dependent setup times under due date constraints, Ying and Lin (2012) suggest an artificial bee colony algorithm with the aim of minimizing total tardiness. They compare their solution with several state-of-the-art solutions on the same problem set and show that the bee algorithm outperforms the existing ones.

This literature review reveals that the majority of existing papers focus on improving the performance of complex optimization algorithms or developing new methods for different variants of this scheduling problem. Typically, an evaluation in this context is based on how well the proposed algorithms perform on large test problems in comparison to alternative approaches that serve as benchmarks. However, from a practitioner’s perspective, it remains unclear to what extent simple and easy-to-use scheduling heuristics and dispatching rules might contribute to performance improvements in a real-world parallel machine setting with sequence-dependent setup times and unrelated machines. Hence, it is the purpose of this paper to close this research gap by evaluating different techniques for the assignment and sequencing of production orders on two unrelated parallel machines.

5.3 Problem description and heuristic strategies

5.3.1 Problem setting

Figure 5.1 provides an overview of the entire scheduling problem that we address in our study. First, production orders differing in their specific characteristics have to be assigned to one of the two unrelated parallel machines. Second, the orders have to be sequenced before they are released to production.

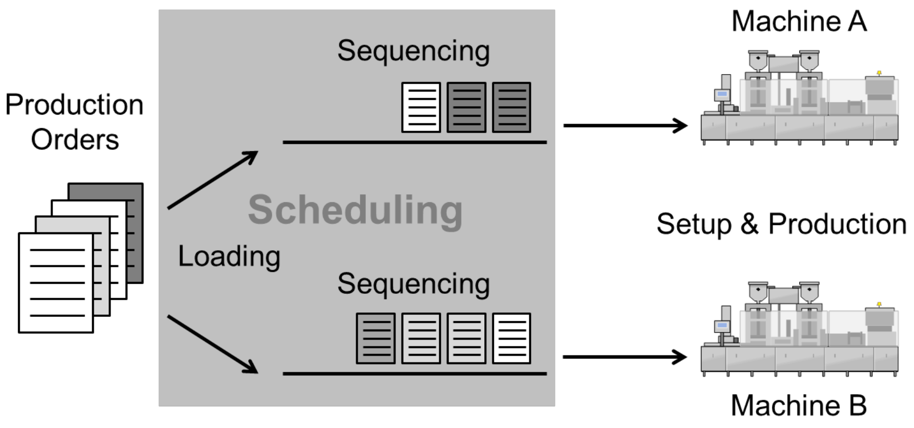

Figure 5.1: Scheduling task with two unrelated parallel machines and sequence-dependent setup times

Ideally, the two interrelated tasks – loading and sequencing – of the scheduling problem could be integrated and solved simultaneously in order to receive a truly optimal solution (Shanker and Tzen, 1985). However, as such, this problem is $\mathcal{NP}$-hard and therefore requires complex and CPU-intensive optimization methods in order to obtain a solution (Graham et al., 1979; Lenstra et al., 1977). By decomposing the scheduling problem into two parts the problem structure is simplified, which provides the basis for the application of heuristics with a lower conceptual complexity.

Hence, the two problems are solved sequentially in a combined manner: the outcome of the first problem, the loading problem, is regarded as input for the second problem, the sequencing problem. Note that in general orders could also first be sequenced and then assigned to a line. However, with respect to the sequence dependence of setup times this does not seem to be reasonable: if sequencing is done prior to loading, order sequences that result in short setups are likely to be torn apart again when the products are assigned to different machines.

In order to test heuristics for the assignment and sequencing problem, discrete event simulation modelling and analysis have been chosen as the research methodology (e.g., Law, 2007). This approach allows for a production system to be modelled in a realistic way and for practical decision rules to be tested in a wide array of scenarios.

In order to ensure the practical relevance of this research and the applicability of the results, our study is grounded in an empirical case – the packaging production system at a pharmaceutical company. In this company, a portfolio of 288 different products was filled into bottles of ten different sizes, which were put into cartons of seven different types and completed with leaflets in three different sizes. Packaging was accomplished on two parallel lines that differed from each other in terms of technology, resulting in different machine speeds (which also differ between products) and setup times. Setup times were dependent on the scope of the changeover between two production runs: setup was slow if orders differed in various aspects, for instance product, bottle size, carton size and leaflet size; it was fast if only the order number changed and everything else remained unchanged. Only very few technical restrictions had to be considered in assigning production orders to the two packaging lines so that nearly complete exchangeability was provided: almost all products could be packaged on either of the two lines.

For the simulation study, production data of one year were available. Specifically, a list of 1127 orders including product specifics, due dates and quantities was at hand, whereby in this case an order always contains one particular product. In addition to this, we also had access to a broad range of master data, for instance the product-specific production rates as well as sequence-dependent setup times for both lines. Furthermore, data on the work shifts for the two packaging lines were available.

5.3.2 Heuristic strategies

In this section, different heuristics for the underlying problem are explained. In the first step, we discuss three heuristics for the assignment problem. In the second, we suggest three different heuristics for the consecutive sequencing problem. In order to address the requirement for practical usage, we focus on simple heuristics. This approach is reasonable since the planning problem is characterized by its dynamic short-term focus and a situation that can be described by a rolling horizon; that is, only short time periods (typically a single-digit number of days) are considered for solving the assignment and sequencing problem. As motivation for using a rolling horizon approach, de Araujo et al. (2007) raise the question of why detailed schedules for later periods should be calculated with great effort if they are never implemented. Furthermore, Drexl and Kimms (1997) stress the importance of a rolling horizon due to its practical relevance.

For solving the assignment problem the following three heuristics are proposed: *random loading* (RL), *scheduled capacity-based assignment* (SC), and an *order quantity and scheduled capacity assignment* (OQSC). RL reflects a situation that is often observed in practice where the assignment is simply defined ex ante during product development, fixed in the master data and finally used for scheduling. When it comes in the short term to imbalances in machine utilization and bottleneck situations, orders are reassigned on an ad hoc basis following no strict rules. The result, as we found in our case company, could be interpreted as more or less random loading. We model this heuristic by assigning orders to one of the packaging lines using a predefined probability that takes line speed, setup times and shift model differences into account.

According to the SC heuristic, the already scheduled capacity on both packaging lines is taken into account. More precisely, setup and production times of all orders assigned to a specific line and not yet finished are summed up (for instance, 35.9 h for line A and 45.1 h for line B) and percentages of scheduled capacity are calculated (44.3% on line A and 55.7% on line B). Based on production rates, setup times and shift plans, desired loading percentages are calculated (for example, 55% on line B, 45% on line A). In addition a rule has to be set for which machine is loaded first (for instance, line B is filled first as it is generally faster). Then, based on the SC heuristic and using the example data provided above, a new order would be assigned to line A, as the actual scheduled capacity percentage of line B is 55.7%, which is larger than the desired value of 55%.

The OQSC heuristic refines the SC heuristic by always assigning orders with production quantities larger than a specific threshold (for example, 10,000 units) to one of the lines (for instance, because one line is generally faster when producing but slower when setups have to be made) instead of following a fixed, predefined loading rule for all orders regardless of their size, as in the SC heuristic. All orders smaller than the threshold follow the SC heuristic.

Besides the assignment heuristics, different dispatching rules are applied. A dispatching rule determines which job is given the highest priority to be processed next on a particular machine. We selected three approaches that have either been applied in previous research or are frequently used by practitioners: *earliest due date* (EDD), s*hortest processing time* (SPT), and *shortest setup time* (SST) (see, e.g., Kia et al., 2010; Pickardt and Branke, 2012; Vinod and Sridharan, 2009).

Following EDD, jobs are processed with respect to their due date: if a job has a due date that is earlier than the other jobs, this one will be taken first. This is a very simple approach of sequencing due to the fact that other aspects such as capacity-oriented performance criteria are not taken into consideration. However, we use EDD as a benchmark because it is widely considered in both theory and practice. Moreover, our case company relied on this dispatching rule.

According to SPT, jobs are sequenced based on their processing times (not including setup times, as these can be determined only after sequencing is completed). The highest priority is given to the job with the shortest processing time. Hence, the job with the shortest processing time is scheduled first.

Following SST, all products are categorized in terms of their setup-relevant product characteristics, which in our case are bottle size, carton/leaflet size and bulk type – a procedure similar to creating so-called setup families. The more two subsequent production orders differ with respect to these characteristics, the higher is the setup time. Therefore, orders should be sequenced so that subsequent orders are as similar as possible to the preceding ones. The SST heuristic achieves this by lexicographical sorting of the production orders (e.g., Hendizadeh et al., 2008; Wemmerlöw and Vakharia, 1991). In our case, the bottle size is considered first since it has only 10 different variants, followed by the carton/leaflet size with 12 variants and the bulk type with 80 variants (see the appendix for detailed numbers).

Note that many production scheduling problems belong to the class of $\mathcal{NP}$-hard problems and hence require appropriate and often complex optimization methods if they are large (Graham et al., 1979; Lenstra et al., 1977). However, in this case the number of orders to be scheduled is relatively small within the individual time buckets (approximately 25 to 43 orders maximum per bucket) due to the rolling planning horizon of ten days. Hence, we expect even simple dispatching rules to yield relatively good results.

Altogether, the heuristics for the assignment problem and the sequencing problem can be combined so that nine different combinations are possible as heuristic strategies. We define a heuristic strategy as a combination of two or more heuristics that are applied simultaneously to solve an overall scheduling problem. In our case, a heuristic strategy consists of one heuristic for the assignment problem and one heuristic for the sequencing problem. For example, one heuristic strategy might combine the heuristics OQ and SPT. Table 5.1 provides an overview of all possible heuristic strategies.

Table 5.1: Possible heuristic strategies

	EDD	SPT	SST
RL	RL/EDD	RL/SPT	RL/SST
SC	SC/EDD	SC/SPT	SC/SST
OQSC	OQSC/EDD	OQSC/SPT	OQSC/SST

In the following, these nine strategies are applied to the scheduling problem as described in section 5.3.1. In order to generalize our analysis of which heuristic strategies are most effective, different scenarios are developed, since it can be assumed that the effectiveness of a certain heuristic strategy depends on several operational factors such as the diversity of setup times and production rates or the overall similarity of the parallel machines. Note that the assignment of products to packaging lines primarily influences the processing time, while the sequencing of orders affects the setup time. In order to evaluate the impact of individual heuristics, we report processing times and setup times separately.

5.4 Model, data, and simulation analysis

5.4.1 Simulation model and input data

The model of the production system was developed using the simulation software Arena (Kelton et al., 2010). It consists of four major parts: (1) production order creation; (2) order scheduling; (3) order completion; and (4) production. Figure 5.2 provides an overview of the model.

