THESIS FOR THE DEGREE OF LICENTIATE OF ENGINEERING

On the Optimization of Schedules of a Multitask Production Cell

Karin Thörnblad

CHALMERS | UNIVERSITY OF GOTHENBURG

Volvo Aero Corporation
Department of Logistics Development
SE-461 81 Trollhättan, Sweden

and

Department of Mathematical Sciences, Chalmers University of Technology
Department of Mathematical Sciences, University of Gothenburg
SE-412 96 Göteborg, Sweden

Göteborg, 2011

This work was supported by:
Volvo Aero, Vetenskapsrådet (The Swedish Research Council), and NFFP (National Aviation Engineering Research Programme).

Cover illustration: An example of an optimal schedule for the resources in the multitask production cell; see Figure 9 in the thesis.

On the Optimization of Schedules of a Multitask Production Cell
Karin Thörnblad

©Karin Thörnblad, 2011

NO 2011:19
ISSN 1652-9715
Department of Mathematical Sciences, Chalmers University of Technology
Department of Mathematical Sciences, University of Gothenburg
SE-412 96 Göteborg
Sweden
Telephone +46 (0)31 772 1000

Printed in Göteborg, Sweden 2011

On the Optimization of Schedules of a Multitask Production Cell

Karin Thörnblad

Department of Logistics Development, Volvo Aero Corporation

and

Department of Mathematical Sciences, Chalmers University of Technology
Department of Mathematical Sciences, University of Gothenburg

Abstract

Volvo Aero has invested in a complex production cell containing a set of multipurpose machines. The problem of finding optimal schedules for this multitask cell is a complex combinatorial optimization problem which is recognized as a flexible job shop problem.

This thesis proposes an approach to find such schedules using mathematical optimization. The mathematical models developed so far are presented together with a study of their interrelations, both from a computational and from a modelling perspective. One result of the study is that one of these models, a time-indexed model with nail variables, outperforms all the others presented. To our knowledge, this model is the first for the flexible job shop problem using this type of variables.

In order to reduce the number of variables in the time-indexed models, a heuristic has been developed which finds an upper bound on the optimal value of the makespan.

Computational results are presented for several variants of the time-indexed model and the engineer's model, the latter belonging to a family of models widely used in job shop scheduling. The objective employed for the computations is the minimization of a weighted sum of the total tardiness and the sum of job completion times.

A comparison is made between optimal schedules emanating from the time-indexed model and schedules resulting from the use of three well-known dispatching rules, using the data from 21 real production scenarios. The tardiness and the sum of job completion times are on average 6–22% larger in the schedules resulting from the use of the dispatching rules compared to those obtained in the optimal schedules. The first appended paper provides the results from a similar comparison for a number of scenarios, constructed by using data emanating from the multitask cell.

The computational complexity of the problem of scheduling the multitask cell is also investigated. The second appended paper contains a proof of a complexity result for a related scheduling problem, namely flow-shop scheduling with deteriorating jobs.

This thesis has been written in close cooperation with the Department of Logistics Development at Volvo Aero Corporation.

Keywords: mathematical optimization; flexible job shop scheduling; mixed integer linear programming (MILP); complexity analysis; mathematical modelling; production planning; multi-purpose machine; dispatching rule; priority function; total flow time; total tardiness; release date; due date

Appended papers:

Paper I: *Optimization of schedules for a multitask production cell*, Proceedings of 22nd Nofoma conference, Kolding, Denmark, 2010 (with Ann-Brith Strömberg, Torgny Almgren, and Michael Patriksson).

Paper II: *A note on the complexity of flow-shop scheduling with deteriorating jobs*, Published in Discrete Applied Mathematics, 159 (2011), pp. 251–253 (with Michael Patriksson).

Other publications, not included in this thesis:

A comparison of schedules resulting from priority rules and mathematical optimization for a real production cell, Proceedings of PLANs forsknings- och tillämpningskonferens, Skövde, Sweden, 2010 (with Linea Kjellsdotter Ivert).

Mathematical modelling of a real flexible job shop in aero engine component manufacturing, Proceedings of 10th Workshop on Models and Algorithms for Planning and Scheduling Problems, Nymburk, Czech Republic, 2011 (with Michael Patriksson, Ann-Brith Strömberg, and Torgny Almgren).

Optimering av scheman för en verklig produktionscell: tidsdiskretisering reducerar lösningstiden utan att lösningarnas kvalitet försämras, Proceedings of PLANs forsknings- och tillämpningskonferens, Norrköping, Sweden, 2011 (with Torgny Almgren, Ann-Brith Strömberg, and Michael Patriksson).

Acknowledgements

First of all, I would like to express my gratitude towards my industrial supervisor Torgny Almgren for taking the initiative to this project and for his constant support and encouragement. I have been fortunate to have two nice academic supervisors Michael Patriksson and Ann-Brith Strömberg. I thank both of you for always taking the time for discussions and providing advice and for your wise guidance in this interesting subject.

I would also like to thank my colleagues at the department of Logistics Development at Volvo Aero (Fredrik, Sture, Ulla, Maria, and Joakim to name but a few) for your support and willingness to share experience, and for creating a good working environment. I would like to thank the personnel working with the multitask cell (especially Nicklas Gödebu, Christina Kann, and Martin Lundstedt) as well as the project leader of the former multitask project, Fredrik Johansson, for your useful information and for putting up with all my questions. I would further like to thank Tomas Lindsta, manager of the production of high volume products, for your encouraging support as a member of the steering committee.

A special thanks goes to Thomas Ericsson at the Department of Mathematical Sciences at Chalmers for your support with problems encountered during the computational testing, and all the members of the optimization group and especially Adam Wojciechowski for interesting discussions. Yet another special thanks to Linea Kjellsdotter Ivert at the Department of Technology Management and Economics (Division of Logistics and Transportation) at Chalmers for nice collaboration and for being a good friend.

Furthermore, I would like to gratefully acknowledge the financial support from Volvo Aero, Vetenskapsrådet (The Swedish Research Council), and NFFP (National Aviation Engineering Research Programme).

Finally, I would like to express my thanks to my family. My parents and parents-in-law for your constant support, Gabriel, for being an excellent IT consultant as well as a wonderful supportive husband, and my three boys for bringing happiness and laughter.

Karin Thörnblad
Trollhättan, August 2011

Contents

1 Introduction

1.1 Background

Volvo Aero has invested in a complex production cell containing a set of multi-purpose machines, with the aim to decrease product costs, shorten lead times, and increase the quality level and delivery precision. The planning and control of this so-called *multitask cell* result in a complex combinatorial optimization problem, which needs to be solved in a reasonable amount of time. The control system of the multitask cell contains a built-in scheduling algorithm, which is based on a simple priority function. In the master's thesis [28], it was shown that this algorithm is not sufficiently efficient, as it is not adapted to the production of complex structures such as the aircraft engine components, which is the case at Volvo Aero. Therefore, the built-in scheduling algorithm is not in use and the production of components in the multitask cell is manually planned at present. This inferiority of the existing decision support leads to unnecessarily long lead times and an inefficient use of the resources.

1.2 Aims and objectives

The overall aim of the research project is to contribute to the goal of enabling the construction of optimal, or near-optimal, schedules for multi-purpose production cells, similar to the Volvo Aero multitask cell. An algorithm which is fast and appropriate then needs to be developed for the problem of scheduling the multitask cell, since the conditions are unceasingly changing with new jobs continuously arriving at the queue.

This thesis focuses primarily on finding a mathematical optimization model that can be used for solving the problem of scheduling the processes in the multitask cell in a reasonable computation time. This goal is achieved by

- the development of a number of mixed integer linear programming (MILP) models for the scheduling problem at hand;
- the study of these models' interrelations, from both a computational and a modelling perspective;
- a discussion on appropriate objective functions and their impact on the logistic performance of the production cell.

1.3 Limitations

This thesis is limited to the study of the scheduling of the machine resources in the multitask cell. The number of staff in operation in the multitask cell is assumed to be sufficient for performing the manual work associated with all the operations scheduled, and their scheduling is therefore not included in the MILP models. Other areas not taken into account in this thesis are the maintenance planning of the multitask cell, and the availability of machining tools. All of these areas are subjects for future research.

1.4 Outline

In this thesis, we present all the MILP models of the scheduling problem that have been developed so far, together with computational results. Section 2.1 gives an overview of the planning situation at Volvo Aero, where the supply chain, the products, and the multitask cell are described.

In Section 3, a literature overview of possible approaches to problems similar to the problem of scheduling the multitask cell is given. First, the perspective of operations research is presented; both exact approaches—as *mathematical optimization*—and approximate approaches—as the use of *metaheuristics*—are described. The section ends with a description of the perspective of logistics. This is then followed by Section 4, dealing with the complexity of scheduling problems. First, a motivation for the proof presented in [53] (Paper II) is given. The problem studied in Paper II is a so-called flow shop scheduling problem with deteriorating jobs.

The main section in this thesis is Section 5, where the MILP models for the problem of scheduling the multitask cell are presented. A discussion regarding different objective functions is included. This section ends with a description of a greedy heuristic, developed with the aim of obtaining good feasible schedules. The heuristic also provides parameter values for use in the MILP models to ensure reduced computation times.

Computational results are presented in Section 6. It is clear that a time-indexed model with so-called *nail variables* outperforms all the other models developed so far. To our knowledge, this model is the first model for the flexible job shop problem which uses this type of variables. Since the time-indexed model performs best, it is used in most of the computational tests performed. The consequences of the length of the discretization interval for the time-indexed model is investigated, and the model is compared with the so-called *engineer's model*, being of a common type used for the formulation of similar scheduling problems. A comparison between optimal schedules emanating from the time-indexed model and the schedules resulting from the use of three well-known dispatching rules is also given. The impact of the choice of a parameter for the engineer's model on the computation times is presented as the last test in this section. All data used for the tests described in Section 6 consist of real production scenarios from the multitask cell at Volvo Aero.

In Sections 7 and 8, conclusions and proposed areas for future research are found, and Section 9 contains a summary of the appended papers.

2 The planning situation at Volvo Aero

2.1 The company Volvo Aero

Volvo Aero is a small and distinguished member of the Volvo Group, since the company manufactures mainly aero engine and aerospace components, i.e., not the final products as most of the members of the Volvo Group. An overview of the supply chain for the civil aero engine components manufactured by Volvo Aero is illustrated in Figure 1. Volvo Aero's main production sites are situated in Trollhättan, Sweden,

in Kongsberg, Norway, and in Newington, Connecticut, USA. Since the main production is in the aero engine industry, the quality requirements are high. Therefore, the processing machines in the production are often very large and expensive, while the product volumes are small.

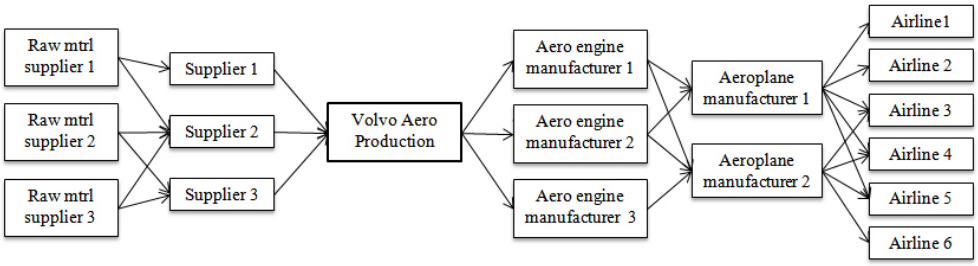

Figure 1: *An overview of the supply chain for the civil aero engine components produced by Volvo Aero.*

The planning and control of the supply chain is affected by the fact that the production of aero engine components are subject to flight safety regulations issued by flight safety authorities. As an example, all suppliers have to be approved by the authorities, and Volvo Aero has to keep track of which mine the raw material for each final product comes from. The operations performed by Volvo Aero also has to be approved for each processing machine.

2.2 The products

The products currently processed in the multitask cell are mainly combustor structures, such as diffuser cases and compressor rear frames; see Figure 2. The diffuser cases and the compressor rear frames are integrated cast structures in a nickel-based material. The manufacturing process is a complex combination of turning, milling, and drilling operations, which are mixed with NDT-procedures (Non Destructive Testing), and in some cases welding and heat treatment of the products.

2.3 The multitask cell

The multitask cell contains ten processing resources along with a central tool storage and an input/output conveyor; see Figure 3. The production cell is supposed to carry out a large variety of jobs, since five of the cell's resources are multi-purpose machines that are able to process three different types of operations (turning, milling, and drilling).

The multitask cell was built with the aim of achieving a high degree of machine utilization, reducing product lead times and being flexible with regard to both the product mix and the type of processing. Presently, the multitask cell is executing about 30 different operations on eight different products. Each part typically visits

Figure 2: *The compressor rear frame is one part processed in the multitask cell.*

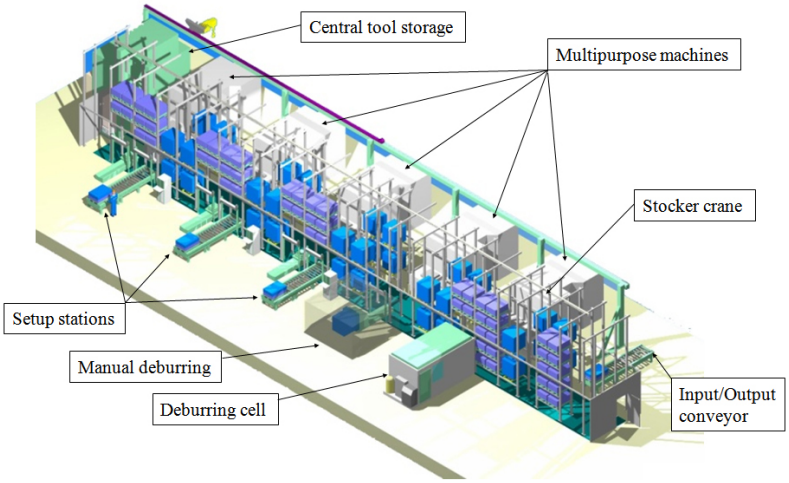

Figure 3: *An overview of the multitask cell.*

the multitask cell multiple times on its way to completion; see Figure 4. One such visit to the multitask cell is called a *job* in this thesis.

The parts that are ready to be processed in the multitask cell are those that are checked-in at the input conveyor but not yet put into a fixture at a set-up station. After the check-in, the parts are transported by a stocker crane to special storage locations inside the multitask cell. There are also storage areas inside the cell for parts that are already mounted into fixtures. Each part to be processed in the multitask cell follows a specific routing through the set of resources, which consists of three to five so-called *route operations*, starting and ending by the mounting and removing of fixtures at one of the three set-up stations; see Figure 5. The second route operation in this routing is always the processing in one of the multitask machines. Some parts

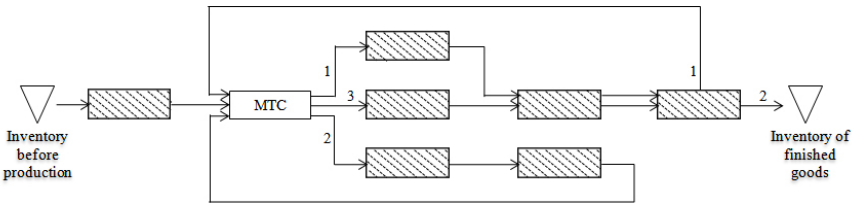

Figure 4: *A typical path through the factory for a part on its way to completion, with multiple visits in the multitask cell (MTC). The striped boxes represent operations performed outside the multitask cell and the numbers indicate the order in which the routes are to be taken.*

need manual and/or robot deburring. See Table 1 in Section 5.1.2 for a list of all possible paths through the multitask cell.

Figure 5: *A part mounted into a fixture at a set-up station in the multitask cell.*

2.4 The production planning of the multitask cell

The current production planning of the multitask cell is described in [55] (Paper I). In the planning system at Volvo Aero—the so-called ERP system (Enterprise Resource Planning)—two reports are commonly used which propose the job priorities. One report is based on the Earliest Due Date (EDD) priority rule and the other is based on the First In First Out (FIFO) priority rule. The prerequisites for the logistics of the multitask cell has been studied in the master's thesis [41], in which the current detail planning is described. Currently, the planning is manually performed by a detail planner, with the help of the EDD-list and other priorities based on the current logistic situation. The decision on which job to schedule on which machine is currently made by the manager of the multitask cell together with the detail planner. As each

job is only allowed to be processed in a subset of the multitask machines, this is not a simple task: Even though the processing machines are of the same kind, they are not identical, and some of the machines have been altered in order to be able to process some large parts. Also, some machines yield a result which are better repetitive for certain jobs having requirements on extremely small tolerances on e.g. rotundity and thickness, due to flight safety issues. These are two of the reasons why some jobs can only be processed in some machines. Another reason is that some products consist of a different material than the rest, and the price paid by scrap dealers is much lower for mixed metal chips (the metal scrap from the turning process) than for sorted materials. This reason differs from the others, since it can be eliminated by either scheduling a cleaning operation in a machine between any two operations comprising parts of different materials, or selling the metal chips at the lower price. As a consequence of the low product volumes and the expensive machines, which are difficult to move, most of the parts take different routes in the factory, such as the one illustrated in Figure 4. Getting an overview of the planning situation with regard to incoming jobs is therefore difficult for a manual planner.

The problem described—to schedule the operations in the multitask cell—can be classified as *a flexible job shop problem* in the operations research nomenclature. In the following section, this problem is described in more general terms.

3 Job shop scheduling: A subject orientation

The job shop scheduling problem is one of three classic *shop scheduling* problems, the other two being the *flow shop* and the *open shop* scheduling problems; see [8]. The *job shop problem* is defined as that to find the optimal sequences of a given set of jobs on a given set of machines. Each job consists of a number of operations, which must be processed in a given order. The constraints indicating that one operation must precede another are called *precedence constraints*. Associated with each operation is a job, a machine, and a processing time. The flexible job shop problem is an extension of the job shop problem, in the sense that each operation may be scheduled in more than oneof the machines; see [3].

3.1 The perspective of operations research

Operations research is an interdisciplinary science that focuses on the efficient use of technology in order to arrive at optimal, or near-optimal, solutions to complex decision-making problems; see e.g. [50].

3.1.1 Mathematical optimization

The first *integer programming* formulations of job-shop problems were formulated in the late 50's by Manne ([33]), Wagner ([58]), and Bowman ([7]). These formulations are all different, since they model the dimension of time in three different ways, which in turn reflect their respective definitions of the binary decision variables.

In [33], Manne studied the problem of sequencing jobs with precedence constraints on a single machine. Here, the jobs' starting times are represented by integer variables (which may be respresented by continuous variables, but here they are considered integral since Manne applied integer rather than mixed integer programming). The decision variables are defined as y_{jq} equals 1, if job j precedes job q, and 0 otherwise. There are many examples in the operations research literature of models from the so-called *Manne family*, i.e., models that use this type of variables. Some recent formulations of models for the flexible job shop problem within the Manne family are found in [40, 14, 44] and [63]. The only model presented in the section of job shop scheduling in the Wiley Encyclopedia of Operations Research and Management Science 2011 ([42]) is a Manne family model. In Section 5.2, a model in the Manne family is formulated for the problem of scheduling the multitask cell.

Wagner ([58]) also considers the classical job shop problem, although calling it the *machine-scheduling problem*. Since here solely the ordering of the jobs on each machine is considered, the time dimension is implicitly treated. In this model, each decision variable equals 1 if the corresponding job is scheduled in a specific resource at a specific order-position, and 0 otherwise. In [46], the authors have compared models of the Manne family with models of the so-called Wagner family for the regular *permutation flow shop problem* to minimize the *makespan*, i.e., to minimize the time at which all jobs are completed. The flow shop problem is a special case of the job shop problem, as each job is required to follow exactly the same processing sequence across all machines. In the permutation flow shop, the processing order of the jobs is the same in all of the resources. Since, in the models of the Wagner family, the decision variables are related to the ordering of jobs, these models seem to be well adapted for the permutation flow shop problem, according to the computational results presented in [46].

A third way of modelling time is considered by Bowman, in [7], who denotes the classical job shop problem by the *schedule-sequencing problem*. The planning period is divided into an integer number of time periods with equal length. The decision variables used in this article equal 1 if the corresponding job is processed by a specific resource during a specific time period, and 0 otherwise. A formulation of scheduling problems to minimize total earliness/tardiness on parallel machines using this type of variables is found in [29]. In Section 5.4.5 we formulate a model for the scheduling of the multitask cell using this type of variables, which we call *plateau variables*.

Another time-indexed formulation using decision variables which equal 1 if the corresponding job starts in a specific discrete time period, and 0 otherwise (called *nail variables* in this thesis), is found in [45], and more generally in [60] for production planning and scheduling problems. The formulations using variables for each discrete time period lead to very large models in terms of numbers of both constraints and variables, but formulations using nail variables typically yield better so-called *lower bounds* (see Section 4.3) than other MILP formulations of scheduling problems; see [56]. A time-indexed model using nail variables is formulated in Section 5.4.4 for the problem of scheduling the operations in the multitask cell. This is, to our knowledge, the first model published for this kind of a flexible job shop, and it outperforms substantially all the other models developed and tested within the project regarding

the computation time required to solve the problems and also regarding which sizes of instances they are able to solve within reasonable computation time.

The objective that is the most studied for scheduling problems is the minimization of makespan; see [27]. Other common objectives are related to the jobs' earliness/tardiness and completion times, and/or inventory holding costs associated with the jobs. A model with a time-indexed formulation, in which the time-indexed variables are integral and not binary, as in the time-indexed models mentioned above, is presented in [19]. The objective function in [19] is to minimize the costs associated with in-process inventory, earliness/tardiness and costs associated with orders not fully completed at the end of the scheduling horizon. See Section 5.1.5 for a discussion on different objective functions and their suitability for the planning of operations in the multitask cell.

3.1.2 Metaheuristics

Integer linear programming models formulated for scheduling problems in the period of 1959–1990 are correct, but the computers were then able to solve only very small instances, and it was not possible to employ an exact mathematical optimization on instances of sizes relevant for real applications. Therefore, a lot of research was concentrated on obtaining an approximate solution of job shop problems by the application of heuristic methods; see [27] for an historical overview. According to [27], the development of so-called *metaheuristics* for job shop scheduling started with the development of the so-called *shifting bottleneck heuristic* by Adams et al. in 1988 ([1]). Since then, many metaheuristics have been proposed for finding schedules for job shops, among these are *simulated annealing*, *tabu search*, *genetic algorithms*, and *ant colony optimization*; see, e.g., [14, 62, 36] and [21] for some recently proposed hybrid approximation algorithms which are based on the metaheuristics mentioned above. A genetic algorithm for the job shop problem with the tardiness objective is found in [35]. The shifting bottleneck heuristic is still popular; see [37] for a variant for the flexible job shop, and [10] which combines the shifting bottleneck heuristic with a MILP model. Other recent work on the flexible job shop problem are [12, 59, 23] and [16], of which [59] also considers preventive maintenance activities.

The major disadvantage with metaheuristics is that there is often no other stopping criteria than a maximum allowed number of iterations, or limited computation time. Hence, no quality measure of the solution is provided as for the case of applying mathematical optimization. Therefore, the quality of the solutions obtained become unknown. Another weakness of the approximation algorithms is that often there are several parameters that have to be carefully selected in order for the algorithms to produce good solutions ([27]). Comparisons between different heuristics often become invalid since they are usually performed from an unbalanced perspective: Typically, the approaches taken are presented with a very well calibrated set of parameter values, while other approaches included in the study for comparisons, are employed using a standard parameter setting. Hence, a fair comparison is often not achieved.

Arostegui et al. ([2]) have made an attempt to make a fair comparison between

tabu search, simulated annealing, and genetic algorithms for three variants of the facility location problem (FLP). FLP is a complex combinatorial optimization problem, which is as hard to solve as the job shop problem, according to the authors. In this article, tabu search seems to be the best algorithm, with good performance for all three variants, while both simulated annealing and genetic algorithms sometimes get stuck in worse local minima, and hence perform badly for some variants.

Other approaches to find good solutions to the job shop problem is to combine simulation models with meta-heuristics; see, e.g., [49], which is especially interesting since the case study in that article is the Volvo Aero multitask cell considered in this thesis. A more general description of this method is found in [48]. Yet another approach is the use of *constraint programming*; see, e.g., [6]. In [4], a hybrid algorithm which combines a tabu search algorithm with constraint programming is proposed for the job shop problem and represents, according to the authors, the first occasion in which a constraint programming algorithm obtains a performance that is competitive with the best so-called *local search algorithms* (e.g., tabu search is a local search algorithm).

3.2 The perspective of logistics

According to ([11]) the discipline of *logistics* is the management of the flow of goods and services between the point of origin and the point of consumption in order to meet the requirements of customers. The term *production logistics* is used to describe logistic processes in-house a factory [57]. The purpose of production logistics is to ensure that each machine and workstation is being fed with the right product of the correct quantity and quality at the right time. This corresponds approximately to the goal of the job shop problem, and although quality is seldom explicitly dealt within the scheduling community of operations research, the two disciplines have a lot in common.

Scientific articles related to production planning written in the view of production logistics typically do not have an as quantitative perspective as those written in the view of operations research. For example, Stoop and Wiers ([47]) list a number of causes for automated scheduling techniques not functioning in practice. Such causes can be disturbances, machine breakdowns, rush orders, or the unavailability of raw materials. Another cause can be personnel overruling the scheduling tools, believing that they can outperform the technique. It is hard to prove to personnel responsible for operations planning the advantages of an automated scheduling technique, since the quality of a schedule is usually very hard to assess.

In [22], Herrmann describes production scheduling from a logistics point of view, and with a hierarchical view of the organizational, decision-making, and problem-solving perspectives of production scheduling. In this view, the problem of finding optimal schedules equals production scheduling from the problem-solving perspective. In the decision-making perspective, the schedulers also perform tasks such as crisis identification, and make decisions in order to avoid possible future trouble. The broadest perspective is the organizational perspective, which takes the whole organization around the production into account. The author points out that pro-

duction scheduling is not just an optimization problem, but a complex system of information flow, decision-making, and problem-solving.

Although job shop scheduling is complex, feasible schedules are found every day all around the world for all the various real applications that need to be scheduled. The most common means for constructing a schedule is to make use of one of many *dispatching rules*, proposed by, e.g. [5] and [24]. A dispatching rule (also called *priority rule*) is classified as a *static rule* if the priority value once calculated remains the same throughout the planning horizon; it is called a *dynamic rule* if the priority value calculated at a certain point in time differs from that calculated at a later time. The most common static rules are the so-called *shortest–processing–time* (SPT), *earliest–due–date* (EDD) and *first–in–first–out* (FIFO); a description of these and other dispatching rules can be found in e.g. [25].

In this work, the schedules generated by static dispatching rules applied to data from real instances from the multitask cell have been compared with the resulting schedules emanating from the mathematical optimization models presented in Sections 5.3 and 5.4.4, for the same set of instances. The results from these comparisons are reported in Section 6, in [55] (Paper I in this thesis), and in [52].

The priority lists resulting from the use of dispatching rules can be generated by commercial planning softwares, henceforth called ERP (Enterprise Resource Planning) systems. Other commercial planning softwares include the more sofisticated APS (Advanced Planning and Scheduling) systems, which are often integrated with the ERP systems; see e.g. [25, pp. 147–149]. APS systems are, however, usually designed to cover all the different classes of scheduling problems—e.g. job shop, flow shop, open shop, assembly line, and continuous flow (e.g. process industry) scheduling problems—and, therefore, they typically offer only a small number of generic scheduling algorithms. The disadvantage of the commercial planning softwares is that a built-in generic scheduling algorithm probably most often results in worse schedules than does a tailored scheduling algorithm for a certain problem class, such as, e.g., flexible job shop ([31]).

4 On the complexity of scheduling problems

Scheduling problems such as flow shop, job shop, and open shop problems have been known to be NP-complete since the mid seventies ([18]). An NP-complete or NP-hard problem is such that no algorithm exists that in polynomial time is able to solve all possible instances of the problem ([17]). Hence, the solution time risks to increase exponentially with the number of jobs. The difference between the notions NP-hard and NP-complete is that NP-hard problems are at least as hard as NP-complete problems; see [17, p. 109]. In [18], the flow shop problem with the objective of minimizing makespan was proven to be NP-complete for instances with three or more machines. It was also shown that the flow shop problem with the objective of minimizing the mean flow time, i.e., the sum of the jobs' completion times, and the job shop problem with the makespan criterion, are NP-complete for instances with two or more machines.

4.1 On the complexity of flow shop problems with deteriorating jobs

In [53] (Paper II in this thesis), we present a proof of NP-hardness for flow shop problems for which the processing time of each job is equal to a deterioration rate times the job's starting time. It is a note on an article written by Mosheiov in 2002 ([38]), in which the proof regarding the complexity of a flow shop problem with deteriorating jobs is incorrect. We provide a correct proof of the statement that this problem is NP-hard for instances with three or more machines, when the objective is to minimize the makespan.

During the work with this note, it came to our notice that a correct proof, for an even stronger result, was published already in 1996 in a Russian journal; see Kononov [30]. This proof, originally given in Russian, is summarized in [53] (Paper II). However, we noted, while tracing the citation history of the two proofs by Mosheiov and Kononov, respectively, that the incorrect proof by Mosheiov was by far the one most often cited. Hence, we considered a correction of the proof by Mosheiov being necessary and quite timely.

4.2 On the complexity of flexible job shop problems

We consider here a subproblem of the flexible job shop problem studied in this thesis, namely scheduling only the five processing machines in the multitask cell. We further assume that all parts to be scheduled are available from the start, i.e., all release dates are set to zero for all jobs, and that there are no precedence constraints between the jobs. Moreover, we consider the objective of minimizing the weighted sum of completion times.

Using the $\alpha|\beta|\gamma$-notation introduced in [20], this problem can then be described as *F MPM5*|*stages* $= 1|\sum w_i C_i$. The first field, α, specifies the machine environment, where *F*, *J*, and *O* denote flow shop, job shop and open shop, respectively. These notations can be combined with, e.g. *P* and *MPM*, which denote identical parallell machines, and multi-purpose machines, respectively, and a number k, representing the number of machines. The subproblem considered above has only one *stage*, that is, each job consists of one operation. Therefore, the first element of α can be set to any of *F*, *J*, and *O*. We have chosen $\alpha =$ "*F MPM5*" for convenience.

The β-field is used to describe the job characteristics. There can be at most six elements in this field, for example *prec*, r_i, or d_i for precedence constraints, release dates and due dates, respectively, to name a few elements relevant to the problems studied in this thesis. In the subproblem considered here, these job characteristics are not relevant due to the assumptions made above. We therefore set β equal to *stages*$= 1$.

The last field in this notation is the γ-field which is used to describe the optimality criterion. The most common objective functions are the minimization of the makespan ($\gamma = C_{\max}$), total completion times ($\gamma = \sum C_i$) and weighted completion times ($\gamma = \sum w_i C_i$). Note that the indices on the $\sum$-symbol are skipped in this notation. Other notations of interest are E_i and T_i which denote earliness and tardiness,

respectively. For further explanation of the $\alpha|\beta|\gamma$-notation; see e.g. [8].

The considered subproblem (*F MPM*5$|stages = 1|\sum w_iC_i$) of the flexible job shop problem is a generalization of the problem *P*5$||\sum w_iC_i$, i.e., the problem of scheduling five parallell machines with the weighted completion times criterion; see [9]. Since the problem $Pk||\sum w_iC_i$ has been shown to be NP-hard for $k \geq 2$ ([17]), it follows that the subproblem considered (*F MPM5*$|stages = 1|\sum w_iC_i$) is NP-hard ($k = 5$). Since this subproblem is a special case of the problem studied in this thesis, we can conclude that the problem of scheduling the multitask cell is NP-hard.

4.3 Weak and strong formulations

Since we are dealing with NP-hard problems the computation times may become very long, as mentioned in the beginning of this section, and hence the choices of both the solution algorithm and the problem formulation are of great importance. In this thesis we have chosen to use a state-of-the-art optimization software, and let this software make the choice of solution algorithm. Instead, we made an effort to develop good mixed integer linear programming (MILP) models for the problem of scheduling the multitask cell. The MILP models are formulated with both integer (or binary) and continuous variables, and with all the relations between the variables in the objective and constraints being linear ([39]). For linear programs (LP), in which all variables are continuous, an optimal solution, if it exists, is found at an extreme point of the feasible region (the polyhedron defined by the linear relations between the continuous variables). This is, however, not the case for MILP models, since the extreme points may contain fractional variable values. A lower bound on the value of the optimal MILP solution is found when solving the so-called *LP relaxation* of the problem, i.e. when relaxing all integer and/or binary constraints on the variables. The constraints formulated to describe the feasible region of a MILP model are called *strong*, if the objective value of the corresponding LP relaxation is close to that of the optimal solution; it is called *weak*, if the constraints in the LP relaxation yield a solution whose objective value is far from that of the corresponding optimal solution. An example is given in 2D in Figure 6.

In the example in Figure 6, the feasible set of the MILP model is illustrated by small filled circles. Both variables are subject to non-negativity constraints, and there are two sets of the remaining constraints corresponding to a weak and a strong formulation, respectively, of the MILP model. The constraints in the weak formulation are illustrated with solid lines, and those in the strong formulation by dashed lines. The constraint marked with (1) in the figure is valid for both formulations. The arrow indicates the direction of decreasing objective function values. The optimal solution of the MILP model equals that of the LP relaxation of the strong formulation, i.e., the intersection of the two dashed lines. The LP relaxation of the weak formulation yields an optimal solution with fractional variable values, i.e., the intersection of the lines marked (1) and (2) in Figure 6.

With the help of so-called strong valid inequalities, it is possible to cut off fractional solutions from the feasible region. The gap between the optimal value and the optimal value of the corresponding continuous relaxation (considering the min-

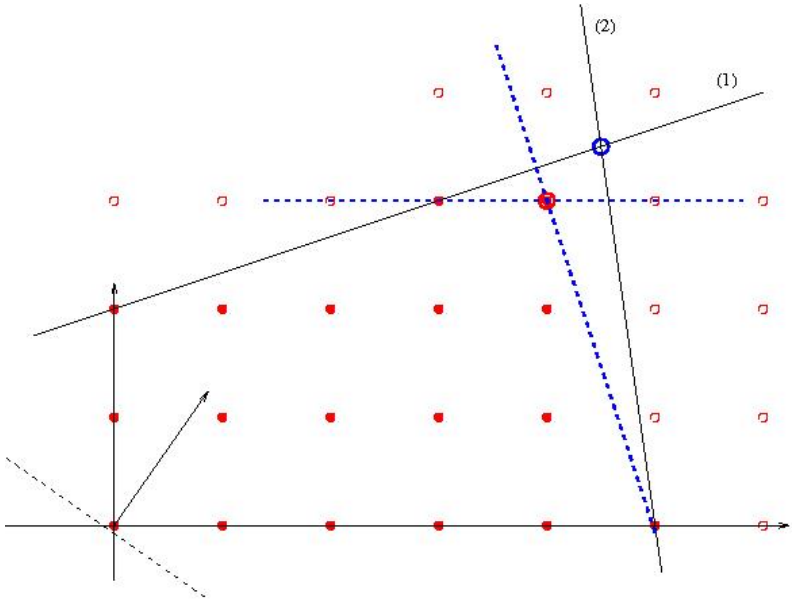

Figure 6: *Weak and strong formulations of the same feasible set of a MILP model. The optimal solution to the MILP model equals that of the LP relaxation of the strong formulation, the constraints of which are indicated by the dashed lines. The LP relaxation of the weaker formulation, whose constraints are indicated by solid lines, yields a solution with fractional variable values.*

imization of the objective function) is then reduced. Hence, it is of great importance to try to find strong valid inequalities for the problem studied ([61]).