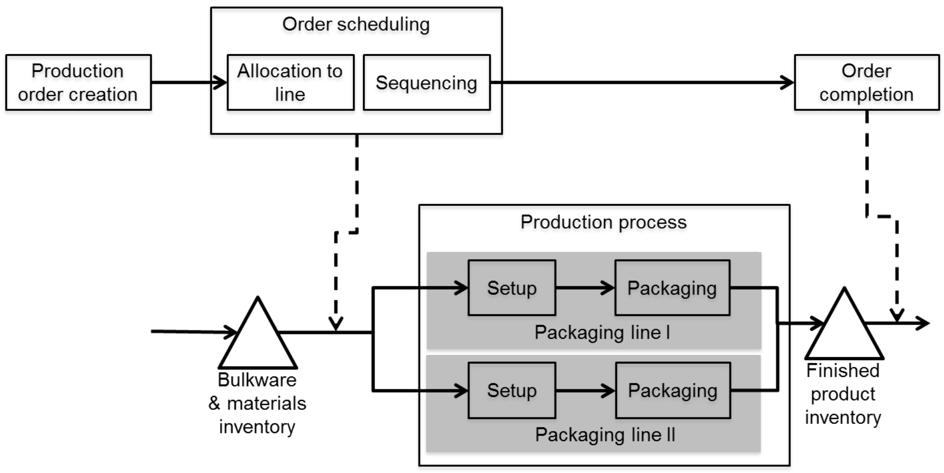

Figure 5.2: Overview of the discrete event simulation model structure

With regard to a single simulation run, 1127 production orders (as in the practical case) are created using a bootstrapping procedure; that is, the sequence and due dates of the actual orders are changed using randomly computed numbers (actual order quantities in a randomized sequence). Due dates are set based on delivery times that are drawn from a normal distribution with a mean value of 28 days and a standard deviation of 4 days. All attributes characterizing an order, such as product number, order quantity, bulk type, bottle size, carton size and leaflet size, are assigned. Then orders are sent to the scheduling part of the model, which includes both the three line-loading heuristics RL, SC and OQSC and the three sequencing heuristics EDD, SPT and SST, as described above.

While applying one of the three line-loading heuristics, it is assumed that all products can be produced on both packaging lines. Once an order is loaded to a packaging line, the processing time on this line is computed by dividing the product-specific production rate by the quantity ordered. Note that production rates differ between products and lines (examples are given in Table 5.4).

In order to determine the processing sequence, the jobs first wait in two queues (one for each line) that resemble a pre-shop pool, which is commonly used in the context of order review and release methods (Bergamaschi et al., 1997; Sabuncuoglu and Karapınar, 1999). Before they are released to production after the ten-day time bucket, the orders are sorted according to the respective sequencing heuristic. The waiting time of a maximum of ten days prevents the production system from being overloaded; that is, the waiting time functions in the sense of a workload control (Stevenson et al., 2005). Furthermore, it reduces lead times as well as the inventory of work in progress (WIP) and finished goods (Hendry and Wong, 1994), at least as long as the orders are waiting in a virtual rather than a physical way.

Once released from the sequencing queues, it is assumed that production is never starved of bulk-ware and materials. Therefore, the setup process can start as soon as the production order is released and the assigned packaging line becomes available. The setup time between two subsequent orders is determined by the following four factors: order change, bulk type change, bottle size change and carton or leaflet size change. Depending on the number of changes that have to be prepared, the production order is assigned to one of the four setup categories XS, S, M and L, as listed in Table 5.2.

Table 5.2: Setup matrix and categories

Carton/leaflet change	Bottle size change	Bulk change	Order change	Setup category
-	-	-	X	XS
-	X	-	X	S
X	-	-	X	S
X	X	-	X	M
-	-	X	X	M
-	X	X	X	M
X	-	X	X	M
X	X	X	X	L

Grounded in our case data but also generalizing to some extent from this specific situation, four setup scenarios – as shown in Table 5.3 – were defined. Scenarios DS (which is closest to the case) and DW represent situations where setup matrices differ (D) between packaging lines A and B. The sequence dependence of setup times is strong (S) in scenario DS and weak (W) in scenario DW. This can be seen from larger mean absolute deviations in setup times across the four categories for scenario DW compared to DS (line A: 0.833 > 0.275; line B: 0.963 > 0.381). Scenarios ES and EW again represent strongly and weakly sequence-dependent situations, but now the setup times are equal (E) for both lines. As is typical for pharmaceutical production and packaging processes, setup times are generally long (see, for instance, Strohhecker et al., 2014). In our case they can easily exceed the production time, especially when order quantities are limited.

Table 5.3: Setup time scenarios (setup time in hours)

		Setup category					
Scenario	Line	XS	S	M	L	Mean	MAD
DS	A	1.333	2.167	3.167	3.667	2.583	0.833
	B	1.540	2.503	3.659	4.236	2.985	0.963
DW	A	2.100	2.500	2.700	3.000	2.575	0.275
	B	2.425	2.887	3.118	3.464	2.973	0.318
ES	A	1.542	2.083	3.417	4.083	2.781	0.969
	B	1.542	2.083	3.417	4.083	2.781	0.969
EW	A	2.250	2.650	2.950	3.250	2.775	0.325
	B	2.250	2.650	2.950	3.250	2.775	0.325

Besides the setup scenarios, three different production rate scenarios were defined. Table 5.4 shows for five exemplary products how many units per hour can be produced on either line for each of these scenarios. It also compiles descriptive measures.

Scenario FCL stands for the case where one line is always faster compared to the other line (F) and the ratio between the lines stays nearly constant (C); that is, line B is between 2.50 and 2.89 times as fast as line A. Moreover, the differences between production rates are large (L); that is, line B produces on average 1898.09 units per hour more than line A. In this, scenario FCL resembles closely the situation found in the case company. Scenario FDS describes a situation where one line is faster (F) but, in contrast to scenario FCL, variations across products are high; that is, for some products the processing rates are almost equal while for other products there are great differences (D) – ranging from B running as fast as A to B being three times as fast as A. Moreover, in the FDS scenario the absolute differences between production rates are smaller (S) than in the FCL scenario (a mean of 1091.96 vs. 1898.09 units per hour). Scenario NDS illustrates the case where no line shows a permanently higher production rate (N); that is, for some products line A is faster and for some products line B is faster (implying different ratios, D). Furthermore, the differences regarding the absolute processing rates are very small (S). Both lines can be seen as nearly equal regarding production rates.

Table 5.4: Illustration and descriptive data of production rate scenarios (units per hour)

	Scenario FCL		Scenario NDS		Scenario FDS	
Product	Line A	Line B	Line A	Line B	Line A	Line B
1	591.24	1528.82	973.74	1146.32	835.39	1284.67
2	1027.79	2947.94	2023.53	1952.20	1982.83	1992.90
3	1044.02	2821.77	2046.04	1819.75	1929.89	1935.90
4	1026.55	2798.43	2001.50	1823.47	1347.47	2477.50
⋮	⋮	⋮	⋮	⋮	⋮	⋮
288	1516.17	3885.44	2734.74	2666.87	2475.21	2926.40

Min	4.66	12.29	7.91	9.04	4.26	12.69
Max	4659.03	12840.97	8415.27	9084.73	5574.53	11925.47
Mean	1130.60	3028.69	2083.30	2075.99	1533.67	2625.63
SD	564.20	1518.85	1042.52	1055.81	815.42	1369.30

In order to simplify and generalize the analysis, identical shift models are used for both lines. A five-day workweek with two eight-hour shifts per day is assumed, resulting in a capacity of 4016 hours per line and year (based on 251 workdays per year).

In order to build as much confidence as possible in the simulation model described above, a large number of validity tests have been conducted. Model validation is essential in any simulation study and has attracted intense methodological research (e.g., Cochran, 1987;

Naylor and Finger, 1967). Numerous test runs were conducted in which the assignment of entity attributes, for instance the different setup categories and setup times, were carefully traced in debug mode. Using animation features provided by the software, the flow logic of all different entity types was traced and examined. In addition, the computation of all model variables was carefully debugged. Simulation results for selected performance measures, for instance total production time, were compared against values calculated statically using the master data and a spreadsheet. For all validity tests that we conducted, the model showed plausible behaviour.

5.4.2 Performance measures

In the existing literature, different performance dimensions are considered in the context of parallel machine scheduling. Widely used measures include completion time–based, due date–based and flow time–based criteria (e.g., Cheng and Sin, 1990; Schmidt, 2000). For instance, Chuang et al. (2010) minimize total completion time, while Cortés et al. (2012) also take due date–based measures into account. In addition, utilization metrics and busy time measures, which are related to completion-based measures, are minimized (e.g., Flammini et al., 2009). Since the scheduling heuristics analysed in this study mainly aim at increasing manufacturing performance, the total busy time – that is, the sum of production and setup times – is chosen as the major performance criterion (which correlates perfectly with the average utilization in our case). In addition to this, we also report results for a flow time indicator, more precisely the 'packaging cycle time', which measures the time span between production order release to the loading and sequencing process and termination of the packaging process, not including storage time as finished product. To cover typical conflicting behaviour between completion time–based and due date–based goals in short-term machine scheduling, and motivated by the existence of service-level agreements with customers in our case company, the beta-service level (or fill rate) is also taken into account. Principally, the service level is considered as an externally given restriction; that is, when optimizing the busy time, service levels need to be kept constantly at a certain level, for example 95%.

5.5 Results

The simulation was started with empty sequencing queues and inventories (both finished products and work in progress). Each simulation run was terminated as soon as all 1127 production orders had been delivered to customers. All scenarios were simulated with 100 replications, which provided for an analysis of performance measures with half-widths in the ± 5% range.

Using a full factorial experimental design, the nine heuristic strategies (see Table 5.1) are evaluated in a total of 12 different scenarios that result from combining the three production rate scenarios (see Table 5.4) with the four setup time scenarios (see Table 5.3). This generates 108 unique strategy-scenario combinations (see also Table 5.10 in the appendix), providing a robust view of the interaction of loading and sequencing heuristics and production system variability in an unrelated parallel machine setting. The Process Analyzer included in the Arena software package was used to run the simulations as well as to analyse and statistically compare the results. By default, the best experiments are identified using 95% confidence (Kelton et al., 2010).

The impact of the various scheduling heuristics can best be seen by looking at the two most extreme scenario combinations: the case with almost identical machines (NDS) and weakly sequence-dependent setup times (EW); and the case with very different machines (FCL) and strongly sequence-dependent setup times (DS), the latter being the closest to the situation we observed in our case company. Hence, we focus on these two scenario combinations in this section. The outcomes for the remaining eight scenario combinations are included in the appendix.