5 Mathematical modelling of the scheduling for the multitask cell

In this section we formulate the four complete MILP models of the problem to construct an optimal schedule for the processing of parts in the multitask cell, as described in Section 2.3.

5.1 Definitions required for the MILP models

Most of the parameters and sets defined in Sections 5.1.2 and 5.1.3 are used in all the MILP models for the scheduling of the multitask cell which are described in Sections 5.2–5.4. In Section 5.1.4 a way of calculating realistic release dates and interoperation times from the data given in the ERP system is described and in Section 5.1.5 some

different objective functions are discussed.

5.1.1 Assumptions made for the mathematical formulations

All the processing tools for the multitask cell are assumed to be available and transported to the appropriate resource on time for each route operation. The number of available fixtures is not a limiting factor at present in the multitask cell and it is therefore assumed to be unlimited. For some of the operations, namely, mounting into fixtures, manual deburring, and removing parts from the fixtures, personnel are required during the entire operation, while most of the processing operations require manual work only during a fraction of the operation processing time. There are also other tasks regarding, for example, the machining tools, that the personnel are expected to perform simultaneously with their work with the route operations in the multitask cell. As mentioned in Section 1.3, the availability of personnel for the manual work of the cell is here considered to be sufficient, but in practice this will not always be the case, especially not if the work load of the cell is considerably increased in comparison with the current situation; therefore, we plan to include manpower planning in the future research. The availability of storage before and between the resources is assumed to be unlimited. The assumptions made above are, however, not always true; how they best can be included in the mathematical models is an area for future studies.

5.1.2 Indices and definitions of sets

In order to acquire the appropriate parameter data for the MILP models, the jobs that are to be processed in the multitask cell must be studied and categorized. They can be divided into three categories, since each order (part) passes through three different phases before its processing in the multitask cell. These are

1. planned orders not yet released, i.e. orders that exist in the planning system only;
2. released orders, or so-called production orders, i.e. physical parts being processed outside the cell on their way to the multitask cell; and
3. jobs checked-in into the multitask cell, i.e. parts inside the multitask cell waiting to be processed.

The whole set of jobs to be processed during the planning period considered, i.e. the queue of jobs, is denoted by $\mathcal{J}$. Some of these jobs are to be processed on the same physical part; hence, there are two categories of jobs in phases 2 and 3. The first category is characterized by the part being inside or on its way to the multitask cell for the processing of the corresponding job. The second category is characterized by the part being inside or on its way to the multitask cell for the processing of a preceding job in the routing, before making another round through the factory and, finally, reaching the multitask cell for the processing of the job in question.

As an example of the second category, consider a job $q \in \mathcal{J}$ being the job to be processed at a part's third visit to the multitask cell, as illustrated in Figure 4, while the physical part, on which job q is going to be performed, is on its way to visit the multitask cell for the second time, for processing of job $j \in \mathcal{J}$. Another example, as for the jobs q and l illustrated in Figure 7, occurs when no operations are required outside the multitask cell before the processing of the next job. The pairs of all jobs adjacent in the routing form the product set $\mathcal{Q} \subset \mathcal{J} \times \mathcal{J}$. For the part, whose routing is illustrated in Figure 7, the pairs (j, q) and (q, l) belongs to the set $\mathcal{Q}$.

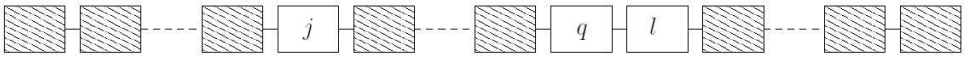

Figure 7: *Routing of a part in the factory with jobs j, q, and l to be processed in the multitask cell. The set of pairs $\{(j, q), (q, l)\} \subseteq \mathcal{Q}$.*

Each job j consists of n_j operations i, i.e., $i \in \mathcal{N}_j = \{1, \ldots, n_j\}$, to be processed inside the multitask cell. The route operations that can be performed inside the cell are listed in Table 1, together with the four possible variants of the ordering of the route operations belonging to the set $\mathcal{N}_j$ for a job j. Note that the route operations $i = 1, 2$ are of the same kind for all the jobs.

Description of operation	**Job type**			
	(i)	(ii)	(iii)	(iv)
mounting into fixture	1	1	1	1
turning/milling/drilling	2	2	2	2
manual deburring	3	3		
automatic deburring	4		3	
removing from fixture	5	4	4	3

Table 1: Each job, j, consists of an ordered set of route operations that are performed inside the multitask cell. There are four variants, here denoted (i),...,(iv), of these sets, which are mathematically denoted by $\mathcal{N}_j, j \in \mathcal{J}$.

The route operations are performed in the ten processing resources in the multitask cell. The first ($i = 1$) and the last ($i = n_j$) route operations are always performed in one of the three set-up/tear-down stations. The set of k resources is denoted by $\mathcal{K}$, and the set of multitask machines is denoted by $\tilde{\mathcal{K}} \subset \mathcal{K}$. All the resources considered for scheduling in the thesis are listed in Table 2.

5.1.3 Definition of parameters and time variables

Linked with each route operation i of job j, henceforth denoted operation (i, j), and each resource k, is a parameter λ_{ijk}, which equals 1 if operation (i, j) is allowed to

k	**Description**	**Route operation**
M/DM 1–3	three set-up/tear-down stations	mounting into & removing from fixture
M/C 1–5	five multitask machines ($\tilde{\mathcal{K}}$)	turning/milling/drilling
Man Gr	one manual deburring station	manual deburring
DBR	one automatic deburring machine	automatic deburring

Table 2: The resources of the multitask cell and the corresponding route operations.

be processed on resource k, and equals 0 otherwise. Since most jobs are only allowed to be processed in a subset of the multitask machines, λ_{2jk} may equal 0 for some $k \in \tilde{\mathcal{K}}$. All data related to time are given in hours relative to a starting time t_0 for the schedule to be calculated. Since the resources are often occupied by the processing of a route operation for another job at time t_0, the parameter a_k is introduced, denoting the time when the resource k will be available for processing a new operation. In Table 3 all parameters describing the input data are listed.

Notation	**Definition**
λ_{ijk}	1, if operation (i, j) can be processed on resource k, 0, otherwise.
a_k	the time when resource k will be available the first time.
d_j	the due date of job j, i.e., the point in time when the last operation n_j of job j is planned to be completed.
r_j	the release date for job j.
p_{ij}	the processing time in hours for operation i of job j.
w	the transportation time for a product between any resources inside the multitask cell.
v_{jq}	the interoperation time between the jobs j and q, where $(j, q) \in \mathcal{Q}$.
M	a sufficiently large positive number that must be greater than the planning horizon.

Table 3: Definition of parameters for indices $i \in \mathcal{N}_j, j \in \mathcal{J}$, and $k \in \mathcal{K}$, if not otherwise specified.

How the parameters r_j and v_{jq} are calculated from the available data for the multitask cell is explained in detail in Section 5.1.4. The parameter M, which is used to represent the logical precedence constraints by linear inequalities, has to be sufficiently large, but not too large, which would cause numerical rounding errors. Therefore, the value of M must be chosen with great care. Therefore, we have developed a heuristic, for computing a "good" value of this parameter, using the same logic as Algorithm 1 presented in Section 5.4.7.

As the continuous time variables appear in the discussion about different objec-

tive functions in Section 5.1.5, they are defined below in Table 4. The completion time and tardiness variables are also common for all the models, while the binary decision variables differ between the different models, and are hence defined next to their respective models.

Notation	Definition
t_{ij}	the starting time of operation (i, j)
s_j	the completion time of job j ($s_j = t_{n_j,j} + p_{n_j,j}$)
h_j	the tardiness of job j ($h_j = \max\{0; s_j - d_j\}$)

Table 4: Definition of the continuous time variables.

5.1.4 Realistic release dates and interoperation times

The means to calculate a realistic value of the release date for a specific job depend on which of the phases 1–3, described in Section 5.1.2, that the job belongs to. If a job j belongs to phase 3, i.e., the part is ready to be processed at time t_0 and is checked-in into the multitask cell, then r_j is set to 0. Release dates for jobs in the phases 1 and 2, i.e., planned orders and released jobs being processed elsewhere on their way to the multitask cell, are somewhat more complicated to determine. In order to calculate these, we need to introduce some new parameters that are used in this section only; they are collected in Table 5 but also explained in the text.

Parameter	Description	Source
ϱ_j	the planned latest release date for job j	ERP
ϑ_j	the standard queue time for job j including transport time	ERP
$\mu_{\mathtt{act}}$	the actual operation, i.e., the part's location at time t_0	ERP
μ_{m_j-1}	the operation preceding job j	ERP
ν_j^0	the standard lead time from start of $\mu_{\mathtt{act}}$ to arrival at MTC	Calc
ρ_μ	the processing time for operation μ	ERP
ς_μ	the set-up time for operation μ	ERP
ϑ_μ	the standard queue time for operation μ	ERP

Table 5: Description of the parameters used in Section 5.1.4 and their sources. ERP denotes the planning system and Calc means that the parameter value is calculated by equation (2).

In the planning system of the multitask cell, there is a *planned latest release date* for each job, which is denoted by ϱ_j. This means that the job j in the multitask cell is planned to be started at the latest at time ϱ_j. Another parameter in the planning system is the so-called *standard queue time*, here denoted by ϑ_j, which is the planned

time between the completion of the preceding operation performed outside the multitask cell and the latest release date ϱ_j, including the time for the transport to the multitask cell. The *desired release date*, r_j, that should be included in the optimization model, should, however, be a realistic estimation of the point in time when the part arrives to the multitask cell. Therefore, a fraction of the standard queue time must be subtracted from ϱ_j. A measure that is often used at Volvo Aero is the transport time constituting about 20% of the standard queue time; hence we choose to compute r_j as

$$r_j = \max\{\varrho_j - 0.8\vartheta_j - t_0; \nu_j^0\}, \tag{1}$$

where ν_j^0 is the standard lead time from the preceding operation on the part to be processed outside the multitask cell at time t_0 till it arrives at the multitask cell. Let $\mu_{\texttt{act}}$ denote this actual operation and $\mu_1, \ldots, \mu_{m_f}$ denote the operations in the routing for the completion of the part to a final product; see Figure 8. Using this notation, job j equals operation μ_{m_j}, and hence μ_{m_j-1} denotes the operation preceding job j. Note that the word job is only used to denote the operations to be scheduled in the multitask cell. The operations listed in Table 1 (which are performed inside the multitask cell) are called route operations, in order to distinguish them from the operations $\mu_l, l = 1, \ldots, f$. Using this notation, ν_j^0 is given by the equality

$$\nu_j^0 = \sum_{\mu=\mu_{\texttt{act}}+1}^{\mu_{m_j-1}} (\rho_\mu + \varsigma_\mu + \vartheta_\mu) + 0.2\vartheta_j, \tag{2}$$

where ρ_μ, ς_μ, and ϑ_μ denote the process, set-up, and queue time, respectively, of operation μ, which is processed elsewhere, i.e., not in the multitask cell.

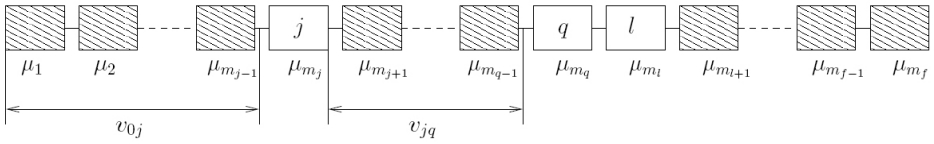

Figure 8: *The interoperation time, v_{jq}, between jobs j and q is shown together with ν_{0j} for the case of a planned order for job j.*

The parameter v_{jq} is utilized to prevent the jobs j and q being scheduled too close in time, for the case when both jobs are to be performed on the same physical part; the parameter v_{jq} is defined in Table 3. It is the planned lead time between the completion time of job j and the starting time of job q of the tranportation and processing times for operations performed outside the multitask cell; see Figure 8. The definition of v_{jq} is given by the equality

$$v_{jq} = \sum_{\mu=\mu_{m_j}+1}^{\mu_{m_q-1}} (\rho_\mu + \varsigma_\mu + \vartheta_\mu) + 0.2\vartheta_q,$$

where ρ_μ, ς_μ, and ϑ_μ denote respectively the process, set-up and queue times of operation μ, which is processed elsewhere (cf. (2)). As mentioned before, $0.2\vartheta_q$ is the estimated transport time to the multitask cell from the operation preceding job q for this physical part.

5.1.5 Objectives for the scheduling of the multitask cell

The classic scheduling objective is to minimize the *makespan*, i.e., to minimize the completion time of the latest job in the schedule; see e.g. [27]. This is, however, not a suitable objective for instances where not all jobs are available from the start, but are expected to arrive at given release dates, as is the case for the multitask cell. Another weakness of the makespan objective is that it does not take the tardiness h_j into account. Another objective often used is to minimize the *total tardiness*, i.e., the sum of the tardiness of all jobs. This is a suitable objective for instances with tardy jobs, but not for instances with jobs that may be completed before their respective due dates. One way to consider also these jobs in the objective is the addition of an objective term expressing total earliness (see [63]), which results in a suitable objective for a just-in-time environment with a reliable and stable process.

According to the managers of the multitask cell, the most important objectives are to maximize the utilization of the multitask cell and to minimize the tardiness of the jobs. Even though it is, in fact, not desirable to finish jobs too early (because of the possibility to choke the system with large total lead times as a consequence) we have chosen to add an objective term with a weight A times the total completion time to the term of a weight B times the total tardiness. We have chosen this compromise with the two weights A and B, with $0 \leq A \leq B$, since the minimization of the sum of the completion times (or mean flowtime as it is called in some references, e.g. [18]) is a secondary objective, while the main objective is to minimize the total tardiness. With this objective function, the utilization is maximised through the minimization of the sum of the completion times, while tardy jobs are prioritized. The benefit of scheduling jobs early is the possibility to cope with the everyday reality in the multitask cell which is filled with unexpected events such as operators getting ill, non-conformance parts leaving the queue of parts to be scheduled, and machine break-downs, only to name a few.

Since each job occupies a fixture throughout the whole sequence $1, \ldots, n_j$ of operations, the negative objective term $-\varepsilon t_{1j}$, ε being a small positive number, is added in order to reduce the time when each fixture is occupied. Hence, the objective is to minimize the function during which

$$\sum_{j \in \mathcal{J}} \left(A s_j - \varepsilon t_{1j} + B h_j \right). \tag{3}$$

It is important that $\varepsilon \ll \min\{A, B\}$, since the second term in the sum strives to schedule the jobs as late as possible. The discussion on the advantages and disadvantages of this objective function is continued in Section 5.4.6, where an alternative objective function, minimizing the work load variance, is also presented.

5.2 The full engineer's model

In this section we present a mixed integer optimization model for the full problem, i.e., for the whole multitask cell including all ten resources. This model belongs to the Manne family (see Section 3.1.1), and is called the *full engineer's model*, since it is a quite intuitive model for the whole multitask cell.

5.2.1 Definition of variables

The time variables used in this model are the continuous variables defined in Section 5.1.3 for starting, completion, and tardiness of a job. There are two sets of binary decision variables in this model, namely z_{ijk}, which determines to which resource a certain operation is allocated, and y_{ijpqk}, denoting the precedence relations between the operations. They are defined as

$$z_{ijk} = \begin{cases} 1, & \text{if operation } (i,j) \text{ is allocated to resource } k, \\ 0, & \text{otherwise,} \end{cases}$$

$$y_{ijpqk} = \begin{cases} 1, & \text{if op } (i,j) \text{ is processed before op } (p,q) \text{ on resource } k, \\ 0, & \text{otherwise,} \end{cases}$$

for $i,p \in \mathcal{N}_j,\ j,q \in \mathcal{J}$, such that $(i,j) \neq (p,q)$, and $k \in \mathcal{K}$, and where "op (i,j)" denotes route operation i of job j.

5.2.2 The engineer's mathematical optimization model

Given the parameters and sets defined in Section 5.1.2 and the objective function (3) given in Section 5.1.5, the problem to schedule the multitask cell is formulated as that to

$$\text{minimize} \quad \sum_{j \in \mathcal{J}} (As_j - \varepsilon t_{1j} + Bh_j) \tag{4a}$$

$$\text{subject to} \quad \sum_{k \in \mathcal{K}} z_{ijk} = 1, \qquad i \in \mathcal{N}_j,\ j \in \mathcal{J}, \tag{4b}$$

$$z_{ijk} \leq \lambda_{ijk}, \qquad i \in \mathcal{N}_j,\ j \in \mathcal{J},\ k \in \mathcal{K}, \tag{4c}$$

$$y_{ijpqk} + y_{pqijk} \leq z_{ijk}, \qquad i \in \mathcal{N}_j,\ p \in \mathcal{N}_q,\ j,q \in \mathcal{J},\ (i,j) \neq (p,q),\ k \in \mathcal{K}, \tag{4d}$$

$$y_{ijpqk} + y_{pqijk} + 1 \geq z_{ijk} + z_{pqk}, \qquad i \in \mathcal{N}_j,\ p \in \mathcal{N}_q,\ j,q \in \mathcal{J},\ (i,j) \neq (p,q),\ k \in \mathcal{K}, \tag{4e}$$

$$t_{ij} + p_{ij} - M(1 - y_{ijpqk}) \leq t_{pq}, \qquad i \in \mathcal{N}_j,\ p \in \mathcal{N}_q,\ j,q \in \mathcal{J},\ (i,j) \neq (p,q),\ k \in \mathcal{K}, \tag{4f}$$

$$t_{ij} + p_{ij} + w \leq t_{i+1,j}, \qquad i \in \mathcal{N}_j \setminus \{n_j\},\ j \in \mathcal{J}, \tag{4g}$$

$$t_{1j} \geq r_j, \qquad j \in \mathcal{J}, \tag{4h}$$

$$t_{ij} \geq a_k z_{ijk}, \qquad j \in \mathcal{J},\ k \in \mathcal{K}, \tag{4i}$$

$$t_{1q} \geq s_j + v_{jq}, \qquad (j,q) \in \mathcal{Q}, \tag{4j}$$

$$s_j - t_{n_j j} = p_{n_j j}, \qquad j \in \mathcal{J}, \tag{4k}$$

$$
\begin{aligned}
h_j &\geq s_j - d_j, && j \in \mathcal{J}, && \text{(4l)}\\
h_j &\geq 0, && j \in \mathcal{J}, && \text{(4m)}\\
t_{ij} &\geq 0, && i \in \mathcal{N}_j,\ j \in \mathcal{J}, && \text{(4n)}\\
z_{ijk} &\in \{0,1\}, && i \in \mathcal{N}_j,\ j \in \mathcal{J},\ k \in \mathcal{K}, && \text{(4o)}\\
y_{ijpqk} &\in \{0,1\}, && i \in \mathcal{N}_j,\ p \in \mathcal{N}_q,\ j,q \in \mathcal{J}, && \text{(4p)}\\
& && (i,j) \neq (p,q),\ k \in \mathcal{K}. &&
\end{aligned}
$$

The constraints (4b) ensure that every operation is processed exactly once, and the constraints (4c) make sure that each operation is scheduled on a resource allowed for this specific operation. The constraints (4d) and (4e) induce an unambiguous ordering of the operations that are to be processed in the same resource. The constraints (4d) make sure that y_{ijpqk} and y_{pqijk} do not both equal 1; i.e., if operation (i,j) precedes operation (p,q), operation (p,q) cannot precede operation (i,j). If operations (i,j) and (p,q) are to be performed on the same machine, then the constraints (4e) regulate that one of y_{ijpqk} and y_{pqijk} must possess the value 1.

Furthermore, the constraints (4f) make sure that the starting time of operation (p,q) is scheduled after the completion of the previous operation in the same resource. The parameter M is a big number, whose purpose is to relax the constraints whenever y_{ijpqk} is valued zero, i.e., when the operations are not scheduled for processing in the same resource. See Section 6.2.4 for an analysis on the impact of different values of M on the computation time. Generally, in scheduling problems the constraints $t_{pq} + p_{pq} - My_{ijpqk} \leq t_{ij}$, being symmetric to the constraints (4f), are required, as e.g., in [40], but these become redundant here since y_{ijpqk} and y_{pqijk} are regulated by the inequalities (4d) and (4e). The effect of an alternative formulation of these constraints for the engineer's model for the five multitask machines is discussed in Section 5.3.1.

The constraints (4g) ensure that the operations within the same job j are scheduled in the correct order and that each operation is not scheduled until the previous operation has been completed and the corresponding part transported to the current resource in the set $\mathcal{N}_j$. The constraints (4h) regulate the starting time of the first operation of every job, so that no job is scheduled prior to its release date. Moreover, the constraints (4i) make sure that no operation is scheduled in any resource before the resource is available for the first time. Any pair of two jobs, (j,q), say, belonging to the set $\mathcal{Q}$ must be separated at least by the planned interoperation time v_{jq}; this is indicated by the constraints (4j). The completion time and the tardiness of each job j are defined by the constraints (4k)–(4m). Note that the nonnegativity constraints (4n) for the starting times are redundant due to equations (4g)–(4i) whenever the release date r_j is nonnegative.

5.2.3 Preliminary computations using the full engineer's model

Tests performed on real data instances with this model on a 4 Gb quad-core Intel Xeon 3.2 GHz system using AMPL-CPLEX12 [15, 26] as optimization software, were able to solve very small instances with 10 jobs to optimality within 3 minutes, but for instances of 15 jobs, the computation times were as high as three months; see [52].

5.3 The machining/feasibility problem decomposition

Since the computation times for the so-called full engineer's model (4) were too long for practical utilization, the corresponding full scheduling problem was decomposed into two subproblems. The first subproblem, henceforth called the *machining problem*, is to find an optimal sequence of operations for each of the five processing machines, i.e., the resources defined by the set $\tilde{\mathcal{K}}$. The reason for this choice of decomposition is that the work load on these resources is much higher than that on the other resources; an optimal schedule based on real data is illustrated in Figure 9. The second subproblem, henceforth called the *feasibility problem*, is to generate a feasible schedule for all ten resources, with an optimal sequence for each of the five processing machines as input data.

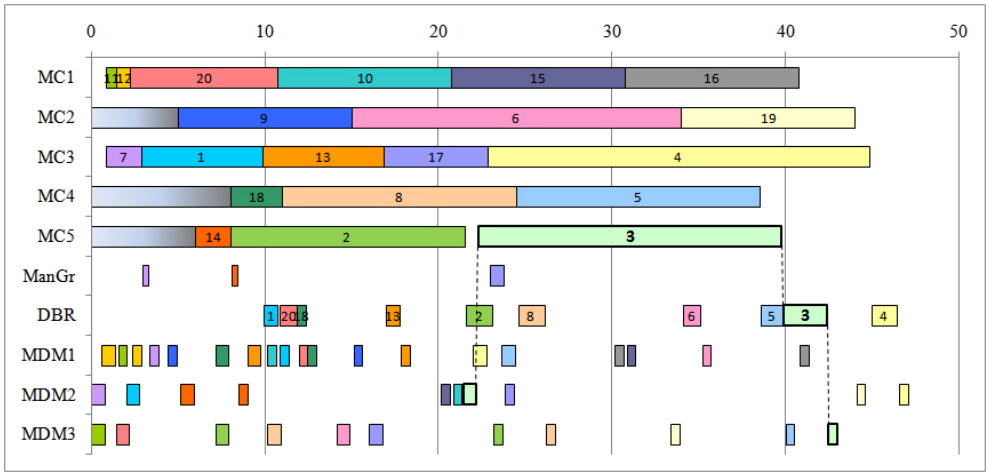

Figure 9: *An example on a schedule based on real data. The work load on the processing machines (MC1–5) is much higher than that on the other resources. The machining problem is to find an optimal schedule for the five processing machines. The route of job 3 is indicated by dotted lines.*

5.3.1 The engineer's model of the machining problem

The model first developed for the machining problem was based on the same logic as the full engineer's model; it is presented in [55] (Paper I in this thesis). The machining problem is to find an optimal schedule for the five processing machines, which define the set $\tilde{\mathcal{K}}$. From Table 1 follows that only the operations $(2, j),\ j \in \mathcal{J}$ are included in this problem. Therefore, the operation indices (corresponding to the set N_j, typically denoted by i or p) can be dropped from the variables and parameters in this model. The release date r_j^{m} for job j in the machining problem is thus composed by the actual release date r_j, the processing time p_{1j} of the first operation (mounting the part into

a fixture), and the internal transportation time w, i.e.,

$$r_j^{\mathrm{m}} = r_j + p_{1j} + w, \quad j \in \mathcal{J}. \tag{5}$$

Accordingly, the resulting completion times are adjusted by the time required for the processing of the post-machining operations and the corresponding internal transports as

$$p_j^{\mathrm{pm}} = \sum_{i=3}^{n_j} (p_{ij} + w), \quad j \in \mathcal{J},$$

so that it reflects the completion of the whole job, including all the operations included. The parameters a_k, d_j, and M are unchanged and are thus as defined in Table 3, and the remaining parameters are redefined according to

$$\lambda_{jk} := \lambda_{2jk}, \qquad j \in \mathcal{J}, k \in \tilde{\mathcal{K}}, \tag{6a}$$
$$p_j := p_{2j}, \qquad j \in \mathcal{J}, \tag{6b}$$
$$v_{jq}^{\mathrm{m}} := v_{jq} + p_{1q} + w, \quad (j,q) \in \mathcal{Q}. \tag{6c}$$

The variables of the engineer's model are also redefined so that they represent the second operation of the jobs, except for the completion time s_j and the tardiness h_j of job j, which are kept related to the end of the last operation of the job. Hence, the variables are defined as

$$z_{jk} = \begin{cases} 1, & \text{if op } (2,j) \text{ is allocated to resource } k, \\ 0, & \text{otherwise,} \end{cases} \qquad j \in \mathcal{J}, k \in \tilde{\mathcal{K}},$$

$$y_{jqk} = \begin{cases} 1, & \text{if op } (2,j) \text{ is processed before op } (2,q) \text{ on resource } k, \\ 0, & \text{otherwise,} \end{cases} \qquad j,q \in \mathcal{J}, k \in \tilde{\mathcal{K}},$$

$t_j =$ the starting time of op $(2,j)$, $\quad j \in \mathcal{J}$,
$s_j = t_j + p_{2j} + p_j^{\mathrm{pm}}$, i.e., the completion time of job j, $\quad j \in \mathcal{J}$,
$h_j = \max\{0; s_j - d_j\}$, i.e., the tardiness of job j, $\quad j \in \mathcal{J}$,

where op $(2,j)$ denotes the machining operation of job j. Since the machining problem deals with scheduling only the second operation, the second term in the objective function (3) are not applicable here. The engineer's model of the machining problem is thus to

$$\text{minimize} \quad \sum_{j \in J} (A s_j + B h_j), \tag{7a}$$

$$\text{subject to} \quad \sum_{k \in \tilde{\mathcal{K}}} z_{jk} = 1, \qquad j \in \mathcal{J}, \tag{7b}$$

$$z_{jk} \le \lambda_{jk}, \qquad j \in \mathcal{J},\ k \in \tilde{\mathcal{K}}, \tag{7c}$$

$$y_{jqk} + y_{qjk} \le z_{jk}, \qquad j,q \in \mathcal{J},\ j \ne q,\ k \in \tilde{\mathcal{K}}, \tag{7d}$$

$$y_{jqk} + y_{qjk} + 1 \ge z_{jk} + z_{qk}, \qquad j,q \in \mathcal{J},\ j \ne q,\ k \in \tilde{\mathcal{K}}, \tag{7e}$$

$$t_j + p_j - t_q \leq M(1 - y_{jqk}), \quad j, q \in \mathcal{J},\ j \neq q,\ k \in \tilde{\mathcal{K}}, \tag{7f}$$
$$t_j \geq r_j^{\mathrm{m}}, \quad j \in \mathcal{J}, \tag{7g}$$
$$t_j \geq a_k z_{jk}, \quad j \in \mathcal{J}, \tag{7h}$$
$$t_q \geq s_j + v_{jq}^{\mathrm{m}}, \quad (j, q) \in \mathcal{Q}, \tag{7i}$$
$$s_j - t_j = p_j + p_j^{\mathrm{pm}}, \quad j \in \mathcal{J}, \tag{7j}$$
$$h_j \geq s_j - d_j, \quad j \in \mathcal{J}, \tag{7k}$$
$$h_j \geq 0, \quad j \in \mathcal{J}, \tag{7l}$$
$$t_j \geq 0, \quad j \in \mathcal{J}, \tag{7m}$$
$$z_{jk} \in \{0, 1\}, \quad j \in \mathcal{J},\ k \in \tilde{\mathcal{K}}, \tag{7n}$$
$$y_{jqk} \in \{0, 1\}, \quad j, q \in \mathcal{J},\ j \neq q,\ k \in \tilde{\mathcal{K}}. \tag{7o}$$

Zhu and Heady [63] propose a mixed integer programming model for a similar multi-machine problem. It employs the variables $\bar{y}_{jq}$, which are only defined for $j < q$ and equals 1 if job j precedes job q, and 0 otherwise (Y_{ij} in their notation). Instead of using the constraints (7d)–(7f), they propose the following constraints (the constraints (5a) and (5b) in [63]), rewritten in our notation:

$$t_j + p_j - M(3 - \bar{y}_{jq} - z_{jk} - z_{qk}) \leq t_q,\ j \neq q, j = 1, \ldots, N, q = j+1, \ldots, N, k \in \tilde{\mathcal{K}}, \tag{8a}$$
$$t_q + p_q - M(2 + \bar{y}_{jq} - z_{jk} - z_{qk}) \leq t_j,\ j \neq q, j = 0, \ldots, N, q = j+1, \ldots, N, k \in \tilde{\mathcal{K}}, \tag{8b}$$

where N denotes the total number of jobs. Here $j = 0$ denotes a fictitious job which is introduced to simplify the definition of the constraints. This job is always present and always performed first with zero processing time, and hence $t_0 = 0,\ z_{0k} = 1,\ k \in \mathcal{K}$, and $\bar{y}_{0j} = 1,\ j = 1, \ldots, N$.

Preliminary tests indicate that the CPU time needed to solve the engineer's model (7) is significantly lower than that needed to solve the model utilizing the constraints (8a)–(8b). A difference that seems to be crucial is that the variables $\bar{y}_{jq}$ are not defined for $j \geq q$. Defined in this way, the constraints (7d) and (7e) do not make any sense, but as pointed out in Section 5.2.2, these constraints make the constraint that would be symmetric to (4f) redundant, which seems to be significantly beneficial with regard to CPU times.

The effect of the value of the parameter M on the computation times are investigated, and the computational results presented, in Section 6.

5.3.2 The feasibility problem

The goal of the feasibility problem is to produce good feasible schedules for the remaining resources of the multitask cell, i.e., the three set-up and the two deburring stations. Since the three set-up stations are identical, long computation times may arise due to the corresponding symmetries in the mathematical model. Therefore, the objective function of the full engineer's model, i.e., the function (4a), is here extended with a weighted sum of the variables z_{ijk}. The weights ω_k are small in comparison with the weights A and B in the objective, and of different magnitudes for the three set-up stations in order to break the symmetry. For alternative ways of tackling these problem; see [34] and [32].

The job sequence in each of the processing machines, that is, the sequence for the operations $(2, j),\ j \in \mathcal{J}$, are fixed to the corresponding optimal values y^{m}_{jqk} and z^{m}_{jk}, of the variables in the machining problem; this is expressed by the constraints (9o) and (9p). However, none of the time variables are fixed to their optimal values in the solution of the machining problem, since the feasibility problem may then become infeasible if, e.g., two deburring operations collide in the solution to the feasibility problem, which would delay the completion times and possibly also the starting times of succeeding jobs for the same part in the processing machines. This is illustrated in Figure 9, where job 3 is delayed about one hour (compared to the optimal solution of the corresponding machining problem), since jobs 3 and 5 would collide in the resource for robot deburring if the completion times of the jobs were fixed to their optimal values from the machining problem. The only model for the feasibility problem that is presented in this thesis is equivalent to the full engineer's model except for the objective function and the constraints (9o) and (9p) described above. We call it the feasibility model and it is formulated as that to

$$\text{minimize} \quad \sum_{j\in\mathcal{J}} \left(As_j - \varepsilon t_{1j} + Bh_j + \sum_{i\in\mathcal{N}_j} \sum_{k\in\mathcal{K}} \omega_k z_{ijk} \right), \tag{9a}$$

$$\text{subject to} \quad \sum_{k\in\mathcal{K}} z_{ijk} = 1, \qquad i \in \mathcal{N}_j,\ j \in \mathcal{J}, \tag{9b}$$

$$z_{ijk} \le \lambda_{ijk}, \qquad i \in \mathcal{N}_j,\ j \in \mathcal{J},\ k \in \mathcal{K}, \tag{9c}$$

$$y_{ijpqk} + y_{pqijk} \le z_{ijk}, \qquad i \in \mathcal{N}_j,\ p \in \mathcal{N}_q,\ j,q \in \mathcal{J},\ (i,j) \ne (p,q),\ k \in \mathcal{K}, \tag{9d}$$

$$y_{ijpqk} + y_{pqijk} + 1 \ge z_{ijk} + z_{pqk}, \qquad i \in \mathcal{N}_j,\ p \in \mathcal{N}_q,\ j,q \in \mathcal{J},\ (i,j) \ne (p,q),\ k \in \mathcal{K}, \tag{9e}$$

$$t_{ij} + p_{ij} - M(1 - y_{ijpqk}) \le t_{pq}, \qquad i \in \mathcal{N}_j,\ p \in \mathcal{N}_q,\ j,q \in \mathcal{J},\ (i,j) \ne (p,q),\ k \in \mathcal{K}, \tag{9f}$$

$$t_{ij} + p_{ij} + w \le t_{i+1,j}, \qquad i \in \mathcal{N}_j \setminus \{n_j\},\ j \in \mathcal{J}, \tag{9g}$$

$$t_{1j} \ge r_j, \qquad j \in \mathcal{J}, \tag{9h}$$

$$t_{ij} \ge a_k z_{ijk}, \qquad j \in \mathcal{J},\ k \in \mathcal{K}, \tag{9i}$$

$$t_{1q} \ge s_j + v_{jq}, \qquad (j,q) \in \mathcal{Q}, \tag{9j}$$

$$s_j - t_{n_j j} = p_{n_j j}, \qquad j \in \mathcal{J}, \tag{9k}$$

$$h_j \ge s_j - d_j, \qquad j \in \mathcal{J}, \tag{9l}$$

$$h_j \ge 0, \qquad j \in \mathcal{J}, \tag{9m}$$

$$t_{ij} \ge 0, \qquad i \in \mathcal{N}_j,\ j \in \mathcal{J}, \tag{9n}$$

$$y_{2j2qk} = y^{\mathrm{m}}_{jqk}, \qquad j,q \in \mathcal{J},\ k \in \tilde{\mathcal{K}}, \tag{9o}$$

$$z_{2jk} = z^{\mathrm{m}}_{jk}, \qquad j \in \mathcal{J},\ k \in \tilde{\mathcal{K}}, \tag{9p}$$

$$z_{ijk} \in \{0,1\}, \qquad i \in \mathcal{N}_j,\ j \in \mathcal{J},\ k \in \mathcal{K}, \tag{9q}$$

$$y_{ijpqk} \in \{0,1\}, \qquad i \in \mathcal{N}_j,\ p \in \mathcal{N}_q,\ j,q \in \mathcal{J},\ (i,j) \ne (p,q),\ k \in \mathcal{K}. \tag{9r}$$

Tests performed on the same software and hardware as mentioned in 5.2.3, were also carried out with the engineer's machining and feasibility models, solved in sequence. The computation times decreased substantially compared to the computation time for the full engineer's model; see Figure 10.