The results of the nine simulation experiments for the production rate and setup time scenario combinations resembling the empirical case (FCL-DS) are shown in Table 5.5. For all but one key performance indicator (KPI) – the packaging cycle time – we find a clearly dominating heuristic strategy: the combination of the loading heuristic OQSC with the sequencing heuristic SST. This strategy leads to the lowest total busy time, the best beta service level and the lowest utilization of lines A and B. Regarding the packaging cycle time KPI, the shortest processing time sequencing rule outperforms the shortest setup time heuristic in combination with OQSC significantly by 0.41 days. This is in line with other research showing that the SPT rule often results in the lowest cycle times (e.g., Chang et al., 1996; Lu et al., 2011; Vinod and Sridharan, 2011). However, using SPT instead of SST in combination with OQSC comes with significant disadvantages in all other KPIs; for instance, total busy time increases by 417 hours (5.9%).

Table 5.5: KPI mean values for production rate and setup time scenario combination FCL-DS

Exp. No.	Line Loading Heuristic	Sequencing heuristic	Total Busy Time	Total Setup Time	Total Production Time	Beta Service Level	Packaging Cycle Time	Utilization of Lines A and B
1	RL	EDD	8,159 h	4,250 h	3,909 h	86.9%	16.54 d	85.0%
2	RL	SPT	7,989 h	4,080 h	3,909 h	86.5%	14.01 d	83.5%
3	RL	SST	7,595 h	3,685 h	3,909 h	94.2%	13.75 d	79.6%
4	SC	EDD	8,219 h	4,242 h	3,977 h	91.8%	15.42 d	86.1%
5	SC	SPT	8,013 h	4,064 h	3,949 h	90.2%	13.27 d	84.0%
6	SC	SST	7,550 h	3,654 h	3,896 h	95.1%	13.46 d	79.2%
7	OQSC	EDD	7,600 h	4,114 h	3,485 h	95.0%	13.01 d	79.7%
8	OQSC	SPT	7,438 h	3,959 h	3,479 h	94.4%	**11.58 d**	78.0%
9	OQSC	SST	**7,021 h**	**3,545 h**	**3,476 h**	**95.7%**	11.99 d	**73.6%**

Best values are highlighted in bold; mean values calculated from 100 replications.

Regardless of the loading heuristic used, in terms of total busy time SST always significantly outperforms SPT and EDD. This also holds for the beta service level measure. Only when directly compared does EDD achieve slightly higher averages than SPT; however, only the difference between experiments 4 and 5 – that is, 91.8% vs. 90.2% – is statistically significant. Moreover, SPT has statistically significant advantages in lowering the cycle time only when combined with the OQSC heuristic. In combination with the SC and RL heuristics, no significant differences are observed.

When comparing the nine experiments by means of the box and whisker diagrams shown in Figure 5.3, it becomes obvious that the various combinations of loading and sequencing heuristics also result in different KPI distributions. The span between minimum and maximum total busy time decreases from RL over SC to OQSC (with the exception of experiment 9 compared to 6). Sequencing, in contrast, does not have much of an impact. This changes to some extent when the service level and cycle time KPIs are considered. The maximum distance between minimum and maximum service level can be drastically reduced if RL is combined with SST instead of EDD or SPT. The same is true for the SC and OQSC heuristics. Overall, one can state that the OQSC-SST strategy does not only outperform the other combinations in terms of average total busy time and average beta service level, it also decreases variations in cycle time and service level to a minimum. When focusing only on total busy time, the strategies with the lowest differences between minimum and maximum values are OQSC/EDD and OQSC/SPT.

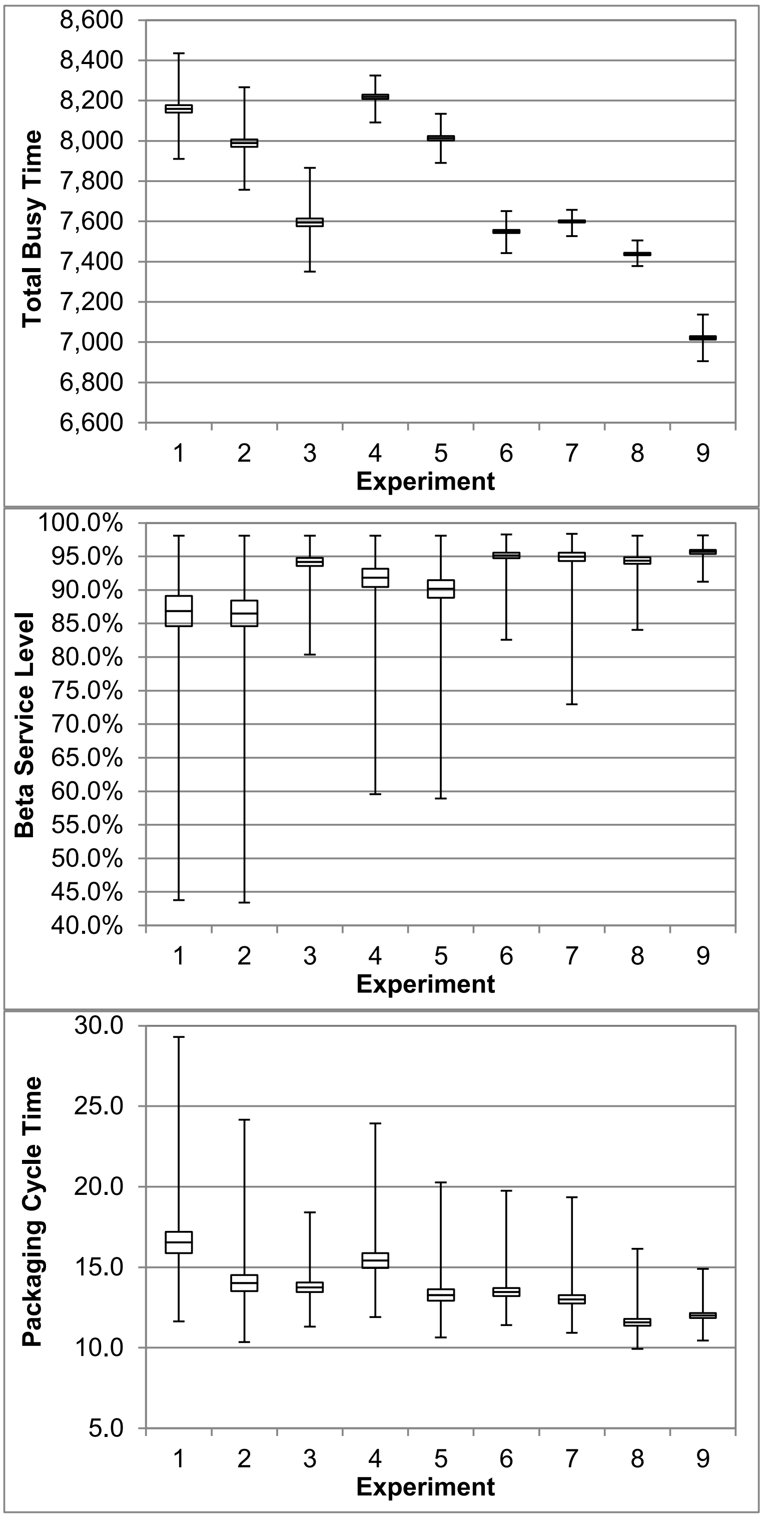

Figure 5.3: Box and whisker diagrams of KPIs for the scenario combination FCL-DS

The results for the set of experiments that contrast most with the FCL-DS scenario combination – that is, the production rate scenario with nearly identical line characteristics (NDS) and the setup time scenario with weakly sequence-dependent setup times (EW) – are shown in Table 5.6.

When comparing these outcomes to the results shown in Table 5.5, it can be seen that all measures are improved. For instance, random loading with earliest due date sequencing (experiment 1 vs. 64) results in 877 h lower total busy time (731 h less setup time and 146 h less production time), 8.3 percentage points better service level, and 4.22 days less cycle time. This result is not surprising. With nearly identical machines and equal setup times that are only weakly sequence dependent, the production system can be expected to show superior performance.

Table 5.6: KPI mean values for production rate and setup time scenario combination NDS-EW

Exp. No.	Line Loading Heuristic	Sequencing heuristic	Total Busy Time	Total Setup Time	Total Production Time	Beta Service Level	Packaging Cycle Time	Utilization of Lines A and B
64	RL	EDD	7,282 h	3,518 h	3,764 h	95.2%	12.32 d	76.4%
65	RL	SPT	7,210 h	3,447 h	3,764 h	94.2%	10.91 d	75.6%
66	RL	SST	7,050 h	3,286 h	3,764 h	95.2%	12.41 d	73.9%
67	SC	EDD	7,246 h	3,518 h	3,727 h	95.6%	12.07 d	76.0%
68	SC	SPT	7,168 h	3,444 h	3,725 h	95.0%	10.71 d	75.2%
69	SC	SST	6,988 h	3,275 h	3,713 h	95.7%	12.17 d	73.3%
70	OQSC	EDD	6,990 h	3,491 h	**3,499 h**	**95.7%**	11.55 d	73.3%
71	OQSC	SPT	6,927 h	3,427 h	3,501 h	95.4%	**10.53 d**	72.7%
72	OQSC	SST	**6,751 h**	**3,246 h**	3,504 h	95.7%	11.56 d	**70.8%**

Best values are highlighted in bold; mean values calculated from 100 replications.

Much less obvious, though, is the result that the OQSC/SST strategy is still the best choice when total makespan and service level matter most. It achieves clearly and significantly the lowest total makespan, while differences to the next best experiments in beta service level are not significant. It also results in the smallest maximum–minimum difference for the service level and cycle time KPIs, with an only marginally larger value regarding total makespan (111.96 h vs. 104.01 h in experiment 71). Only when minimizing cycle time is the most important goal is SPT in combination with OQSC significantly lower. This can be attributed to the SPT rule that prefers orders with lower packaging times over those with longer ones, resulting typically in very good cycle time measures. The same effect can be observed with the random loading and scheduled capacity loading heuristics. SPT performs in all three cases significantly better than SST and EDD.

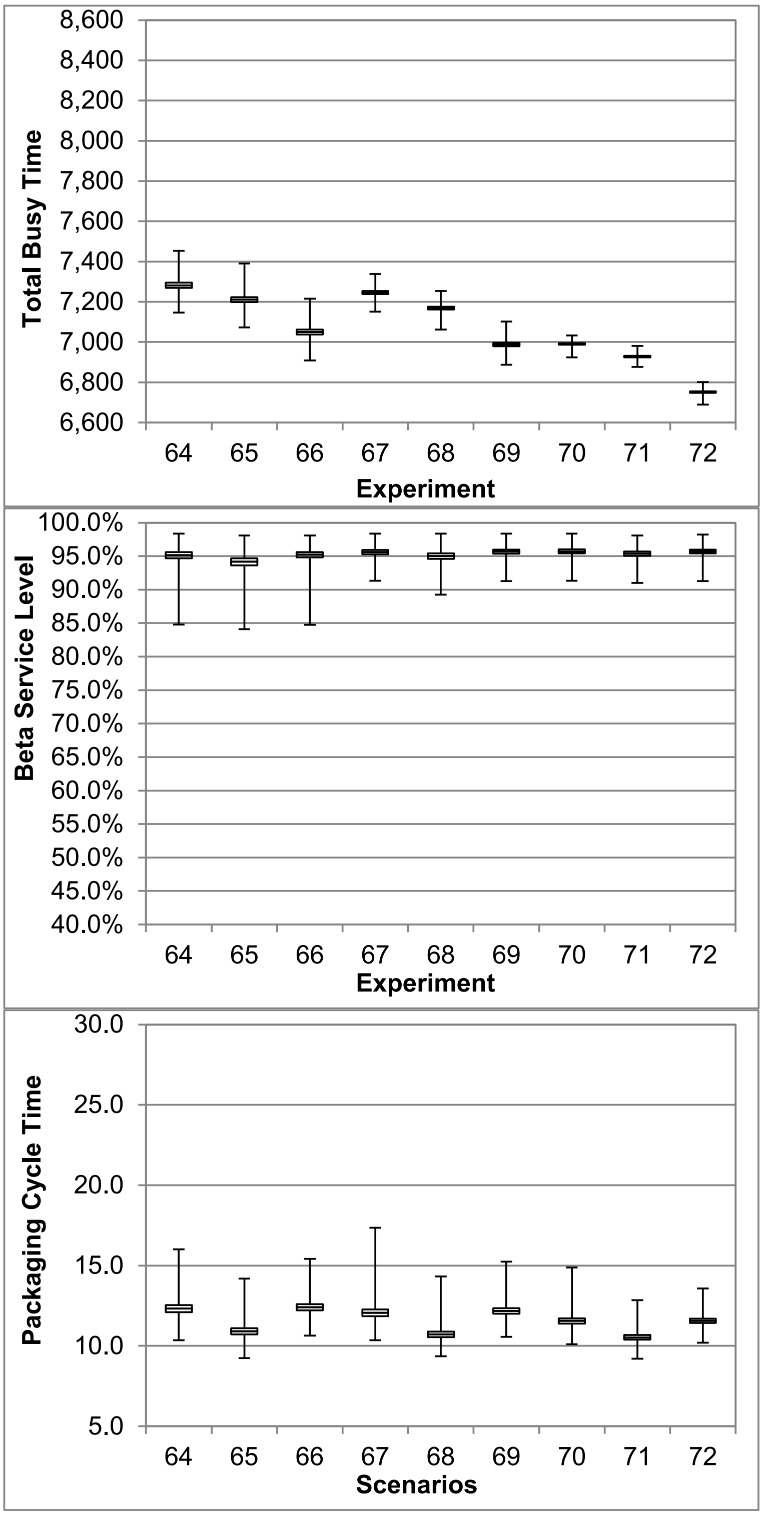

Figure 5.4: Box and whisker diagrams of KPIs for the scenario combination NDS-EW

Table 5.7 gives an overview of the best-performing heuristic strategies based on all 108 simulation experiments. Separately for each of the 12 production rate and setup time scenario combinations and each of the three KPIs – total busy time, beta service level, packaging cycle time – the best-performing heuristic strategy was identified using the Arena Process Analyzer with 95% confidence (Kelton et al., 2010).

Table 5.7: Analysis of best-performing heuristic strategies for all 12 production rate and setup time scenario combinations

Heuristic strategies	KPIs	Total Makespan	Beta Service Level	Packaging Cycle Time
RL	EDD	0	2	0
RL	SPT	0	0	2
RL	SST	0	5	0
SC	EDD	0	7	0
SC	SPT	0	2	5
SC	SST	0	12	0
OQSC	EDD	0	11	0
OQSC	SPT	0	6	11
OQSC	SST	12	12	4

Table 5.8 illustrates this process for the FCL-DS scenario combination. Comparing the beta service level results for the nine heuristic strategies identifies SC/SST, OQSC/EDD and OQSC/SST as the three strategies that perform significantly better than the other six. Counting the scenarios for which a strategy is among the best-performing ones results in Table 5.8.

Table 5.8: Best-performing heuristic strategies for the FCL-DS scenario combination

		Best performing heuristic strategies (95 % Confidence)		
Line Loading Heuristic	Sequencing heuristic	Total Makespan	Beta Service Level	Packaging Cycle Time
RL	EDD			
RL	SPT			
RL	SST			
SC	EDD			
SC	SPT			
SC	SST		X	
OQSC	EDD		X	
OQSC	SPT			X
OQSC	SST	X	X	

Based on the figures shown in Table 5.8, one can state that the OQSC/SST strategy is clearly outperforming all other strategies when total busy time is the most important KPI to be minimized. At the same time, this strategy is also always among the best regarding the beta service level measure. Moreover, in four out of twelve scenario combinations, this strategy also leads to the best results for the packaging cycle time measure. Therefore, combining OQSC with SST achieves very good results for a broad range of scenario combinations. Only if minimizing packaging cycle time is a more important goal than total busy time and service level should OQSC be combined with SST in seven out of the twelve scenario combinations.

5.6 Conclusion

This paper analyses different heuristic strategies for the scheduling of production orders on two unrelated parallel machines with sequence-dependent setup times. In particular, the study examines the combined effects of loading and sequencing heuristics on several performance indicators such as total busy time (total setup and production time), customer service level and packaging cycle time.

The results reveal that depending on the specific context – that is, shop floor characteristics concerning setup times and processing rates – the heuristic strategies have a significant impact on manufacturing performance despite their low conceptual and computational complexity. Surprisingly, the results also hold for scenarios with very similar machines and weakly sequence-dependent setup times (NDS-EW). Due to these similarities it could have been expected that the generation of savings in terms of setup times and production times by assigning and sequencing production orders would hardly be possible; that is, that the shift of an order to another line or another slot in the production sequence would cause only minor changes regarding the performance indicators. However, this is not the case. The difference between the worst-performing strategy (RL/NDD) and the best-performing one (OQSC/SST) is a respectable 531 hours' total busy time (7.9%), 1.88 days' packaging cycle time and 1.6 percentage points in service level.

Therefore, it must be highlighted that the heuristic strategies show a rather strong impact on manufacturing performance over all 12 scenario combinations. For instance, in the scenario combination FCL-ES, the best combination of a loading rule (OQSC) and a sequencing heuristic (SST) leads to a decrease of 1202 hours' total busy time per year when compared to the worst strategy (RL/EDD); that is, 17%. In particular, a combination of an OQSC approach for the machine loading problem and an SST for the sequencing problem yielded very good results for the majority of the simulated scenarios.

This paper's results clearly indicate that already simple heuristic approaches have a substantial impact on manufacturing performance and should be given greater consideration. Note that the results can be generalized in principle, since the study is based on multiple scenarios that reflect distinctive production environments as observed in different industries. By evaluating the scenario that best fits the actual characteristics of their production setting, practitioners can easily adopt the appropriate heuristic strategy with the highest improvement potential for production scheduling.

For further research, we suggest investigation of the approaches analysed in this paper for similar problems of production scheduling. This would enable practitioners to improve production performance without the need to use highly sophisticated optimization techniques.

6

Conclusion

This section provides a brief summary of the main contribution of this thesis.

The first two papers present advancements and applications of the ant colony optimization metaheuristic. In chapter 3, a novel generic pheromone model for ant colony optimization is proposed. The corresponding algorithm is named NormANTS, which relates to the key characteristics of the algorithm: Based on the normalization of pheromone and heuristic values between zero and one and the coupling of the iterative pheromone delta with the total number of iterations, the search behaviour of artificial ants is improved. A computational study has been carried out in order to examine the impact of the novel pheromone model and to evaluate the overall performance of NormANTS. The analysis of the development of resampling ratio, similarity ratio, and pheromone trails throughout the iterative search shows that the novel pheromone model provides a high level of control over the pheromone trail evolution independently from problem specifics. Accordingly, the ant colony explores the solution space more effectively with a reduced risk of premature convergence or stagnation situations. The comparison against other ant algorithms for 33 frequently used MKP benchmark problems from the literature reveals that, despite its problem-unspecific character, NormANTS outperforms alternative ACO designs using traditional pheromone models.

In chapter 4, a novel multi-period variant of the mixed-model sequencing problem is presented. The classical mixed-model assembly line sequencing problem, which addresses situations where different orders have to be scheduled for production, is extended by a multi-period view. Complementary to the traditional objective to minimize work overload, the MPMMS considers the optimization of the workforce strength, i.e. the numbers of workers along the assembly line. In practice, this objective is of relevance when different orders demand a different number of workers in order to adhere to the given cycle time of the assembly line. Based on the techniques introduced in chapter 3, a specifically adapted ant algorithm is developed. The algorithm has been applied to 48 different randomly generated problem instances of various sizes and compared against results obtained with Gurobi as well as problem-specific probabilistic and deterministic greedy algorithms. The comprehensive experimental study revealed the potential of the ant algorithm, which considerably outperformed the other algorithms.

Chapter 5 differs from chapter 3 and 4 in terms of research perspective and methodology. It does not aim at improving highly sophisticated heuristic algorithms but lays the focus on the practicality of heuristic approaches. As in practice short-term planning usually makes use of rolling planning horizons, detailed scheduling is typically done in short-range time buckets only. As a consequence, merely a moderate number of orders needs to be scheduled at a time for a short period of several days (instead of hundreds or thousands orders per year). The question arises to what extent and under which preconditions simple and consequently more practical scheduling heuristics improve operational performance. A simulation study has been carried out in which 108 different strategy-scenario combinations for a two-step scheduling problem (comprising the loading and sequencing of orders on unrelated parallel machines) have been investigated. The results show that some of the heuristic approaches considerably improve both, completion time–based and flow time–based performance measures, while basically keeping due date–based criteria (i.e. service levels) on a constant level. Especially a combination of the heuristics OQSC and SST performed well, even in scenarios where the characteristics of the operational environment as such provided hardly any improvement potential due to similar machines and similar setup times with low sequence dependency.

Considered conjointly, the results of this work particularly support two aspects regarding heuristic solution methods for combinatorial optimization problems. First, the computational studies presented in chapter 3 and 4 indicate the potential of the ant colony metaheuristic in general and NormANTS in particular. Both newly developed ant algorithms obtained promising results and outperformed a set of different benchmark algorithms. Second, however, the simulation study reported in chapter 5 shows that simple heuristics (which are considerably less complex than ACO), can deliver satisfying results in real-world settings, too. Although the potential of simpler techniques depends on operational conditions and characteristics of the specific optimization problem at hand, it seems worthwhile to keep them in mind as an alternative to the nowadays more popular families of metaheuristics such as evolutionary algorithms, variable neighbourhood search, simulated annealing, tabu search, scatter search, or ant colony optimization.