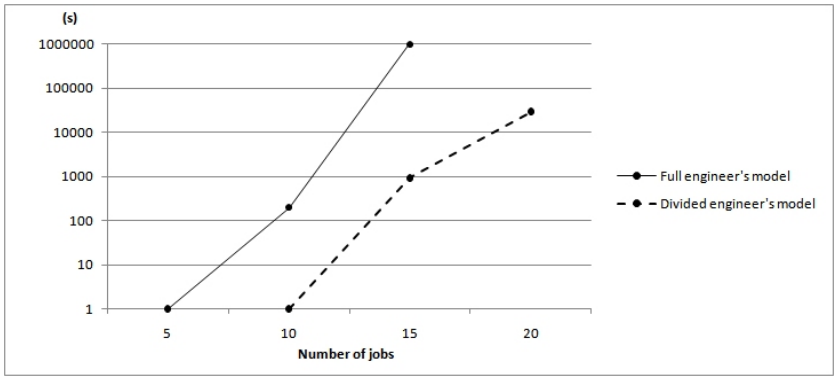

Figure 10: *The mean computation time (in CPU-seconds) plotted for a small test with real data instances as a function of the number of jobs included in the instances. The engineer's machining and feasibility models to be solved in sequence are called the* Divided engineer's model.

5.4 Time-indexed formulations

Although the computation times decreased substantially due to the decomposition, they were not short enough to be applicable for a real implementation in the multitask cell. Therefore, we have developed two additional models for the machining problem, employing time-indexed decision variables, for solving the machining problem; they are presented in Sections 5.4.4 and 5.4.5. The results gained from one of these models are very promising; see Section 6.2.2 for a comparison with the engineer's model w.r.t. CPU times. The reason for the choice of developing alternative models for the machining problem is that the computation time needed to solve the feasibility problem is substantially shorter than that needed to solve the machining problem.

5.4.1 Time intervals

The planning horizon of the schedule is divided into $T + 1$ intervals, each of length ℓ hours; see Figure 11. The index $u \in \mathcal{T} = \{0, 1, \ldots, T\}$ denotes the interval starting at time ℓu and ending at time $\ell(u + 1)$.

Figure 11: *The planning horizon is divided into $T + 1$ intervals of length ℓ hours.*

The value of the parameter T has to be large enough such that the planning horizon of $(T+1)\ell$ hours contains an optimal schedule, but as small as possible, since the computation times become shorter for smaller values of T. The reason for this is explained in Section 5.4.7, where a heuristic for determining a suitable value of T is presented.

5.4.2 Definition of variables

In Sections 5.4.4 and 5.4.5 two time-indexed formulations of the machining problem are presented. They use two kinds of time-indexed decision variables, called *nail* and *plateau* variables, respectively. The nail variables are defined as

$$x_{jku} = \begin{cases} 1, & \text{if the processing of op } (2,j) \text{ starts at resource } k \\ & \text{at the beginning of time interval } u, \\ 0, & \text{otherwise,} \end{cases} \qquad j \in \mathcal{J},\ k \in \tilde{\mathcal{K}},\ u \in \mathcal{T},$$

and the plateau variables are defined as

$$w_{jku} = \begin{cases} 1, & \text{if op } (2,j) \text{ is processed at resource } k \\ & \text{during time interval } u, \\ 0, & \text{otherwise.} \end{cases} \qquad j \in \mathcal{J},\ k \in \tilde{\mathcal{K}},\ u \in \mathcal{T},$$

The plateau variables are of the same kind as those used by Bowman in [7]; see the discussion in Section 3.1.1. In the model using plateau variables, also the binary variables z_{jk} from the engineer's model, defined in Section 5.3.1, are required. Figure 12 illustrates the relations between the nail and plateau variables for the jobs j and q.As before, we denote, in both models, by s_j the completion time of job j, and by h_j the tardiness of job j.

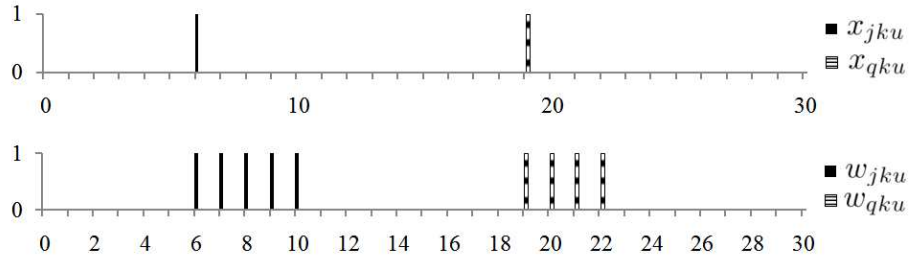

Figure 12: Relations between the plateau and nail variables for the two jobs j and q, which are both processed in resource k. The processing times are given by $p_j = 5\ell$h and $p_q = 4\ell$h, respectively. Operation $(2,j)$ starts at time interval $u = 6$ and operation $(2,q)$ starts at time interval $u = 19$.

5.4.3 New values for all time parameters

All parameters related to time used in the time-indexed models have to be expressed as multiples of time periods. In order to maintain the feasibility of every solution to the resulting model, this means that the values of p_j, a_k, and r_j^{m} must be scaled and rounded up, while the due dates, d_j, have to be scaled and truncated. New parameters are thus defined according to

$$\begin{aligned} \tilde{p}_j &:= \left\lceil \frac{p_j}{\ell} \right\rceil, \ \tilde{r}_j^{\mathrm{m}} := \left\lceil \frac{r_j^{\mathrm{m}}}{\ell} \right\rceil, \text{ and } \tilde{d}_j := \left\lfloor \frac{d_j}{\ell} \right\rfloor, \quad j \in \mathcal{J}, \\ \tilde{a}_k &:= \left\lceil \frac{a_k}{\ell} \right\rceil, \qquad\qquad\qquad\qquad\qquad\qquad\quad k \in \tilde{\mathcal{K}}. \end{aligned}$$

Regarding jobs to be performed on the same physical part, say $(j,q) \in \mathcal{Q}$, the parameter v_{jq}^{m} is redefined to represent the planned lead time between the start of operation $(2,j)$ and the start of operation $(2,q)$ according to

$$\tilde{v}_{jq}^{\mathrm{pm}} := \left\lceil \frac{1}{\ell}(p_j + p_j^{\mathrm{pm}} + v_{jq}^{\mathrm{m}}) \right\rceil, \quad (j,q) \in \mathcal{Q}.$$

The parameter p_j^{pm} is analogously redefined to include the processing time of the machining operation, before it is scaled and rounded up, as

$$\tilde{p}_j^{\mathrm{pm}} := \left\lceil \frac{1}{\ell}(p_j + p_j^{\mathrm{pm}}) \right\rceil, \quad j \in \mathcal{J}.$$

5.4.4 A time-indexed model with nail variables

The time-indexed model with so-called nail variables presented in this section has been developed using findings presented in [60]. We also present an alternative formulation of the constraints (11e).

The model is to

$$\text{minimize} \qquad \sum_{j \in \mathcal{J}} (As_j + Bh_j), \tag{11a}$$

$$\text{subject to} \qquad \sum_{k \in \tilde{\mathcal{K}}} \sum_{u \in \mathcal{T}} x_{jku} = 1, \qquad j \in \mathcal{J}, \tag{11b}$$

$$\sum_{u \in \mathcal{T}} x_{jku} \leq \lambda_{jk}, \quad j \in \mathcal{J}, k \in \tilde{\mathcal{K}}, \tag{11c}$$

$$\sum_{j \in \mathcal{J}} \sum_{\nu=(u-\tilde{p}_j+1)_+}^{u} x_{jk\nu} \leq 1, \qquad k \in \tilde{\mathcal{K}}, u = 0, \ldots, T, \tag{11d}$$

$$\sum_{k \in \tilde{\mathcal{K}}} \left(\sum_{\mu=0}^{u} x_{jk\mu} - \sum_{\nu=0}^{u+\tilde{v}_{jq}^{\mathrm{pm}}} x_{qk\nu} \right) \geq 0, \qquad (j,q) \in \mathcal{Q}, u = 0, \ldots, T - \tilde{v}_{jq}^{\mathrm{pm}}, \tag{11e}$$

$$x_{jku} = 0, \qquad (j,q) \in \mathcal{Q},\ k \in \tilde{\mathcal{K}},\ u = T - \tilde{v}^{\mathrm{pm}}_{jq}, \dots, T, \tag{11f}$$

$$\sum_{k \in \tilde{\mathcal{K}}} \sum_{u \in \mathcal{T}} u x_{jku} + \tilde{p}^{\mathrm{pm}}_{j} = s_j, \qquad j \in \mathcal{J}, \tag{11g}$$

$$s_j - h_j \le \tilde{d}_j, \qquad j \in \mathcal{J}, \tag{11h}$$

$$h_j \ge 0, \qquad j \in \mathcal{J}, \tag{11i}$$

$$x_{jku} = 0, \qquad j \in \mathcal{J},\ k \in \tilde{\mathcal{K}},\ u = 0, \dots, \max\{\tilde{r}^{\mathrm{m}}_{j}; \tilde{a}_k\}, \tag{11j}$$

$$x_{jku} \in \{0,1\}, j \in \mathcal{J},\ k \in \tilde{\mathcal{K}},\ u \in \mathcal{T}, \tag{11k}$$

where $(u)_+ := \max\{0; u\}$ denotes the projection of $u \in \mathbb{Z}$ onto $\mathbb{Z}_+$. The model with nail variables is equivalent to the engineer's model (7) in Section 5.3.1 for the case when all data in the latter are given in entire time intervals, i.e., expressed in multiples of ℓ.

The variables z_{jk}, t_j, and y_{jqk} of the engineer's model are related to the time-indexed model's nail variables x_{jku} as follows:

$$z_{jk} = \sum_{u \in \mathcal{T}} x_{jku}, \qquad j \in \mathcal{J}, k \in \tilde{\mathcal{K}}, \tag{12a}$$

$$t_j = \sum_{k \in \tilde{\mathcal{K}}} \sum_{u \in \mathcal{T}} u x_{jku}, \qquad j \in \mathcal{J}, \tag{12b}$$

$$y_{jqk} = \begin{cases} 1, & \text{if } 0 < \sum_{u \in \mathcal{T}} u x_{jku} < \sum_{u \in \mathcal{T}} u x_{qku}, \\ 0, & \text{otherwise,} \end{cases} \qquad j,q \in \mathcal{J}, j \ne q, k \in \tilde{\mathcal{K}}. \tag{12c}$$

Conversely, the time-indexed model's variables can be expressed in terms of the variables of the engineer's model (provided that all data are given in multiples of the length ℓ of the time intervals)

$$x_{jku} = \begin{cases} z_{jk}, & u = t_j, \\ 0, & u \in \mathcal{T} \setminus \{t_j\}, \end{cases} \qquad j \in \mathcal{J}, k \in \tilde{\mathcal{K}}. \tag{13}$$

The constraints (11b) ensure that every operation $(2, j)$ is processed exactly once, and the constraints (11c) make sure that each machining operation is scheduled in an allowed resource. These constraints are, through the equivalences (12) and (13), equivalent to the constraints (7b) and (7c) of the engineer's model, respectively. The constraints (11d), which regulate that only one operation at a time is scheduled in resource k, correspond to the constraints (7d)–(7f).

The constraints (11e)–(11f) ensure that operation $(2, j)$ is scheduled to start at least the planned lead time of $\tilde{v}^{\mathrm{pm}}_{jq}$ time steps before the start of operation $(2, q)$, for all indices $j, q \in \mathcal{J}$ such that $(j, q) \in \mathcal{Q}$. These constraints correspond to the constraints (7i) of the engineer's model. We have also developed the following set of constraints which equivalently describe these precedence relations:

$$\sum_{k \in \tilde{\mathcal{K}}} \left(\sum_{\mu=u}^{T} x_{jk\mu} + \sum_{\nu=0}^{u+\tilde{v}^{\mathrm{pm}}_{jq}-1} x_{qk\nu} \right) \le 1, \qquad (j,q) \in \mathcal{Q},\ u = 0, \dots, T - \tilde{v}^{\mathrm{pm}}_{jq} + 1. \tag{14}$$

Preliminary computational tests indicate that the constraints (14) are as strong as the constraints (11e). A closer study of the differences between these constraints is an interesting subject for further research.

The constraints (11g)–(11i) define the completion times, s_j and the tardiness, h_j, as do the constraints (7j)–(7l). The constraints (11j) make sure that operation $(2, j)$ is not scheduled before its release date or before the resource k chosen by the model is available, and correspond to the constraints (7g) and (7h) of the engineer's model.

5.4.5 A time-indexed model with plateau variables

Compared with using the nail variables or those in the engineer's model (7), one advantage gained by using the plateau variables w_{jku}, as defined in Section 5.4.2, is that it is easy to employ the objective function described in Section 5.4.6, i.e., to minimize the work load variance. The model with the plateau variables is, however, a weaker formulation than the model (11), since the plateau variables are aggregates of nail variables; see the equations (16) below and the discussion that follows. (For an illustrative example of weak and strong formulations related to variable aggregations, see [61, Section 13.4.1].) Moreover, preliminary tests show that the time-indexed model with plateau variables requires longer computation times for solving our test instances of the scheduling problem, than the model (11).

The model is to

$$\text{minimize} \quad \sum_{j\in\mathcal{J}}(As_j + Bh_j), \tag{15a}$$

$$\text{subject to} \quad \sum_{k\in\tilde{\mathcal{K}}} z_{jk} = 1, \qquad j \in \mathcal{J}, \tag{15b}$$

$$z_{jk} \le \lambda_{jk}, \qquad j \in \mathcal{J},\ k \in \tilde{\mathcal{K}}, \tag{15c}$$

$$w_{jku} + w_{qku} \le 1, \qquad j, q \in \mathcal{J}, j \ne q, k \in \tilde{\mathcal{K}}, u \in \mathcal{T}, \tag{15d}$$

$$w_{jku} \le z_{jk}, \qquad j \in \mathcal{J},\ k \in \tilde{\mathcal{K}},\ u \in \mathcal{T}, \tag{15e}$$

$$\sum_{k\in\tilde{\mathcal{K}}}\sum_{u\in\mathcal{T}} w_{jku} = \tilde{p}_j, \qquad j \in \mathcal{J}, \tag{15f}$$

$$\sum_{k\in\tilde{\mathcal{K}}}\left(\tilde{p}_j w_{jku} - \tilde{p}_j w_{jk,u+1} + \sum_{\nu=u+2}^{T} w_{jk\nu}\right) \le \tilde{p}_j, \qquad j \in \mathcal{J}, u = 0, \ldots, T-2, \tag{15g}$$

$$\sum_{k\in\tilde{\mathcal{K}}}\left(w_{jku} + w_{jk,u+1} + \sum_{\nu=u+\tilde{v}^{\mathrm{pm}}_{jq}}^{T} w_{qk\nu}\right) \le \tilde{p}_q + 1, \quad (j, q) \in \mathcal{Q}, u = 0, \ldots, T - \tilde{v}^{\mathrm{pm}}_{jq}, \tag{15h}$$

$$\sum_{k\in\tilde{\mathcal{K}}}\sum_{u=0}^{\tilde{r}_j} w_{jku} = 0, \qquad j \in \mathcal{J}, \tag{15i}$$

$$\sum_{j\in\mathcal{J}}\sum_{u=0}^{\tilde{a}_k} w_{jku} = 0, \qquad k \in \tilde{\mathcal{K}}, \tag{15j}$$

$$u\sum_{k\in\tilde{\mathcal{K}}}(w_{jk,u-1}-w_{jku})\le s_j,\qquad j\in\mathcal{J},u=1,\dots,T, \tag{15k}$$

$$h_j\ge s_j-\tilde{d}_j,j\in\mathcal{J}, \tag{15l}$$

$$h_j\ge 0,\qquad j\in\mathcal{J}, \tag{15m}$$

$$s_j\ge 0,\qquad j\in\mathcal{J}, \tag{15n}$$

$$w_{jku}\in\{0,1\},\ j\in\mathcal{J},\ k\in\tilde{\mathcal{K}},\ u\in\mathcal{T}, \tag{15o}$$

$$z_{jk}\in\{0,1\},\ j\in\mathcal{J},\ k\in\tilde{\mathcal{K}}. \tag{15p}$$

The plateau variables w_{jku} are related to the time-indexed model's nail variables x_{jku} as follows:

$$w_{jku}=\sum_{\nu=(u-p_j+1)_+}^{u}x_{jk\nu},\quad j\in\mathcal{J},\ k\in\tilde{\mathcal{K}},\ u\in\mathcal{T}. \tag{16}$$

The variables w_{jku} are hence positive linear combinations of the nail variables. The allocation variables, z_{jk}, are given by (12a) in the previous section.