The results of chapter 5 exemplarily illustrate the managerial relevance and the potential business impact of the application of heuristic methods for real-world combinatorial optimization problems. Choosing an appropriate heuristic strategy for solving the production scheduling problem on unrelated parallel machines has a rather strong impact on manufacturing performance (note that this is valid for different scenario combinations that reflect distinctive production environments as observed in different industries). For instance, the difference between the best and worst heuristic strategy can be as large as 17% in terms of the major objective dimension, i.e. machine busy time in hours. In other words, applying adequate optimization heuristics in this case results in capacity savings of up to 17% per annum. The fact that the worst heuristic strategy of the simulation quite accurately reflects the actual planning procedure of the case company the production data has been taken from, suggests that the potential of heuristic optimization methods at least partly remains unexploited by some practitioners.

While it needs to be kept in mind that the specific results of chapter 5 refer to one particular optimization problem only and can hence not be generalized to other combinatorial optimization problems, there is no doubt about the general relevance of heuristic approaches for combinatorial optimization since managerial decisions typically comprise combinatorial optimization problems whenever it comes to the optimization of cost, time, risk, quality, efficiency, or profit (Talbi, 2009). Important examples besides the scheduling problem from chapter 5 are portfolio selection, capital budgeting, design of marketing campaigns, investment planning, facility layout and location, sizing of truck fleets, transportation planning, bus and train scheduling, assignment of workers to jobs such as drivers to buses and airline crew scheduling, and inventory control (Grötschel and Lovász, 1995). All of these problems share the trait that with growing problem size, the identification of good solutions becomes more and more difficult. Consequently, all of these problems, which come from different areas of business management, require heuristic solution approaches such as the ones presented in this study.

Future research should address the transfer of NormANTS to other combinatorial optimization problems in order to further investigate its potential. As the proposed pheromone model is not problem specific, there are no significant barriers regarding an application to other problems. Moreover, the algorithm could be developed further from a methodological perspective. For instance, it could be combined with additional local search procedures to enhance performance. Considering the MPMMS model introduced in chapter 4, related variants of this multi-period problem should be investigated. In particular, an extension of the model considering the practice of floating workers could be of interest and relevance for both practitioners and researchers. Furthermore, other proven heuristic methods such as, e.g., genetic algorithms, simulated annealing, or tabu search could be tested on the MPMMS. Regarding the simpler scheduling heuristics analysed in chapter 5, alternative priority rules for the loading and sequencing of production orders on parallel machines, for instance the *longest processing time* or the *critical ratio* dispatching rule (see, e.g., Guinet et al., 1996; Kia et al., 2010), should be analysed and tested. Future research should also deal with an application of the heuristics to similar problems of production scheduling in order to enable practitioners to improve production performance without the necessity to use highly sophisticated combinatorial optimization techniques.

BIBLIOGRAPHY

Akgündüz, O.S. and Tunalı, S. (2010), "An adaptive genetic algorithm approach for the mixed-model assembly line sequencing problem", *International Journal of Production Research*, Vol. 48 No. 17, pp. 5157–5179.

Alaya, I., Solnon, C. and Ghédira, K. (2004), "Ant algorithm for the multi-dimensional knapsack problem", *International Conference on Bioinspired Optimization Methods and their Applications, BIOMA 2004*, pp. 63–72.

Alaykỳran, K., Engin, O. and Döyen, A. (2007), "Using ant colony optimization to solve hybrid flow shop scheduling problems", *The International Journal of Advanced Manufacturing Technology*, Vol. 35 No. 5-6, pp. 541–550.

Allahverdi, A. and Soroush, H.M. (2008), "The significance of reducing setup times/setup costs", *European Journal of Operational Research*, Vol. 187 No. 3, pp. 978–984.

De Araujo, S.A., Arenales, M.N. and Clark, A.R. (2007), "Joint rolling-horizon scheduling of materials processing and lot-sizing with sequence-dependent setups", *Journal of Heuristics*, Vol. 13 No. 4, pp. 337–358.

Aron, S., Pasteels, J.M. and Deneubourg, J.L. (1989), "Trail-laying behaviour during exploratory recruitment in the Argentine ant, Iridomyrmex humilis (Mayr)", *Biology of Behaviour*, Vol. 14 No. 3, pp. 207–217.

Bautista, J. and Cano, A. (2011), "Solving mixed model sequencing problem in assembly lines with serial workstations with work overload minimisation and interruption rules", *European Journal of Operational Research*, Vol. 210 No. 3, pp. 495–513.

Beasley, J.E. (1990), "OR-Library: distributing test problems by electronic mail", *Journal of the Operational Research Society*, Vol. 41 No. 11, pp. 1069–1072.

Bell, J.E. and McMullen, P.R. (2004), "Ant colony optimization techniques for the vehicle routing problem", *Advanced Engineering Informatics*, Vol. 18 No. 1, pp. 41–48.

Bergamaschi, D., Cigolini, R., Perona, M. and Portioli, A. (1997), "Order review and release strategies in a job shop environment: A review and a classification", *International Journal of Production Research*, Vol. 35 No. 2, pp. 399–420.

Blum, C. and Roli, A. (2003), "Metaheuristics in combinatorial optimization: Overview and conceptual comparison", *ACM Computing Surveys (CSUR)*, Vol. 35 No. 3, pp. 268–308.

Boryczka, U. (2007), “Ants and Multiple Knapsack Problem”, *6th International Conference on Computer Information Systems and Industrial Management Applications, CISIM’07*, IEEE Computer Society, Elk, Polen, pp. 149–154.

Boysen, N. (2005), “Ameisenalgorithmen. Optimierung nach dem Vorbild der Natur”, *Wirtschaftswissenschaftliches Studium*, Vol. 34 No. 11, pp. 607–612.

Boysen, N., Fliedner, M. and Scholl, A. (2009), “Sequencing mixed-model assembly lines: Survey, classification and model critique”, *European Journal of Operational Research*, Vol. 192 No. 2, pp. 349–373.

Boysen, N., Kiel, M. and Scholl, A. (2011), “Sequencing mixed-model assembly lines to minimise the number of work overload situations”, *International Journal of Production Research*, Vol. 49 No. 16, pp. 4735–4760.

Bozorgirad, M.A. and Logendran, R. (2012), “Sequence-dependent group scheduling problem on unrelated-parallel machines”, *Expert Systems with Applications*, Vol. 39 No. 10, pp. 9021–9030.

Bullnheimer, B., Hartl, R.F. and Strauß, C. (1997), *A new rank based version of the Ant System*, Wien, Österreich: Institute of Management Science, University of Vienna.

Cano-Belmán, J., Ríos-Mercado, R.Z. and Bautista, J. (2010), “A scatter search based hyper-heuristic for sequencing a mixed-model assembly line”, *Journal of Heuristics*, Vol. 16 No. 6, pp. 749–770.

Chang, Y.-L., Sueyoshi, T. and Sullivan, R.S. (1996), “Ranking dispatching rules by data envelopment analysis in a job shop environment”, *IIE Transactions*, Vol. 28 No. 8, pp. 631–642.

Cheng, T.C.E. and Sin, C.C.S. (1990), “A state-of-the-art review of parallel-machine scheduling research”, *European Journal of Operational Research*, Vol. 47 No. 3, pp. 271–292.

Chuang, M.-C., Liao, C.-J. and Chao, C.-W. (2010), “Parallel machine scheduling with preference of machines”, *International Journal of Production Research*, Vol. 48 No. 14, pp. 4139–4152.

Chu, P.C. and Beasley, J.E. (1998), “A genetic algorithm for the multidimensional knapsack problem”, *Journal of Heuristics*, Vol. 4 No. 1, pp. 63–86.

Cochran, J.K. (1987), “Techniques for ascertaining the validity of large-scale production simulation models”, *International Journal of Production Research*, Vol. 25 No. 2, pp. 233–244.

Cordón, O., de Viana, I.F., Herrera, F. and Moreno, L. (2000), “A new ACO model integrating evolutionary computation concepts: the best-worst ant system”, in Dorigo, M., Middendorf, M. and Stützle, T. (Eds.), *Abstract Proceedings of ANTS 2000 - From Ant Colonies to Artificial Ants: Second International Workshop on Ant Algorithms*, IRIDIA, Universite Libre de Bruxelles, Brüssel, pp. 22–29.

Cortés, B.M., García, J.C.E. and Hernández, F.R. (2012), “Multi-objective flow-shop scheduling with parallel machines”, *International Journal of Production Research*, Vol. 50 No. 10, pp. 2796–2808.

Deneubourg, J.L., Aron, S., Goss, S. and Pasteels, J.M. (1990), “The self-organizing exploratory pattern of the argentine ant”, *Journal of Insect Behavior*, Vol. 3 No. 2, pp. 159–168.

Dobson, G. (1982), “Worst-case analysis of greedy heuristics for integer programming with nonnegative data”, *Mathematics of Operations Research*, Vol. 7 No. 4, pp. 515–531.

Dorigo, M., Birattari, M. and Stützle, T. (2006), “Ant colony optimization: artificial ants as a computational intelligence technique”, *IEEE Computational Intelligence Magazine*, Vol. 1 No. 4, pp. 28–39.

Dorigo, M., Bonabeau, E. and Theraulaz, G. (2000), “Ant algorithms and stigmergy”, *Future Generation Computer Systems*, Vol. 16 No. 8, pp. 851–871.

Dorigo, M. and Gambardella, L.M. (1997), “Ant colony system: a cooperative learning approach to the traveling salesman problem”, *IEEE Transactions on Evolutionary Computation*, Vol. 1 No. 1, pp. 53–66.

Dorigo, M., Maniezzo, V. and Colorni, A. (1991a), *The ant system: An autocatalytic optimizing process*, Mailand: Dipartimento di Elettronica, Politecnico di Milano.

Dorigo, M., Maniezzo, V. and Colorni, A. (1991b), *Positive feedback as a search strategy*, Mailand: Dipartimento di Elettronica, Politecnico di Milano.

Dorigo, M., Maniezzo, V. and Colorni, A. (1996), “Ant system: optimization by a colony of cooperating agents”, *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, Vol. 26 No. 1, pp. 1–13.

Dorigo, M. and Socha, K. (2007), “Ant Colony Optimization”, in Gonzalez, T.F. (Ed.), *Approximation Algorithms and Metaheuristics*, CRC Press, pp. 26.1–26.11.

Dorigo, M. and Stützle, T. (2003), “The ant colony optimization metaheuristic: Algorithms, applications, and advances”, *Handbook of Metaheuristics*, Springer, pp. 250–285.

Dorigo, M. and Stützle, T. (2004), *Ant Colony Optimization*, The MIT Press, Cambridge, Massachusetts.

Dorigo, M. and Stützle, T. (2010), “Ant colony optimization: overview and recent advances”, *Handbook of Metaheuristics*, Springer, pp. 227–263.

Drexl, A. and Kimms, A. (1997), “Lot sizing and scheduling—survey and extensions”, *European Journal of Operational Research*, Vol. 99 No. 2, pp. 221–235.

Emde, S., Boysen, N. and Scholl, A. (2010), “Balancing mixed-model assembly lines: a computational evaluation of objectives to smoothen workload”, *International Journal of Production Research*, Vol. 48 No. 11, pp. 3173–3191.

Ervolina, T.R., Ettl, M., Lee, Y.M. and Peters, D.J. (2009), “Managing product availability in an assemble-to-order supply chain with multiple customer segments”, *Supply Chain Planning*, Springer, pp. 1–24.

Fanjul-Peyro, L. and Ruiz, R. (2011), “Size-reduction heuristics for the unrelated parallel machines scheduling problem”, *Computers & Operations Research*, Vol. 38 No. 1, pp. 301–309.

Fidanova, S. (2002), “ACO Algorithm for MKP Using Various Heuristic Information”, *Revised Papers from the 5th International Conference on Numerical Methods and Applications*, Springer-Verlag, pp. 438–444.

Flammini, M., Monaco, G., Moscardelli, L., Shachnai, H., Shalom, M., Tamir, T. and Zaks, S. (2009), “Minimizing total busy time in parallel scheduling with application to optical networks”, *IEEE International Symposium on Parallel & Distributed Processing, IPDPS 2009*, pp. 1–12.

Fuellerer, G., Doerner, K.F., Hartl, R.F. and Iori, M. (2009), “Ant colony optimization for the two-dimensional loading vehicle routing problem”, *Computers & Operations Research*, Vol. 36 No. 3, pp. 655–673.

Gagné, C., Gravel, M. and Price, W.L. (2006), “Solving real car sequencing problems with ant colony optimization”, *European Journal of Operational Research*, Vol. 174 No. 3, pp. 1427–1448.

Gambardella, L.M. and Dorigo, M. (1996), “Solving symmetric and asymmetric TSPs by ant colonies”, *IEEE Conference on Evolutionary Computation (ICEC’96)*, IEEE Press, Nagoya, Japan, pp. 622–627.

Gambardella, L.M., Taillard, E.D. and Dorigo, M. (1999), “Ant colonies for the QAP”, *Journal of the Operational Research Society*, Vol. 50 No. 2, pp. 167–176.

Garey, M.R. and Johnson, D.S. (1979), *Computers and Intractability: A Guide to the Theory of NP-completeness*, W. H. Freeman & Co, New York, NY, USA.

Garnier, S., Gautrais, J. and Theraulaz, G. (2007), “The Biological Principles of Swarm Intelligence”, *Swarm Intelligence*, Vol. 1 No. 1, pp. 3–31.

Gendreau, M. and Potvin, J.-Y. (2005), “Metaheuristics in combinatorial optimization”, *Annals of Operations Research*, Vol. 140 No. 1, pp. 189–213.

Glover, F. (1986), “Future paths for integer programming and links to artificial intelligence”, *Computers & Operations Research*, Vol. 13 No. 5, pp. 533–549.

Glover, F. and Kochenberger, G.A. (2003), *Handbook of Metaheuristics*, Springer Science & Business Media, available at: https://books.google.de/books?hl=de&lr=&id=O_10T_KeqOgC&oi=fnd&pg=PR9&dq=metaheuristics&ots=MD89_KVCPh&sig=ZzLugM1On_S3Omm9VokOMcr6Aso (accessed 26 February 2015).

Golle, U., Rothlauf, F. and Boysen, N. (2014), “Car sequencing versus mixed-model sequencing: A computational study”, *European Journal of Operational Research*, Vol. 237 No. 1, pp. 50–61.

Goss, S., Aron, S., Deneubourg, J.L. and Pasteels, J.M. (1989), “Self-organized shortcuts in the Argentine ant”, *Naturwissenschaften*, Vol. 76 No. 12, pp. 579–581.

Gottlieb, J., Puchta, M. and Solnon, C. (2003), “A study of greedy, local search, and ant colony optimization approaches for car sequencing problems”, *Applications of Evolutionary Computing*, Springer, pp. 246–257.

Graham, R.L., Lawler, E.L., Lenstra, J.K. and Kan, A.H.G. (1979), “Optimization and approximation in deterministic sequencing and scheduling: a survey”, *Annals of Discrete Mathematics*, Vol. 5, pp. 287–326.

Grassé, P. P. (1959), “La Reconstruction Du Nid Et Les Coordinations Interindividuelles Chez Bellicositermes Natalensis Et Cubitermes Sp. La Théorie De La Stigmergie: Essai D'interprétation Du Comportement Des Termites Constructeurs”, *Insectes Sociaux*, Vol. 6 No. 1, pp. 41–80.

Grötschel, M. and Lovász, L. (1995), “Combinatorial optimization”, *Handbook of Combinatorics*, Vol. 2, pp. 1541–1597.

Guinet, A., Solomon, M.M., Kedia, P.K. and Dussauchoy, A. (1996), “A computational study of heuristics for two-stage flexible flowshops”, *International Journal of Production Research*, Vol. 34 No. 5, pp. 1399–1415.

Helal, M., Rabadi, G. and Al-Salem, A. (2006), "A tabu search algorithm to minimize the makespan for the unrelated parallel machines scheduling problem with setup times", *International Journal of Operations Research*, Vol. 3 No. 3, pp. 182–192.

Van Hemert, J.I. and Bäck, T. (2002), "Measuring the searched space to guide efficiency: The principle and evidence on constraint satisfaction", *Parallel Problem Solving from Nature—PPSN VII*, Springer, pp. 23–32.

Hendizadeh, S.H., Faramarzi, H., Mansouri, S.A., Gupta, J.N. and ElMekkawy, T.Y. (2008), "Meta-heuristics for scheduling a flowline manufacturing cell with sequence dependent family setup times", *International Journal of Production Economics*, Vol. 111 No. 2, pp. 593–605.

Hendry, L.C. and Wong, S.K. (1994), "Alternative order release mechanisms: a comparison by simulation", *International Journal of Production Research*, Vol. 32 No. 12, pp. 2827–2842.

Holweg, M. and Greenwood, A. (2001), "Product Variety, Life Cycles, and Rate of Innovation–Trends in the UK Automotive Industry", *Lean Enterprise Research Centre, Cardiff Business School, Wales, UK*, available at: http://www.3daycar.com/mainframe/publications/library/ProductVariety.pdf (accessed 23 July 2014).

Holweg, M. and Pil, F.K. (2001), "Start with the customer", *MIT Sloan Management Review*, Vol. 43 No. 1, pp. 74–83.

Ke, L., Feng, Z., Ren, Z. and Wei, X. (2010), "An ant colony optimization approach for the multidimensional knapsack problem", *Journal of Heuristics*, Vol. 16 No. 1, pp. 65–83.

Kelton, W.D., Sadowski, R.P. and Swets, N.B. (2010), *Simulation with Arena*, McGraw-Hill, 5thed.

Kia, H.R., Davoudpour, H. and Zandieh, M. (2010), "Scheduling a dynamic flexible flow line with sequence-dependent setup times: a simulation analysis", *International Journal of Production Research*, Vol. 48 No. 14, pp. 4019–4042.

Kim, D.-W., Kim, K.-H., Jang, W. and Frank Chen, F. (2002), "Unrelated parallel machine scheduling with setup times using simulated annealing", *Robotics and Computer-Integrated Manufacturing*, Vol. 18 No. 3, pp. 223–231.

Kong, M., Tian, P. and Kao, Y. (2008), "A new ant colony optimization algorithm for the multidimensional Knapsack problem", *Computers & Operations Research*, Vol. 35 No. 8, pp. 2672–2683.

Krantz, S.G. and Parks, H.R. (2014), "The P/NP Problem", *A Mathematical Odyssey*, Springer, pp. 217–254.

Law, A.M. (2007), *Simulation Modeling and Analysis*, McGraw-Hill, 4thed.

Lawler, E.L. (1976), *Combinatorial optimization: networks and matroids*, Courier Corporation, available at: https://books.google.de/books?hl=de&lr=&id=m4MvtFenVjEC&oi=fnd&pg=PA1&dq=combinatorial+optimization:+networks+and+matroids&ots=xOdTuJSMGz&sig=BYuEXlT0cWlnkwlGQYHaAWPUpL8 (accessed 16 February 2015).

Leguizamón, G. and Michalewicz, Z. (1999), "A new version of ant system for subset problems", *Congress on Evolutionary Computation, CEC 99*, Washington, DC, USA, Vol. 2.

Lenstra, J.K., Rinnooy Kan, A.H.G. and Brucker, P. (1977), "Complexity of machine scheduling problems", *Annals of Discrete Mathematics*, Vol. 1, pp. 343–362.

Levine, J. and Ducatelle, F. (2004), "Ant colony optimization and local search for bin packing and cutting stock problems", *Journal of the Operational Research Society*, Vol. 55 No. 7, pp. 705–716.

Logendran, R., McDonell, B. and Smucker, B. (2007), "Scheduling unrelated parallel machines with sequence-dependent setups", *Computers & Operations Research*, Vol. 34 No. 11, pp. 3420–3438.

López-Ibáñez, M., Blum, C., Ohlmann, J.W. and Thomas, B.W. (2013), "The travelling salesman problem with time windows: Adapting algorithms from travel-time to makespan optimization", *Applied Soft Computing*, Vol. 13 No. 9, pp. 3806–3815.

Lu, H.L., Huang, G.Q. and Yang, H.D. (2011), "Integrating order review/release and dispatching rules for assembly job shop scheduling using a simulation approach", *International Journal of Production Research*, Vol. 49 No. 3, pp. 647–669.

McMullen, P.R. (2001), "An ant colony optimization approach to addressing a JIT sequencing problem with multiple objectives", *Artificial Intelligence in Engineering*, Vol. 15 No. 3, pp. 309–317.

McNaughton, R. (1959), "Scheduling with deadlines and loss functions", *Management Science*, Vol. 6 No. 1, pp. 1–12.

Mokotoff, E. (2001), "Parallel machine scheduling problems: a survey", *Asia Pacific Journal of Operational Research*, Vol. 18 No. 2, pp. 193–242.

Mokotoff, E. and Jimeno, J.L. (2002), “Heuristics based on partial enumeration for the unrelated parallel processor scheduling problem”, *Annals of Operations Research*, Vol. 117 No. 1-4, pp. 133–150.

Montgomery, J. and Randall, M. (2002), “Anti-Pheromone as a Tool for Better Exploration of Search Space”, in Dorigo, M., Di Caro, G., Sampels, M., Hartmanis, J. and Van Leeuwen, J. (Eds.), *Proceedings of ANTS 2002 - From Ant Colonies to Artificial Ants: Third International Workshop on Ant Algorithms*, Lecture Notes in Computer Science, Springer, Berlin, pp. 100–110.

Moradi, H. and Zandieh, M. (2013), “An imperialist competitive algorithm for a mixed-model assembly line sequencing problem”, *Journal of Manufacturing Systems*, Vol. 32 No. 1, pp. 46–54.

Moradi, H., Zandieh, M. and Mahdavi, I. (2011), “Non-dominated ranked genetic algorithm for a multi-objective mixed-model assembly line sequencing problem”, *International Journal of Production Research*, Vol. 49 No. 12, pp. 3479–3499.