Conversely, the nail variables can be expressed in terms of the plateau variables as

$$x_{jku}=\begin{cases}\max\{0,w_{jku}-w_{j,k,u-1}\}, & u=1,\dots,T,\\ w_{jku}, & u=0,\end{cases}\qquad j\in\mathcal{J},k\in\tilde{\mathcal{K}}. \tag{17}$$

Like the constraints (11b) and (11c), the constraints (15b) and (15c) ensure that every machining operation is processed exactly once, and that each operation is scheduled on an allowed resource. The constraints (15d) make sure that no two operations $(2,j)$ and $(2,q)$ are simultaneously scheduled in the same resource, as do the constraints (11d) in the time-indexed model with nail variables. The constraints (15e) relate the plateau variables w_{jku} to the allocation variables z_{jk}, so that $w_{jku}=0$ for all resources k but the one chosen for the job j, i.e., the resource for which $z_{jk}=1$. The constraints (15f)–(15g) ensure that there are exactly $\tilde{p}_j$ 1-valued plateau variables and that these appear consecutively along the time axis. The constraints (15h), which correspond to the constraints (11e), separate the machining operations of the jobs j and q, where $(j,q)\in\mathcal{Q}$, by the planned interoperation time $\tilde{v}_{jq}^{\mathrm{pm}}$. Analogously to the constraints (11j), the constraints (15i) and (15j) ensure, respectively, that no machining operation is scheduled before its release date or in a resource that is not available.

The value of $(w_{jk,u-1}-w_{jku})$ in the constraints (15k) equals -1 when u is the starting time of operation $(2,j)$; it equals 1 when u is the operation's completion time. Therefore, the completion time s_j cannot be defined by using an equality in contrast to the models (7) and (11). Denote by $\bar{t}_j$ and $\bar{s}_j$ the values of the starting and completion times, respectively, of a job j in an optimal solution. Using the optimal values of the plateau variables, the left hand side of (15k) then equals $-\bar{t}_j$ and $\bar{s}_j$ for $u=\bar{t}_j$ and $\bar{s}_j$, respectively; it equals 0 for $u\in\mathcal{T}\setminus\{\bar{t}_j,\bar{s}_j\}$. Hence, the largest value of any left hand side of (15k) will always be equal to the completion time of job j. Since the minimization of the sum of the completion times is part of the objective function (15a), the variables s_j will attain their respective correct values.

5.4.6 Minimizing work load variance

There are many views of what defines a good schedule, and one has to analyze the current logistic environment in order to determine what suits best the current situation. The objective functions presented in the preceding formulations of the scheduling problem focus on scheduling the jobs as soon as they are available and on minimizing the total tardiness. This way the lead times within the cell are minimized. But since the processing in the multitask cell is a part of the whole production process, the total lead time of the parts processed in the cell will probably not decrease much if some parts are produced too early, due to the objective of minimizing total completion times and total tardiness in the multitask cell. Another disadvantage of this objective function is that it produces schedules in which short jobs are typically processed before longer jobs. This is in fact the property of the well known dispatching rule shortest-processing-time (SPT) (of the theory of production logistics) mentioned in Section 3.2. However, if parts of the same type arrive in clusters to the next operation (e.g. the operation μ_{m_j+1} being the operation to be performed directly after job j in Figure 8), there is a risk for long queues before this operation which may result in longer total lead times for the product.

One alternative to the objective of minimizing the sum of completion times and total tardiness is to produce a schedule in which the instant work load being as close to a target work load as possible, while the objective term of weighted total tardiness is retained. Hence, the objective will be to minimize the weighted sum of the sum of the absolute differences between the actual work load and the target work load in each time step of the main planning period and the total tardiness. With the plateau variable w_{jku}, whose value equals 1 if resource k is occupied by job j during time interval u, a linear approximation of the work load in the multitask cell can be formulated. The time-indexed model with nail variables can also be used with this objective, replacing w_{jku} below by the expression given in (16).

The work load per time unit during time interval u of length ℓ is given by the expression $\ell^{-1} \sum_{j \in \mathcal{J}} \sum_{k \in \tilde{\mathcal{K}}} w_{jku}$. Letting the target work load per time unit for the planning period be denoted by $b_u^{\texttt{target}}$, the objective function can be expressed as that to

$$\text{minimize} \quad A \sum_{u=0}^{T^{\texttt{plan}}} \left| \frac{1}{\ell} \sum_{j \in \mathcal{J}} \sum_{k \in \tilde{\mathcal{K}}} w_{jku} - b_u^{\texttt{target}} \right| + B \sum_{j \in \mathcal{J}} h_j, \tag{18}$$

where $T^{\texttt{plan}} \leq T$ is the end of the *main* planning period; the value of $T^{\texttt{plan}}$ is less than the value of T, since the planning horizon T must allow all jobs to be scheduled, and hence there needs to be a period with declining work load towards the end of the total planning period; see Figure 13.

In the example schedule in Figure 14, the length of the planning horizon is $T = 70$ and a suitable choice for the length of the main planning period is $T^{\texttt{plan}} = 40$, since there is a decline in the work load after this point in time.

In order to retain a linear objective function, we introduce the two continuous non-negative variables b_u^- and b_u^+; their sum is defined to equal the absolute difference between the instant work load and the target work load in time interval u. The

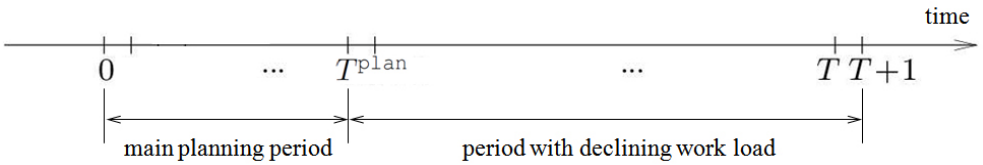

Figure 13: *The main planning period includes the time steps 0 through T^{plan}, after which the work load (typically) is declining.*

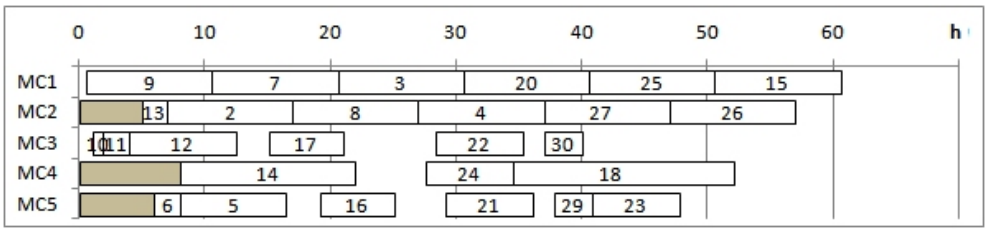

Figure 14: *An example of an optimal schedule based on real data for the five processing machines. A suitable choice for the length of the planning period for this case is $T^{plan} = 40$.*

variables b_u^- and b_u^+ are constrained by the inequalities

$$-\frac{1}{\ell}\sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}} w_{jku} + b_u^{\texttt{target}} \le b_u^-, \qquad u = 0,\dots,T^{\texttt{plan}}, \tag{19a}$$

$$\frac{1}{\ell}\sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}} w_{jku} - b_u^{\texttt{target}} \le b_u^+, \qquad u = 0,\dots,T^{\texttt{plan}}, \tag{19b}$$

$$b_u^- \ge 0, \qquad u = 0,\dots,T^{\texttt{plan}}, \tag{19c}$$

$$b_u^+ \ge 0, \qquad u = 0,\dots,T^{\texttt{plan}}. \tag{19d}$$

The objective function can then be formulated as to

$$\text{minimize} \quad A\sum_{u=0}^{T^{\texttt{plan}}}(b_u^- + b_u^+) + B\sum_{j\in\mathcal{J}} h_j. \tag{20}$$

The scheduling of the multitask cell while minimizing the work load variance is then given by the objective (20) subject to the constraints (15b)–(15p) and (19a)–(19d).

Energy consumption, or energy costs related to machine runtimes, can be minimized in the same way as the work load, since the work load can be viewed as the accumulated runtime of the machines in the cell. This objective has not been much studied, and Fang et al. [13, 2011] claim to be the first to consider shop scheduling with energy- and environment-related constraints. It may, however, increase in importance under the pressure of global climate change and rising energy costs.

5.4.7 A heuristic for determining the time horizon

The number of decision variables, x_{jku} and w_{jku}, respectively, of the time-indexed models (nail and plateau) risks being very large, since it depends on the length ℓ of the time intervals, as well as the time horizon, T. This is also mentioned in [60]. In addition, the number of constraints in, e.g., (11d) and (11e), depends on the value of T. Therefore, it is important to find a value of T small enough not to cause, for example, memory shortage or too long computation times, due to the resulting large number of variables and constraints, but large enough such that the time interval $[0, \dots, T]$ contains an optimal schedule for the problem instance. In order to find an appropriate value of T, we have constructed a greedy heuristic which also finds a feasible schedule; see Algorithm 1.

We let τ_k denote the time from which the resource k is available, $r_j^{\texttt{heur}}$ the release date for job j, and $waiting\_list$ a list of jobs not scheduled instantly in the heuristic. Further, we denote by $\mathbf{T}$ a time horizon that is large enough that the time interval $[0, \mathbf{T}\ell]$ can contain the heuristic schedule; for example, $\mathbf{T} = |\mathcal{J}| p_j^{\max}$. The starting time for each job j is denoted by t_j, as in the engineer's model (7).

Algorithm 1 is initialized by a sorting of the jobs according to increasing values of the release dates, i.e., defining $j^{\max} = |\mathcal{J}|$, such that $r_j \leq r_{j+1}$, $j \in \mathcal{J} \setminus j^{\max}$. After the initialization, in which $x_{jku}, \tau_k, r_j^{\texttt{heur}}$, and t_j are assigned initial values, the algorithm loops over all jobs $j \in \mathcal{J}$. For each job, j, the first available resource $\tilde{k}$ allowed is chosen and job j is scheduled in this resource either directly when the resource is available, or at the job's release date, i.e., at $\max\{\tau_k, r_j^{\texttt{heur}}\}$. Then there is a check whether the job just scheduled equals q, where q fulfills $(\hat{\jmath}, q) \in \mathcal{Q}$ for some $\hat{\jmath} \in \mathcal{J}$. If this is the case and if $q \neq j^{\max}$, and if the required time $v_{\hat{\jmath}q}^{\texttt{pm}}$ has not passed since the starting time of the preceding job in the pair, $t_{\hat{\jmath}}$, it is "descheduled". The job q that has just been descheduled is also included in the $waiting\_list$ and the release date, $r_q^{\texttt{heur}}$, is assigned the realistic value $t_{\hat{\jmath}} + v_{\hat{\jmath}q}^{\texttt{pm}}$, since the time $v_{\hat{\jmath}q}^{\texttt{pm}}$ must elapse from the starting time of job $\hat{\jmath}$ until job q may be scheduled. After this step, the heuristic calls the recursive function Algorithm 2 which updates the release dates of all the succeeding jobs for the corresponding physical part, i.e., the release dates for q_i, $i = 1, \dots, m$, where $q_0 = q$ and $(q_{i-1}, q_i) \in \mathcal{Q}, i = 1, \dots, m$.

After updating all appropriate release dates, Algorithm 1 searches through all jobs in the waiting list; a job is scheduled if its release date is earlier or at the time when the chosen resource, $\bar{k}$, is available. This is the first available allowed resource for the job, and the time when it is available is denoted by $\tau_{\bar{k}}$. If the last job, $j^{\max}$, in the loop over all jobs in the set $\mathcal{J}$ is scheduled, all the remaining jobs in the waiting list are scheduled at the first available allowed resource at their respective release dates. At the exit of the loop, all jobs have been scheduled in a feasible way, and the completion time of the latest job in the schedule is the sought value of the makespan.

Since the heuristic only considers the completion times and not the tardiness, which is actually the most important part of the objective function (11a), T is assigned the value of the makespan found by Algorithm 1 plus $2p_j^{\max}$, where $p_j^{\max} = \max_{j \in \mathcal{J}} p_j$. For most cases (indeed all the real-world cases that we have tested) an optimal schedule is contained in the set $\{0, \dots, T\}$ of time steps. It may be possible to

Algorithm 1 Heuristic algorithm

$x_{jku} \leftarrow 0, \tau_k \leftarrow a_k, r_j^{\texttt{heur}} \leftarrow r_j, t_j \leftarrow \mathbf{T}, \quad j \in \mathcal{J}, k \in \tilde{\mathcal{K}}, u \in \{0, ..., \mathbf{T};\}$
for $j \in \mathcal{J}$ **do**
  find $\tilde{k} \in \operatorname{argmin}_{k \in \tilde{\mathcal{K}}} \{ \tau_k \mid \lambda_{jk} = 1 \}$; # earliest available allowed resource
  $t_j \leftarrow \max\{\tau_{\tilde{k}}; r_j^{\texttt{heur}}\}$; $x_{j\tilde{k}t_j} \leftarrow 1$; $\tau_{\tilde{k}} \leftarrow t_j + p_j$; # job j is scheduled
  for $(\hat{\jmath}, q) \in \mathcal{Q}$ **do**
    if ($j \neq j^{\max}$ **and** $j = q$) **then**
      if ($q \notin waiting\_list$ **and** $t_q < t_{\hat{\jmath}} + v_{\hat{\jmath}q}^{\texttt{pm}}$) **then**
        $\tau_{\tilde{k}} \leftarrow t_q$; $x_{q,\tilde{k},t_q} \leftarrow 0$; $t_q \leftarrow \mathbf{T}$; # job q is "descheduled" and...
        $waiting\_list \leftarrow waiting\_list \bigcup \{q\}$; # ...put in the $waiting\_list$
        $r_q^{\texttt{heur}} \leftarrow t_{\hat{\jmath}} + v_{\hat{\jmath}q}^{\texttt{pm}}$; # $r_j^{\texttt{heur}}$ updated
        Update_release_dates(q); # $r_{\bar{q}}^{\texttt{heur}}$ for succeeding jobs updated
      else if $q \in waiting\_list$ **then**
        if $t_q < t_{\hat{\jmath}} + v_{\hat{\jmath}q}^{\texttt{pm}}$ **then**
          $\tau_{\tilde{k}} \leftarrow t_q$; $x_{q\tilde{k}t_q} \leftarrow 0$; $t_q \leftarrow \mathbf{T}$;
        else
          $waiting\_list \leftarrow waiting\_list \setminus \{q\}$; # no need for rescheduling
        end if
      end if
    else if $q \in waiting\_list$ **then**
      find $\bar{k} \in \operatorname{argmin}_{k \in \tilde{\mathcal{K}}} \{ \tau_k \mid \lambda_{qk} = 1 \}$; # try to schedule the second entry q...
      if $r_q^{\texttt{heur}} \leq \tau_{\bar{k}}$ **then**
        $t_q \leftarrow \tau_{\bar{k}}$; $x_{q,\bar{k},t_q} \leftarrow 1$; $\tau_{\bar{k}} \leftarrow t_q + p_q$; # ...of any pair $\in waiting\_list$
        $waiting\_list \leftarrow waiting\_list \setminus \{q\}$;
      end if
    else if $j = j^{\max}$ **then**
      $t_q \leftarrow r_q^{\texttt{heur}}$; $x_{q,\bar{k},t_q} \leftarrow 1$; $\tau_{\bar{k}} \leftarrow t_q + p_q$; # schedule remaining jobs
      $waiting\_list \leftarrow waiting\_list \setminus \{q\}$;
    end if
  end for
end for
$T \leftarrow \max_{k \in \tilde{\mathcal{K}}}\{\tau_k\} + 2p_j^{\max}$ # T is the makespan found plus $2p_j^{\max}$.

Algorithm 2 Update_release_dates(q)

for $(\bar{\jmath}, \bar{q}) \in \mathcal{Q}$ **do**
  if $\bar{\jmath} = q$ **then**
    $r_{\bar{q}}^{\texttt{heur}} \leftarrow r_{\bar{\jmath}}^{\texttt{heur}} + v_{\bar{\jmath}\bar{q}}^{\texttt{pm}}$;
    $waiting\_list \leftarrow waiting\_list \bigcup \{\bar{q}\}$;
    Update_release_dates($\bar{q}$);
  end if
end for
return

construct an instance for which this is not the case, but since we are dealing with real case data, the risk of receiving such an instance in the multitask cell is very small. We have decided to take this risk, since the existence of feasible solutions within the time interval $[0, \ldots, (T+1)\ell]$ is guaranteed.

To give an example of an instance for which the makespan of the schedule that minimizes the sum of the total completion time and the total tardiness equals that of the makespan found by the Algorithm 1 plus $p_j^{\max}$, we have constructed an instance with two resources, M/C1–2 (see Figure 15) where all jobs are available from the start ($r_j = 0$).

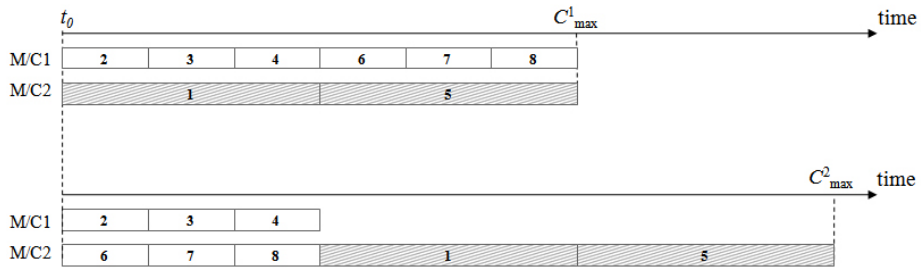

Figure 15: The upper schedule is found by Algorithm 1 and has a makespan of $C^1_{\max} = 12$. The lower schedule is an optimal schedule with respect to the objective (11a), with makespan $C^2_{\max} = 18$.