Morin, S., Gagné, C. and Gravel, M. (2009), “Ant colony optimization with a specialized pheromone trail for the car-sequencing problem”, *European Journal of Operational Research*, Vol. 197 No. 3, pp. 1185–1191.

Morrison, R.W. and De Jong, K.A. (2002), “Measurement of population diversity”, *Artificial Evolution*, Springer, pp. 31–41.

Naylor, T.H. and Finger, J.M. (1967), “Verification of computer simulation models”, *Management Science*, Vol. 14 No. 2, p. B–92.

Osman, I.H. and Kelly, J.P. (1996), “Meta-heuristics: an overview”, *Meta-Heuristics*, Springer, pp. 1–21.

Papadimitriou, C.H. (1994), *Computational complexity*, Addison-Wesley Reading, Massachusetts.

Pickardt, C.W. and Branke, J. (2012), “Setup-oriented dispatching rules–a survey”, *International Journal of Production Research*, Vol. 50 No. 20, pp. 5823–5842.

Pil, F.K. and Holweg, M. (2004), “Linking product variety to order-fulfillment strategies”, *Interfaces*, Vol. 34 No. 5, pp. 394–403.

Pintea, C.-M. (2014), “Combinatorial Optimization”, *Advances in Bio-inspired Computing for Combinatorial Optimization Problems*, Intelligent Systems Reference Library, Springer Berlin Heidelberg, pp. 21–28.

Rabadi, G., Moraga, R.J. and Al-Salem, A. (2006), “Heuristics for the unrelated parallel machine scheduling problem with setup times”, *Journal of Intelligent Manufacturing*, Vol. 17 No. 1, pp. 85–97.

Raja, K., Arumugam, C. and Selladurai, V. (2008), “Non-identical parallel-machine scheduling using genetic algorithm and fuzzy logic approach”, *International Journal of Services and Operations Management*, Vol. 4 No. 1, pp. 72–101.

Ravetti, M.G., Mateus, G.R., Rocha, P.L. and Pardalos, P.M. (2007), “A scheduling problem with unrelated parallel machines and sequence dependent setups”, *International Journal of Operational Research*, Vol. 2 No. 4, pp. 380–399.

Röder, A. and Tibken, B. (2006), “A methodology for modeling inter-company supply chains and for evaluating a method of integrated product and process documentation”, *European Journal of Operational Research*, Vol. 169 No. 3, pp. 1010–1029.

Sabuncuoglu, I. and Karapınar, H.Y. (1999), “Analysis of order review/release problems in production systems”, *International Journal of Production Economics*, Vol. 62 No. 3, pp. 259–279.

Al-Salem, A. (2004), “Scheduling to minimize makespan on unrelated parallel machines with sequence dependent setup times”, *Engineering Journal of the University of Qatar*, Vol. 17 No. 1, pp. 177–187.

Schmidt, G. (2000), “Scheduling with limited machine availability”, *European Journal of Operational Research*, Vol. 121 No. 1, pp. 1–15.

Scholl, A. and Klein, R. (1999), “Balancing assembly lines effectively–a computational comparison”, *European Journal of Operational Research*, Vol. 114 No. 1, pp. 50–58.

Seçkiner, S.U., Eroğlu, Y., Emrullah, M. and Dereli, T. (2013), “Ant colony optimization for continuous functions by using novel pheromone updating”, *Applied Mathematics and Computation*, Vol. 219 No. 9, pp. 4163–4175.

Senthilkumar, K.M., Selladurai, V., Raja, K. and Thirunavukkarasu, V. (2011), “A hybrid algorithm based on pso and aco approach for solving combinatorial fuzzy unrelated parallel machine scheduling problem”, *European Journal of Scientific Research*, Vol. 64 No. 2, pp. 293–313.

Shanker, K. and Tzen, Y.-J.J. (1985), “A loading and dispatching problem in a random flexible manufacturing system”, *International Journal of Production Research*, Vol. 23 No. 3, pp. 579–595.

Shih, W. (1979), “A branch and bound method for the multiconstraint zero-one knapsack problem”, *Journal of the Operational Research Society*, Vol. 30 No. 4, pp. 369–378.

Sidaner, A., Bailleux, O. and Chabrier, J.-J. (2002), “Measuring the spatial dispersion of evolutionary search processes: Application to walksat”, *Artificial Evolution*, Springer, pp. 77–87.

Sivrikaya-Şerifoğlu, F. and Ulusoy, G. (1999), “Parallel machine scheduling with earliness and tardiness penalties”, *Computers & Operations Research*, Vol. 26 No. 8, pp. 773–787.

Solnon, C. (2008), “Combining two pheromone structures for solving the car sequencing problem with Ant Colony Optimization”, *European Journal of Operational Research*, Vol. 191 No. 3, pp. 1043–1055.

Solnon, C. and Fenet, S. (2006), “A study of ACO capabilities for solving the maximum clique problem”, *Journal of Heuristics*, Vol. 12 No. 3, pp. 155–180.

Stevenson, M., Hendry, L.C. and Kingsman†, B.G. (2005), “A review of production planning and control: the applicability of key concepts to the make-to-order industry”, *International Journal of Production Research*, Vol. 43 No. 5, pp. 869–898.

Strohhecker, J., Sibbel, R. and Dick, M. (2014), “Integrating Kanban principles in a pharmaceutical campaign production system”, *Production Planning & Control*, Vol. 25 No. 15, pp. 1247–1263.

Stützle, T. and Hoos, H.H. (2000), “MAX-MIN Ant System”, *Future Generation Computer Systems*, Vol. 16 No. 9, pp. 889–914.

Talbi, E.-G. (2009), *Metaheuristics: from design to implementation*, John Wiley & Sons, Vol. 74, available at: https://books.google.de/books?hl=de&lr=&id=SIsa6zi5XV8C&oi=fnd&pg=PR7&dq=metaheuristics&ots=-9RKpOewFw&sig=_IxF2l3Dliab5EIbtKa_h409NJE (accessed 26 February 2015).

Tay, J.C. and Ho, N.B. (2008), “Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems”, *Computers & Industrial Engineering*, Vol. 54 No. 3, pp. 453–473.

Vinod, V. and Sridharan, R. (2009), “Simulation-based metamodels for scheduling a dynamic job shop with sequence-dependent setup times”, *International Journal of Production Research*, Vol. 47 No. 6, pp. 1425–1447.

Vinod, V. and Sridharan, R. (2011), “Simulation modeling and analysis of due-date assignment methods and scheduling decision rules in a dynamic job shop production system”, *International Journal of Production Economics*, Vol. 129 No. 1, pp. 127–146.

Weingartner, H.M. and Ness, D.N. (1967), “Methods for the solution of the multidimensional 0/1 knapsack problem”, *Operations Research*, Vol. 15 No. 1, pp. 83–103.

Wemmerlöw, U. and Vakharia, A.J. (1991), “Job and family scheduling of a flow-line manufacturing cell: a simulation study”, *IIE Transactions*, Vol. 23 No. 4, pp. 383–393.

Wester, L. and Kilbridge, M. (1963), “The assembly line model-mix sequencing problem”, *Proceedings of the third international conference on Operations Research, Oslo*, pp. 247–260.

Wolsey, L.A. and Nemhauser, G.L. (2014), *Integer and combinatorial optimization*, John Wiley & Sons, available at: https://books.google.de/books?hl=de&lr=&id=MvBjBAAAQBAJ&oi=fnd&pg=PR1&dq=complexity+theory+combinatorial+optimization&ots=1czxAEM3rD&sig=cdleiK7lxr_wAh3WBpP6QRrKZ10 (accessed 27 January 2015).

Yang, S., Jiang, Y. and Nguyen, T.T. (2013), “Metaheuristics for dynamic combinatorial optimization problems”, *IMA Journal of Management Mathematics*, Vol. 24 No. 4, pp. 451–480.

Ying, K.-C. and Lin, S.-W. (2012), “Unrelated parallel machines scheduling with sequence-and machine-dependent setup times and due date constraints”, *International Journal of Innovative Computing, Information and Control*, Vol. 8 No. 5, pp. 3279–3297.

Zhu, Q. and Zhang, J. (2011), “Ant colony optimisation with elitist ant for sequencing problem in a mixed model assembly line”, *International Journal of Production Research*, Vol. 49 No. 15, pp. 4605–4626.

APPENDIX

Table 5.9: Production order characteristics

Bottle Sizes	# Products	Carton & Leaflet Size	# Products
200ml 70x37mm	14	145 x 70 x 100 - 148 x 300	1
240ml 80mm	11	47 x 47 x 80 - --	1
300ml 70x19mm	10	47 x 47 x 80 - 148 x 300	142
400ml 70x19mm	8	55 x 55 x 102 - --	2
400ml 80mm	9	55 x 55 x 102 - 148 x 210	5
600ml 102mm	6	55 x 55 x 102 - 148 x 300	97
80ml 64mm	142	55 x 55 x 86 - 148 x 300	1
100ml 70x37mm	17	55 x 55 x 90 - 148 x 210	4
150ml 70x37mm	18	55 x 55 x 90 - 148 x 300	1
150ml 80mm	53	67 x 67 x 132 - --	2
		67 x 67 x 132 - 148 x 300	27
		70 x 70 x 145 - 148 x 300	5
Total	288	Total	288

Table 5.10: Experiment design with 108 strategy-scenario combinations

		Variations in Production System's Characteristics											
		FCL				NDS				FDS			
		DS	DW	ES	EW	DS	DW	ES	EW	DS	DW	ES	EW
Loading	Sequencing	FCL-DS	FCL-DW	FCL-ES	FCL-EW	NDS-DS	NDS-DW	NDS-ES	NDS-EW	FDS-DS	FDS-DW	FDS-ES	FDS-EW
RL	EDD	1	10	19	28	37	46	55	64	73	82	91	100
	SPT	2	11	20	29	38	47	56	65	74	83	92	101
	SST	3	12	21	30	39	48	57	66	75	84	93	102
SC	EDD	4	13	22	31	40	49	58	67	76	85	94	103
	SPT	5	14	23	32	41	50	59	68	77	86	95	104
	SST	6	15	24	33	42	51	60	69	78	87	96	105
OQSC	EDD	7	16	25	34	43	52	61	70	79	88	97	106
	SPT	8	17	26	35	44	53	62	71	80	89	98	107
	SST	9	18	27	36	45	54	63	72	81	90	99	108

Legend:

RL	Random loading
SC	Scheduled capacity based assignment
OQSC	Order quantity and scheduled capacity assignment
EDD	Earliest due date
SPT	Shortest processing time
SST	Shortest setup time
FCL	One line faster for all products, nearly constant relations, on average large differences
NDS	Varying differences in line speed, differing relations, small differences
FDS	One line faster for all products, different relations, on average small differences
DS	Different setup times between lines A and B, strong sequence dependence
DW	Different setup times between lines A and B, weak sequence dependence
ES	Equal setup times between lines A and B, strong sequence dependence
EW	Equal setup times between lines A and B, weak sequence dependence