This is a very special constructed case, since it requires that the first of the longer jobs is indexed $j = 1$ and the index for the second is $j \in \{2, \ldots, 5\}$, say $j = 5$; see Table 6 for the output data of the two schedules. It also requires that the longer jobs (having $p_j = 6$, $j \in \{1, 5\}$) have no tardiness in neither of the two schedules and that they are only allowed to be scheduled in the resource M/C2 ($d_j = 20$, $\lambda_{j,\text{M/C1}} = 0$, and $\lambda_{j,\text{M/C2}} = 1$, $j \in \{1, 5\}$). All of the shorter jobs (having $p_j = 2$, $j \in \{2, 3, 4, 6, 7, 8\}$) have to be tardy and they must also be allowed to be scheduled on both resources ($d_j = 0$ and $\lambda_{jk} = 1$ for $k \in \{\text{M/C1, M/C2}\}$ and $j \in \{2, 3, 4, 6, 7, 8\}$).

6 Computational tests and results

The aim of the computational tests described in this section is to explore which model performs the best, both regarding computation times and the solution quality. Even though all models presented in this thesis have been subject to computational tests, the results presented in this section emanate from three of the models, namely the engineer's model (7), the feasibility model (9), and the time-indexed model with nail variables (11). The full engineer's model required computation times that were too long for practical purposes, and was therefore used to a very small extent. The time-indexed model with plateau variables (15), performed slightly better than the engineer's model in the tests. The time-indexed model with nail variables, however,

	Schedule found by Algorithm 1				Optimal schedule			
j	k	s_j	h_j	$s_j + h_j$	k	s_j	h_j	$s_j + h_j$
1	M/C2	6	0	6	M/C2	12	0	12
2	M/C1	2	2	4	M/C1	2	2	4
3	M/C1	4	4	8	M/C1	4	4	8
4	M/C1	6	6	12	M/C1	6	6	12
5	M/C2	12	0	12	M/C2	18	0	18
6	M/C1	8	8	16	M/C2	2	2	4
7	M/C1	10	10	20	M/C2	4	4	8
8	M/C1	12	12	24	M/C2	6	6	12
$\sum_{j=1}^{8}$		60	42	102		54	24	78

Table 6: Data describing the schedules illustrated in Figure 15.

outperformed the model (15) (with plateau variables), therefore, the latter model was not further investigated.

The computational results are given for the models (7) and (11) for real data instances collected from the ERP system at Volvo Aero in the year of 2010. The computations regarding the comparisons with the dispatching rules in Section 6.2.3 were carried out on a 4Gb quad-core Intel Xeon 3.2 GHz system using AMPL-CPLEX12 as optimization software. All other computations have been carried out using AMPL-CPLEX12 on a computer with two 2.66GHz Intel Xeon 5650, each having six cores (24 threads); its total memory was 48Gbyte RAM.

6.1 Test data

All data used in all tests described in this section originate from real production instances in the multitask cell. One of the problems encountered due to the choice of using real data is that planned orders that have not yet been started according to the plan may be present in the system. Then, the release dates of all planned orders related to the same product that are currently delayed (calculated by the formula (1) given in Section 5.1.4) will have the same value, namely that of ν_j^0, i.e., the planned lead time—illustrated in Figure 8—from the release of the order into the production until its arrival to the multitask cell. It is, however, not plausible that all planned orders that are currently delayed, will be released at time t_0; hence, these orders—if they exist—have to be (manually) assigned plausible release dates before the calculation of an optimal schedule is made in an implementation. Such plausible release dates have not been set for the instances used for obtaining the results given in this section, and hence some jobs of the same kind may share release dates.

6.2 Validation and tests

As mentioned in Section 5.1.5 it is important that the value of the weight ε in the objective function (9a) chosen for the feasibility problem is much smaller than those of A and B. We have chosen the values $A = B = 1$ and $\varepsilon = 0.001$ in all the computations presented in this section. The choice of having $A = B = 1$ was made since the two goals, *short completion times* and *low tardiness*, are not contradictory. However, when studying some test results, we discovered some cases where jobs with no tardiness, but short processing times, were scheduled in clusters before longer tardy jobs. This indicates that the choice of $A = B$ was unfortunate; finding appropriate values of these weights is an area of future study.

The influence of the size of the discretization interval ℓ for the time-indexed model (11) with nail variables on the solution quality and the computation times is described in Sections 6.2.1 and 6.2.2, respectively. In these sections, results from the engineer's model are also given. In Section 6.2.3, the results from the application of the model (11) are compared with schedules resulting from the use of dispatching rules, and in Section 6.2.4, the influence on computation times of the choice of the value for the parameter M for the engineer's model is discussed. A summary of the results is given in Section 7.

6.2.1 Estimated error due to the choice of ℓ for the time-indexed models

The starting and completion times obtained from the optimal solutions to the time-indexed model (11) with nail variables, are given in terms of multiples of the length ℓ of the time discretization interval. Therefore, the optimal objective value of (11) differs from that of the engineer's model (7). Thus the completion times s_j are recalculated using the original (i.e., non-discrete) data, while retaining the ordering of the operations on each of the processing machines received from the solution to the model (11). For the example illustrated in Figure 16, the discrete release date of job 4 is 18h, while the real release date is 17.5h, which is the reason for the idle period between jobs 2 and 4 in the schedule. The value of the solution obtained after this post-processing can then be compared with the optimal value from the engineer's model (7). This post-processing of data required at most 0.04s of computation time for each of the six scenarios constructed from the data set described below, comprising 70 jobs.

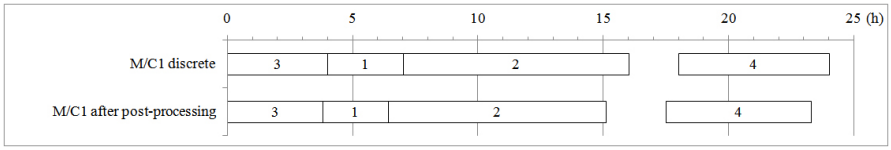

Figure 16: The output of the discrete time machining model is post-processed using the original non-discrete data.

Results from the engineer's model and the time-indexed model with different values of ℓ have been compared for six real scenarios collected from the multitask cell during the autumn of 2010. For each scenario, the jobs were initially sorted according to increasing values of the release dates, r_j. Different test instances were then created by selecting the first $5m$ jobs in the queue of jobs, where $m \in \{1, \ldots, 14\}$, such that the smallest and largest instances comprise 5 and 70 jobs, respectively.

A limit of 7200s was set for the computation time (clocktime), since 2h is in the upper limit of what could be accepted in practice. Fortunately, only a few of the larger instances required such long computation times, at least in the tests performed with the time-indexed model with nail variables. The limit of 7200s was, however, reached when using the engineer's model; the calculations were then disrupted for some scenarios with 20 jobs, and no optimal solutions were obtained for any scenario with $|\mathcal{J}| \geq 25$. The time-indexed model performed, however, better and we obtained optimal solutions for all instances with $|\mathcal{J}| \leq 60$ and $\ell \geq 1$h before the time limit was reached; see Section 6.2.2 for details on these computations.

In Table 7, the mean values of the relative differences between the optimal values from different models are listed. Since optimal values were only obtained for all scenarios with $|\mathcal{J}| \leq 15$ when using the engineer's model, it was solely compared to the result from the time-indexed model with $\ell = 0.25$h; results are missing for $|\mathcal{J}| \geq 20$. The optimal values obtained using the engineer's model were identical with those obtained using model (11) for all instances with 10 and 15 jobs. All the other optimal solutions found by the time-indexed model with $\ell > 0.25$h were compared to those found by this model with $\ell = 0.25$h.

# jobs	Diff to (7) $\ell = 0.25$h	Diff to (11) $\ell = 0.5$h	Diff to (11) $\ell = 1$h	Diff to (11) $\ell = 2$h
10	0.000	0.000	0.000	0.031
15	0.000	0.005	0.010	0.083
20	–	0.001	0.018	0.061
25	–	0.015	0.028	
30	–	0.015	0.034	0.112
35	–	0.019	0.022	
40	–	0.022	0.027	0.172
45	–	0.022	0.025	
50	–	0.028	0.035	0.335
Mean	0.000	0.014	0.022	0.132

Table 7: Mean differences between the optimal values from different models. The time-indexed model (11) with $\ell = 0.25$h is compared with the engineer's model (7), while the time-indexed model with $\ell > 0.25$h is compared with the same model with $\ell = 0.25$h. The differences are given in percent with regard to the optimal value from the model compared. The notation – means that the corresponding results are missing.

The results listed in Table 7 are illustrated in Figure 17. The error seems to increase with the number of jobs for the model (11) with $\ell = 2$h, while the result for the other models with $\ell \in \{0.25\text{h}, 0.5\text{h}, 1\text{h}\}$ stay close to eachother. The mean relative error is less than 0.04% for all of the instances computed. This is remarkable since the smallest processing time for any of the operations in the multipurpose machines is 0.6h (the largest is 22.4h). Note that all the relative differences from the respective optimal values are very small in terms of percentage, but since the total tardiness is large for some scenarios, the model with $\ell = 2$h differs significantly from the others in terms of absolute differences (measured in hours).

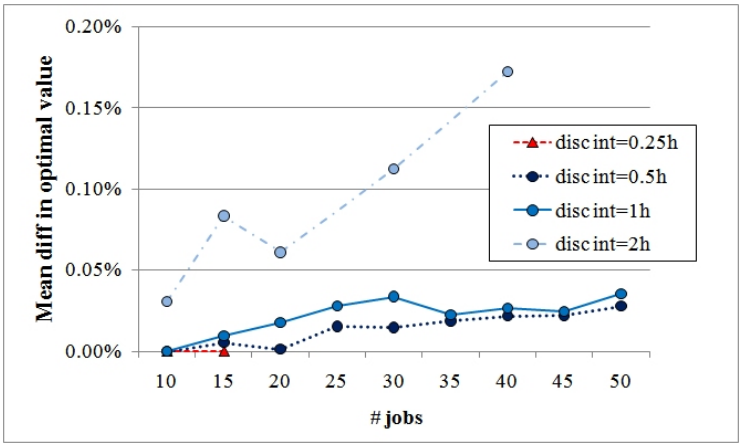

Figure 17: The mean differences between relative values of the optimal solutions found using the model (10) with $\ell \in \{0.25\text{h}, 0.5\text{h}, 1\text{h}, 2\text{h}\}$ the models.

6.2.2 Comparison of computation times for the different models

The multitask cell comprises about 30 storage places for parts checked-in but not yet mounted into a fixture, which means that there can be at most 30 jobs with release date $r_j = 0$, and which all must be taken into account when searching for an optimal schedule for the coming shift. Some additional parts are expected to arrive at the multitask cell during this shift and they also need to be taken into account. We estimate the number of parts arriving at the cell during the coming shift to be at most 15, and hence it is realistic to create detailed schedules for the coming shift for instances with $|\mathcal{J}| \leq 45$.

In Figure 18 mean CPU times are plotted for each model tested as a function of the size of the instance. The mean values are calculated over all six real data scenarios and only the cases for which the optimal values were found for all of the six scenarios are plotted in the figure.

As already mentioned in Section 6.2.1, the time limit of 7200s for the computations was reached—and the calculations interrupted—when using the engineer's

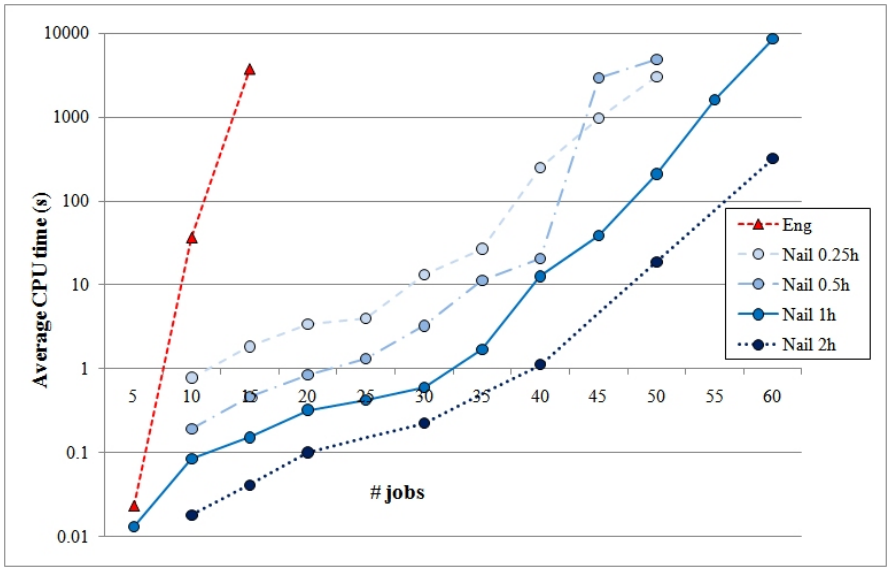

Figure 18: *Mean CPU times plotted for the engineer's model* (7) *(eng) and the time-indexed model* (11) *with nail variables for different values* ℓ *of the discretization interval. The instances computed are represented by the markers in the graph. The mean values are calculated over six real data instances.*

model (7) for solving some scenarios with 20 jobs. The time-indexed model (11) with nail variables performed better; see Figure 18. We verify, for all of the instances but two, the anticipated relation that the computation times grow with a decreasing value of ℓ. The model (11) with $\ell = 0.5$h had trouble finding an optimal solution for one scenario with $|\mathcal{J}| \geq 45$. Hence, one should choose the value of ℓ as large as possible in order to possess short computation times, but also as small as possible in order to get solutions of good quality. From Section 6.2.1 we know that the differences in optimal values between the model (11) with $\ell = 1$h and $\ell = 0.25$h were very small for the test set. Hence $\ell = 1$h seems to be a good choice for the value of this parameter.

The model (11) with $\ell = 1$h required on average 40 CPU-seconds to find an optimal solution for scenarios with 45 jobs, which we have estimated to be the largest size of a realistic instance for computing an optimal schedule for the coming shift. This model succeeded in finding an optimal schedule for two out of the six scenarios with $|\mathcal{J}| = 70$, one within only 18.5 CPU-seconds, and the other within 16832.3 CPU-seconds (20 minutes of clocktime). The other instances were disrupted after 7200 seconds (clocktime), where the maximum *relative mipgap* (the maximum relative difference between the objective function found and the optimal value) was 0.055%.

Although the model (11) with $\ell = 2$h was shown to yield a bigger error than did

the other models tested in Section 6.2.1, it may be used as a tool for computing schedules for a rough long-term planning, since it is able to compute quite big instances within a reasonable amount of time.

6.2.3 Comparison with dispatching rules

In [55] (Paper I in this thesis) and [52] the results from comparisons between the schedules found by the time-indexed model (11) with $\ell = 1$h and the feasibility model (9) with the two dispatching rules EDD (earliest–due–date) and FIFO (first–in–first–out) are presented. The comparisons are made for a mix of real and constructed scenarios in order to simulate different logistic situations.

In this section the results from a similar comparison using data from 21 real production instances with $|\mathcal{J}| = 20$ collected during the period April–August 2010 are presented. These results were not available at the time when the articles [55] and [52] were written and are therefore presented here. The distribution over the weekdays and hours when the samples of data have been extracted from the ERP system is illustrated in Figure 19.

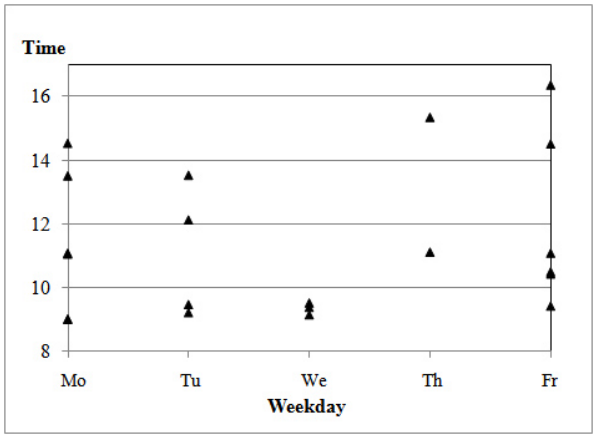

Figure 19: *Distribution in time of the collection of the 21 real production instances over weekdays and hours. Two data points collide at each of the hours 9 and 11 within the Monday samples.*

The data collected contain information about which parts are checked-in at the multitask cell, which parts are on their way to the cell, and which are the planned orders at the time of sampling. The system release and due dates are also extracted, together with information about the current operation of the parts on their way to the cell. The schedules produced by the dispatching rules EDD, FIFO, and shortest–processing–time (SPT) were compared to the optimal schedules found by the machining model (11) solved in sequence with the feasibility model (9).

The role of the feasibility model (9) is here not only to create feasible schedules, but also to post-process the discretized data in a similar manner as described in Section 6.2.1, where a comparison of the outcomes of the different models for the machining problem is performed. The dispatching rules were used to produce schedules for the machining resources. The resulting sequences of jobs for each resource were used as input to the feasibility model (9) in order to create feasible schedules for the whole multitask cell.

As the gain of using an exact mathematical optimization model is expected to be the most profitable at times of high work load, it is interesting to distinguish these instances. The scenarios were unfortunately collected during a time with quite low production volumes. Nethertheless, during the collection period the number of jobs checked-in, let us denote it by J, i.e., the number of jobs j possessing $r_j = 0$, varied from three to more than 20 jobs (see Figure 20 and Section 5.1.4 for the definition of r_j).