Table 5.11: Experiment results for the 90 strategy-scenario combinations not reported in the text

Exp. No.	Line Loading Heuristic	Sequencing heuristic	Total Busy Time	Total Setup Time	Total Production Time	Beta Service Level	Packaging Cycle Time	Utilization of Lines A and B
10	RL	EDD	7,467 h	3,558 h	3,909 h	94.3%	13.07 d	78.3%
11	RL	SPT	7,390 h	3,481 h	3,909 h	92.9%	11.58 d	77.5%
12	RL	SST	7,222 h	3,313 h	3,909 h	95.0%	12.91 d	75.7%
13	SC	EDD	7,463 h	3,560 h	3,903 h	95.2%	12.81 d	78.3%
14	SC	SPT	7,374 h	3,481 h	3,893 h	94.1%	11.43 d	77.4%
15	SC	SST	7,175 h	3,305 h	3,870 h	95.5%	12.74 d	75.3%
16	OQSC	EDD	6,946 h	3,465 h	**3,480 h**	**95.8%**	11.50 d	72.9%
17	OQSC	SPT	6,881 h	3,400 h	3,482 h	95.5%	**10.45 d**	72.2%
18	OQSC	SST	**6,712 h**	**3,227 h**	3,485 h	95.7%	11.51 d	**70.4%**
19	RL	EDD	8,253 h	4,270 h	3,983 h	88.9%	15.72 d	86.1%
20	RL	SPT	8,075 h	4,092 h	3,983 h	88.0%	13.55 d	84.4%
21	RL	SST	7,675 h	3,692 h	3,983 h	93.2%	13.80 d	80.4%
22	SC	EDD	8,192 h	4,271 h	3,921 h	92.2%	15.28 d	85.8%
23	SC	SPT	7,974 h	4,080 h	3,894 h	90.5%	13.14 d	83.6%
24	SC	SST	7,498 h	3,651 h	3,847 h	95.4%	13.28 d	78.6%
25	OQSC	EDD	7,658 h	4,215 h	3,443 h	94.8%	13.24 d	80.3%
26	OQSC	SPT	7,487 h	4,049 h	**3,438 h**	94.0%	**11.77 d**	78.5%
27	OQSC	SST	**7,051 h**	**3,609 h**	3,442 h	**95.7%**	12.09 d	**73.9%**
28	RL	EDD	7,501 h	3,518 h	3,983 h	93.9%	12.91 d	78.6%
29	RL	SPT	7,428 h	3,445 h	3,983 h	92.6%	11.46 d	77.9%
30	RL	SST	7,265 h	3,283 h	3,983 h	94.2%	12.92 d	76.2%
31	SC	EDD	7,368 h	3,519 h	3,849 h	95.5%	12.48 d	77.3%
32	SC	SPT	7,277 h	3,442 h	3,836 h	94.6%	11.13 d	76.3%
33	SC	SST	7,093 h	3,268 h	3,825 h	95.6%	12.50 d	74.4%
34	OQSC	EDD	6,943 h	3,496 h	3,448 h	**95.8%**	11.50 d	72.8%
35	OQSC	SPT	6,876 h	3,430 h	**3,446 h**	95.5%	**10.46 d**	72.1%
36	OQSC	SST	**6,704 h**	**3,254 h**	3,450 h	95.7%	11.55 d	**70.3%**
37	RL	EDD	7,811 h	4,269 h	3,542 h	93.3%	13.80 d	81.8%
38	RL	SPT	7,636 h	4,094 h	3,542 h	92.3%	12.00 d	80.0%
39	RL	SST	7,248 h	3,707 h	3,542 h	95.2%	12.62 d	76.0%
40	SC	EDD	7,813 h	4,271 h	3,541 h	94.6%	13.61 d	81.9%
41	SC	SPT	7,625 h	4,082 h	3,543 h	93.5%	**11.76 d**	80.0%
42	SC	SST	7,218 h	3,676 h	3,542 h	**95.6%**	12.48 d	75.7%
43	OQSC	EDD	7,729 h	4,195 h	3,535 h	94.7%	13.18 d	81.1%
44	OQSC	SPT	7,572 h	4,038 h	3,534 h	94.2%	11.94 d	79.4%
45	OQSC	SST	**7,100 h**	**3,566 h**	**3,534 h**	95.5%	11.98 d	**74.5%**
46	RL	EDD	7,060 h	3,518 h	3,542 h	95.4%	11.81 d	74.0%
47	RL	SPT	6,987 h	3,446 h	3,542 h	94.8%	10.50 d	73.3%
48	RL	SST	6,830 h	3,289 h	3,542 h	95.5%	11.96 d	71.6%
49	SC	EDD	7,060 h	3,518 h	3,541 h	**95.7%**	11.61 d	74.1%
50	SC	SPT	6,984 h	3,443 h	3,541 h	95.3%	**10.33 d**	73.3%
51	SC	SST	6,824 h	3,280 h	3,544 h	95.7%	11.81 d	71.6%
52	OQSC	EDD	7,020 h	3,485 h	**3,534 h**	95.7%	11.53 d	73.6%
53	OQSC	SPT	6,958 h	3,423 h	3,534 h	95.4%	10.67 d	73.0%
54	OQSC	SST	**6,768 h**	**3,233 h**	3,535 h	95.6%	11.50 d	**71.0%**
55	RL	EDD	8,033 h	4,270 h	3,764 h	91.4%	14.73 d	84.1%
56	RL	SPT	7,860 h	4,096 h	3,764 h	90.4%	12.73 d	82.4%
57	RL	SST	7,465 h	3,701 h	3,764 h	94.7%	13.19 d	78.3%
58	SC	EDD	8,016 h	4,269 h	3,747 h	93.6%	14.40 d	84.0%
59	SC	SPT	7,819 h	4,084 h	3,736 h	92.4%	12.40 d	82.0%
60	SC	SST	7,390 h	3,667 h	3,723 h	95.5%	12.90 d	77.5%

61	OQSC	EDD	7,693 h	4,206 h	**3,486 h**	94.9%	13.25 d	80.7%
62	OQSC	SPT	7,530 h	4,043 h	3,487 h	94.3%	**11.81 d**	79.0%
63	OQSC	SST	**7,093 h**	**3,595 h**	3,498 h	**95.6%**	12.10 d	**74.4%**
73	RL	EDD	7,723 h	4,182 h	3,542 h	91.2%	14.18 d	80.8%
74	RL	SPT	7,561 h	4,019 h	3,542 h	90.6%	12.33 d	79.2%
75	RL	SST	7,193 h	3,651 h	3,542 h	94.7%	12.80 d	75.4%
76	SC	EDD	7,704 h	4,162 h	3,542 h	95.0%	13.28 d	80.8%
77	SC	SPT	7,530 h	3,989 h	3,541 h	93.8%	11.53 d	79.0%
78	SC	SST	7,150 h	3,608 h	3,542 h	**95.7%**	12.36 d	75.0%
79	OQSC	EDD	7,544 h	4,009 h	3,534 h	95.2%	12.59 d	79.1%
80	OQSC	SPT	7,397 h	3,862 h	3,535 h	94.7%	**11.46 d**	77.6%
81	OQSC	SST	**6,956 h**	**3,421 h**	**3,534 h**	95.6%	11.66 d	**73.0%**
82	RL	EDD	7,042 h	3,501 h	3,542 h	94.8%	12.03 d	73.9%
83	RL	SPT	6,969 h	3,427 h	3,542 h	94.1%	10.69 d	73.1%
84	RL	SST	6,811 h	3,269 h	3,542 h	95.3%	12.11 d	71.4%
85	SC	EDD	7,027 h	3,484 h	3,543 h	**95.8%**	11.57 d	73.7%
86	SC	SPT	6,951 h	3,409 h	3,543 h	95.4%	**10.28 d**	72.9%
87	SC	SST	6,788 h	3,245 h	3,542 h	95.7%	11.77 d	71.2%
88	OQSC	EDD	6,911 h	3,376 h	3,534 h	95.7%	11.27 d	72.5%
89	OQSC	SPT	6,850 h	3,315 h	3,535 h	95.4%	10.44 d	71.9%
90	OQSC	SST	**6,665 h**	**3,130 h**	**3,534 h**	95.6%	11.28 d	**69.9%**
91	RL	EDD	7,945 h	4,221 h	3,724 h	89.2%	15.39 d	82.9%
92	RL	SPT	7,780 h	4,056 h	3,724 h	88.7%	13.19 d	81.4%
93	RL	SST	7,397 h	3,673 h	3,724 h	94.5%	13.30 d	77.6%
94	SC	EDD	7,988 h	4,207 h	3,781 h	93.5%	14.34 d	83.8%
95	SC	SPT	7,800 h	4,033 h	3,767 h	92.3%	12.34 d	81.8%
96	SC	SST	7,386 h	3,638 h	3,749 h	95.4%	12.99 d	77.5%
97	OQSC	EDD	7,578 h	4,067 h	3,511 h	95.1%	12.83 d	79.5%
98	OQSC	SPT	7,425 h	3,915 h	**3,511 h**	94.5%	**11.50 d**	77.9%
99	OQSC	SST	**7,018 h**	**3,497 h**	3,521 h	**95.7%**	11.88 d	**73.6%**
100	RL	EDD	7,258 h	3,533 h	3,724 h	94.5%	12.57 d	76.1%
101	RL	SPT	7,183 h	3,459 h	3,724 h	93.5%	11.13 d	75.3%
102	RL	SST	7,019 h	3,295 h	3,724 h	95.2%	12.52 d	73.6%
103	SC	EDD	7,277 h	3,526 h	3,751 h	95.6%	12.16 d	76.3%
104	SC	SPT	7,204 h	3,451 h	3,753 h	94.9%	10.81 d	75.6%
105	SC	SST	7,023 h	3,281 h	3,742 h	95.7%	12.29 d	73.7%
106	OQSC	EDD	6,946 h	3,426 h	**3,520 h**	**95.8%**	11.40 d	72.9%
107	OQSC	SPT	6,884 h	3,363 h	3,521 h	95.4%	**10.41 d**	72.2%
108	OQSC	SST	**6,712 h**	**3,186 h**	3,526 h	95.7%	11.44 d	**70.4%**

Best values are highlighted in bold; mean values calculated from 100 replications.

STATEMENT OF CERTIFICATION

I hereby confirm that this dissertation constitutes my own work, produced without aid and support from persons and/or materials other than the ones listed. All used sources are indicated as direct or indirect quotations. Quotation marks indicate direct language from another author. Appropriate credit is given where I have used ideas, expressions or text from another public or non-public source. The thesis in this form or in any other form has not been submitted to an examination body.

Michael Hamann

Frankfurt am Main, 18th of March 2015