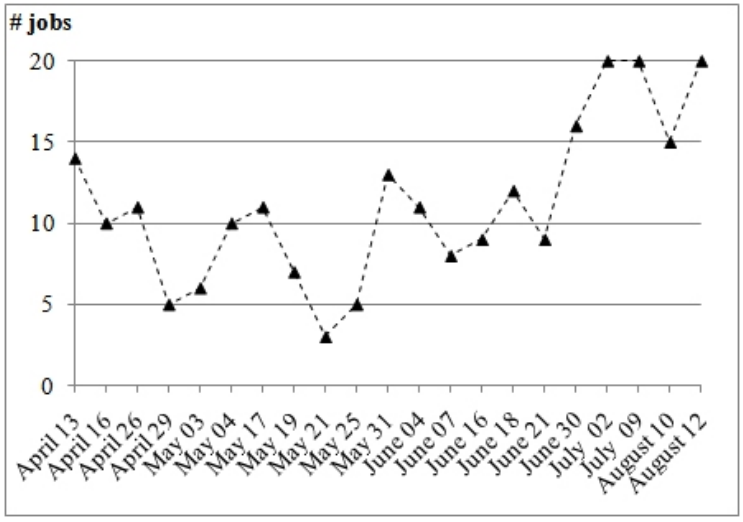

Figure 20: *The number of jobs, J, checked-in, i.e., jobs with $r_j = 0$, versus the collection dates. The differences between $|\mathcal{J}| = 20$ and J consist of the jobs with the earliest release dates emanating from parts on their way to the multitask cell and planned orders.*

The multitask cell does not necessarily possess a heavy work load at times with a lot of parts checked-in that are ready to be processed at time t_0. However, this is often the case. Therefore, we have divided the scenarios into two groups, one with $J \leq 10$ and the other with $J > 10$ comprising 10 and 11 scenarios, respectively.

The result from the comparison is illustrated in Figure 21. The sums of the completion times and tardiness for all 20 jobs in the schedules resulting from using the dispatching rules as input data to the feasibility model (9) has been compared to

those found by the same model (9) using the optimal results from the model (11) with $\ell = 1$h as input data.

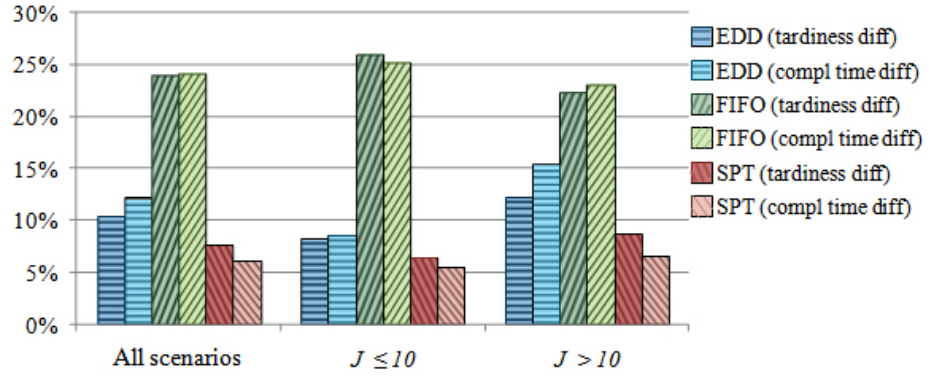

Figure 21: *The mean differences between the sums of the completion times and tardiness in schedules constructed by the dispatching rules EDD, FIFO, and SPT, respectively, and those found when solving model* (11) *with* $\ell = 1$h *to optimality in sequence with the feasibility model* (9). *The mean values are calculated over all 21 scenarios, and for the scenarios with* $J \leq 10$ *and* $J > 10$, *respectively. The differences are measured in percentage of the optimal values found by the model* (9).

As expected, the gain of using mathematical optimization is greater for the instances with $J > 10$. The use of the FIFO dispatching rule, which is quite common in practice, results, on average, in at least 22% higher tardiness and completion times compared to corresponding values for the the optimal schedules found by mathematical optimization at times of high work loads. We remark that the SPT rule—which does not take the due dates into account when giving priorities to the jobs—and the EDD rule yield both completion times and tardiness that are on average 6% and 10% higher, respectively, than the corresponding mean values from the optimal schedules.

6.2.4 The value of the parameter M for the engineer's model

The parameter M defined in Section 5.1.3 has to be assigned a value that is greater than the planning horizon, so that the schedule sought is contained within this time frame. Hence it is a large number, which may cause numerical problems in the solution process. Initially, we thought that we would have shorter computation times if we chose a value of M as small as possible, than if we just picked a very large number. Then we came across the article by Stafford et al. [46] which claimed the opposite, which is the reason for the tests described in this section. According to test results presented in [46] on some models of the Manne family for the permutation flow shop problem, the computation times decrease with an increasing value of M

(or P in their notation). They employed, however, as the smaller value, $M = |\mathcal{J}|p_j^{\max}$, which is a very large over-estimate of the planning horizon, since all 20 jobs then are assumed to have a processing time of $p_j^{\max}$. This formula for computing the planning horizon is also mentioned for a time-indexed model by Wolsey in [60].

We have performed a test with four real data instances, $I \in \mathcal{I}$, with 15 jobs each. The value of M was varied between 500 and 100,000 (see Table 8), and we denote each problem instance by (I, M). $M^{\texttt{heur}}$ is the value of M found by a heuristic, which has been constructed using the same logic as Algorithm 1 (described in Section 5.4.7), but with some adjustments in order to make it compatible with the engineer's model. The values of $M^{\texttt{heur}}$ for the instances in the test range between 75 and 103.

M	δ
$M^{\texttt{heur}}$	10^{-5}
500	10^{-6}
1000	10^{-6}
10000	10^{-7}
20000	10^{-8}
30000	10^{-8}
40000	10^{-8}
50000	10^{-8}
60000	10^{-8}
70000	10^{-8}
80000	10^{-8}
90000	10^{-8}
100000	10^{-9}

Table 8: The values of M forming the set $\mathcal{M}$, and the integrality tolerance, δ, chosen for each instance (I, M). The values of $M^{\texttt{heur}}$ for the instances in the test range between 75 and 103.

Since calculations with big numbers may cause rounding problems during the CPLEX computation, we have also varied a CPLEX parameter, called *integrality tolerance* in the CPLEX manual [26]. The default value of the integrality tolerance is 10^{-5}, which is too low for large values of M. The value of the integrality tolerance, $\delta > 0$, was chosen such that $M\delta \leq 0.001$, since the input data is given with two decimals; see Table 8.

When the integrality tolerance is set to the default value, the CPLEX solver reports "a non-integer solution" (the result reported from the solver is an infeasible solution) for some instances with large values of M. This behaviour was also reported by the authors of [43]; as occuring during the solution of a flow shop model.

From the limited test that we have performed, it is not possible to draw any general conclusions on which value of M that should be used. However, it seems like the engineer's model always performs well when the chosen value of $M = M^{\texttt{heur}}$; see Figure 22, where the normalized CPU times required for solving the instances to

optimality are plotted for different values of M. The normalization with respect to the mean computation time for each instance is defined by the formula

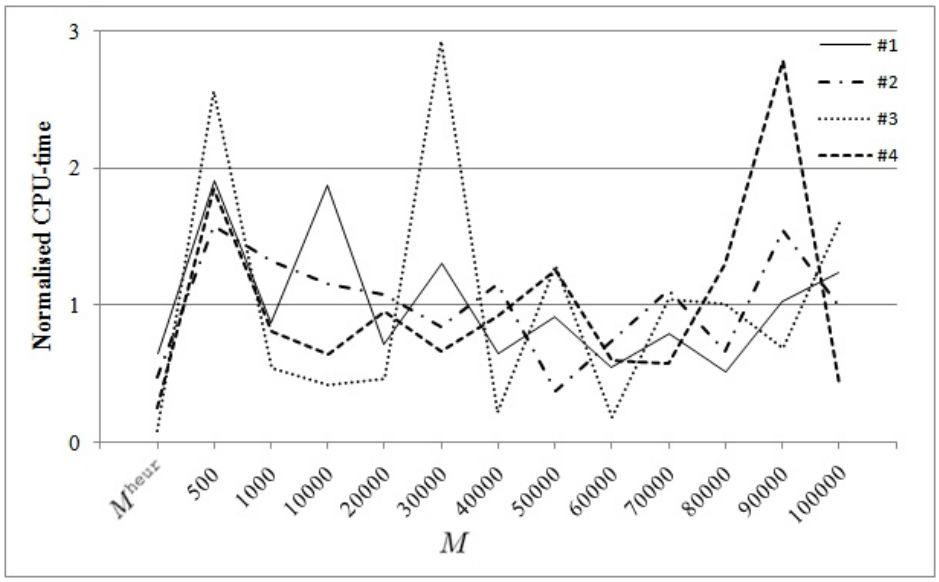

Figure 22: *CPU times normalised with the mean CPU time for each instance* (I, M) *(*$I \in \{\#1, \ldots, \#4\}$*) plotted for different values of* $M \in \mathcal{M}$*. Note that the values of* M *on the horizontal axis are not equidistant.*

$$\overline{\tau_{IM}} = \frac{|\mathcal{M}|\tau_{IM}}{\sum_{M \in \mathcal{M}} \tau_{IM}}, \qquad I \in \mathcal{I},\ M \in \mathcal{M},$$

where τ_{IM} denotes the CPU-time required for the solution of the instances (I, M), $\mathcal{M}$ is the set of values of M listed in Table 8, and $\mathcal{I}$ is the set of instances.

Note that for the case when $M = 500$, the computation times are long. This value of M is quite close to the value of $|\mathcal{J}|p_j^{\max}$, which is in the range $[210, 330]$ for all instances. Another reason for letting the heuristic find a value of M, (i.e., assigning an as low value as possible to M) is that the coefficient matrix corresponding to the linear constraints is better conditioned for lower values of M. Moreover, if the value of one of the variables y_{jqk} in the engineer's model (7) is slightly less than 1—due to the integrality tolerance—the term $M(1 - y_{jqk})$ in the right hand side of the constraints (7f) will attain a small positive value which may falsely indicate that an infeasible solution to the instance is feasible. Furthermore, with lower values of M, there is no need to reset the integrality tolerance for the CPLEX solver, (i.e., the default value can be used). Hence, for the results presented in this thesis emanating from the engineer's model (7), we have chosen to let M attain the value found by the heuristic.

7 Conclusions

The problem of scheduling the multitask cell is proven to be NP-hard in Section 4. Therefore, in the search for the most efficient way to solve the problem to optimality, the choices of variable representation and optimization model is of great importance.

In this thesis several mathematical optimization models for solving the problem of scheduling the multitask cell are developed and presented: three models within the Manne family (4), (7), and (9), one time-discrete model with so-called nail variables (11), and one time–discrete model with so-called plateau variables [(15)]. We have shown that model (11) outperforms the others by far with respect to computation times and the sizes of instances that they are able to solve using standard optimization software. To our knowledge, this is the first model presented for the flexible job shop problem which employs the type of variables that are here called nail variables. As mentioned in Section 3.1.1, most of the models for the job shop problem presented in scientific literature within operations research are members within the Manne family (see [33]), and hence many researchers have developed similar models (see, e.g. [40, 14, 44] and [63]), not being aware of the potential of using nail variables.

A heuristic employed to determine the planning horizon, T, for the time-indexed models has been developed and is presented in Section 5.4.7. The use of this heuristic shortens the computation times substantially for all time-indexed models, compared to the use of larger values of T, which are not specific for each instance.

Another parameter closely related to computation time is the length ℓ of the discretization interval. We have estimated both the error due to the choice of the value of ℓ and the impact of ℓ on the computation time, using data emanating from real production instances. For the current production in the multitask cell, we conclude, in Section 6.2.2, that $\ell = 1$h is a suitable choice. The model (11) with $\ell = 1$h required on average 40 CPU-seconds to find an optimal solution for scenarios with 45 jobs, which we estimate to be the largest size of a realistic instance for scheduling a coming shift. The relative error of the objective value will then be acceptable, with an average below 0.04%.

Another contribution of this thesis is the investigation of the effect on computation time of the choice of the value of the model parameter M (within a model of the Manne family), which has to be large enough such that one can fit an optimal schedule within the time interval $[0, M]$ for a specific instance. Our test results, presented in Section 6.2.4, show that if M is chosen by a heuristic similar to Algorithm 1, then the computation times will be low in comparison with the use of larger or much larger values of M. This is contradictory to the results presented in [46] by Stafford et al. They chose, however, a much larger value than ours as the smallest value in their test set, and, as it seems, did not consider the integrality tolerance employed by the software solver used. This tolerance needs to be reset for large values of M in order to guarantee that a feasible solution is obtained.

8 Future research

Currently, in order to find a feasible schedule for the complete multitask cell—irrespective of which of the models developed for the machining problem that is used—we need to employ the feasibility model (9). This model, which belongs to the Manne family, requires long computation times for instances with many jobs—too long to be acceptable in real applications. Since the time-indexed model (11) with nail variables performs very well, a subject for future research is to develop a time-indexed model either for the full problem, (that is, the problem of scheduling all resources in the multitask cell) or for the feasibility problem.

Another area for future research is to include the scheduling of fixtures, personnel, and tools, in a mathematical optimization model for the problem of scheduling the multitask cell; these resources are currently assumed to be unlimited, which is not the case in the real application.

When performing all of the computations reported in this thesis, we assume that the data are reliable. We calculate, for example, the estimated arrivals of jobs to the cell, and then, sometimes, assign these value to the corresponding release dates. Of course, we can not be certain that a part is going to be available at these release dates, but fortunately the uncertainty decreases the closer the part is to the multitask cell at time t_0. This is good since we intend to produce a new schedule daily or at the beginning of every shift. An area for future research is to construct a robust optimization model, which is less sensitive to data uncertainty. Other ideas are to find ways of measuring the unability to follow a certain schedule caused by data uncertainty, or to produce alternative schedules on short notice for the case of, e.g., machine break-downs.

As mentioned in Section 6.2, finding appropriate values of the weights of the components of the objective functions used in the computational testing is an area of future research. Instead of employing weighted objective functions, multiobjective optimization theory can be applied to this problem, which is another area to explore within future research.

We have observed during the computational testing that the time-indexed model with nail variables (11) yields good lower bounds. The work on finding an even stronger formulation of the problem of scheduling the multitask cell, and eventually so-called *facets* (the strongest formulation possible), is an ongoing master's thesis work at the department of Mathematical Sciences by a student from the University of Gothenburg. A study of the differences between the constraints (11e) and the alternative constraints (14) is an interesting subject for this ongoing work and for future research.

9 Summary of appended papers

Below follows a summary of the papers appended to this thesis. In addition to these two papers, I am the main contributor of [52], [54], and [51].

9.1 Paper I — Optimization of schedules for a multitask production cell

In this paper, we compare production plans resulting from utilizing an earlier version of the time-indexed model with nail variables with schedules formed by the first–in–first–out (FIFO) and earliest–due–date (EDD) priority rules, which are similar to the methods used in the current manual planning of the multitask cell. The problem of scheduling the multitask cell is presented and the engineer's model (7) is presented. The current production planning prerequisites have been studied in order to choose an appropriate objective function.

Six test scenarios have been used for the comparison. Three scenarios were created based on real production data from one day in March 2010. One scenario was left as it was, one was altered from the original scenario to include a larger proportion of short jobs, and the third was altered from the original scenario to include a larger proportion of long jobs. In these scenarios, all jobs are delayed at time $t_0 = 0$. Three other scenarios were created analogously, however based on a constructed scenario of a high volume case. It is a realistic case of a future product mix, created by the group planner working at the time in the multitask cell together with the master planner responsible for most of the products processed in the cell. In the three latter scenarios, approximately half of the jobs are already delayed at time $t_0 = 0$. Each scenario consists of 20 jobs, which all are assumed to be checked-in into the multitask cell, i.e., ready to be processed, at time $t_0 = 0$.

In a discussion towards the end of the paper, there is a small example comprising three jobs showing the complexity of the combinatorial scheduling problem at hand. We explain with this example how a priority rule will yield a schedule risking both tardiness for the job with the least priority and lost capacity in one of the two machines in the example. The optimization model will, however, yield a better schedule, which will make the multitask cell better prepared for the jobs yet to arrive. This is due to the fact that a priority rule only considers one job at a time, while the optimization model takes all jobs into account simultaneously.

Paper I is identical to [55] in the reference list.

9.2 Paper II — A note on the complexity of flow-shop scheduling with deteriorating jobs

This paper is a note on an article named *Complexity analysis of job-shop scheduling with deteriorating jobs* by G. Mosheiov [38], which claims to provide a complete analysis of the complexity of flow-shop, open-shop and job-shop problems. A proportional deterioration rate is assumed, that is, the job processing times are increasing

linear functions of the jobs' starting times. The objective is the minimization of the makespan. Mosheiov introduces a polynomial-time algorithm for the two-machine flow-shop and open-shop problems and presents NP-hardness results for flow-shops and open-shops with three or more machines and for job-shops with two or more machines. The proof of NP-hardness for the flow-shop case is however not correct. Paper II provides a correct proof.

Paper II is identical to [53] in the reference list.

References

[1] J. ADAMS, E. BALAS, AND D. ZAWACK, *The shifting bottleneck procedure for job shop scheduling*, Management Science, 34 (1988), pp. 391–401.

[2] M. AROSTEGUI, JR., S. KADIPASAOGLU, AND B. KHUMAWALA, *An empirical comparison of tabu search, simulated annealing, and genetic algorithms for facilities location problems*, International Journal of Production Economics, 103 (2006), pp. 742–754.

[3] A. BAYKASOGLU AND L. ÖZBAKIR, *Analyzing the effect of dispatching rules in the scheduling performance through grammar based flexible scheduling system*, International Journal of Production Economics, 124 (2010), pp. 269–381.

[4] J. C. BECK, T. K. FENG, AND J.-P. WATSON, *Combining constraint programming and local search for job-shop scheduling*, INFORMS Journal on Computing, 23 (2011), pp. 1–14.

[5] J. H. BLACKSTONE, JR., D. T. PHILLIPS, AND G. L. HOGG, *A state-of-the-art survey of dispatching rules for manufacturing job shop operations*, International Journal of Production Research, 20 (1982), pp. 27–45.

[6] J. BŁAŻEWICZ, K. H. ECKER, E. PESCH, G. SCHMIDT, AND J. WĘGLARZ, *Constraint programming and disjunctive scheduling*, in Handbook on Scheduling, Springer Berlin Heidelberg, 2007, pp. 477–538.

[7] E. H. BOWMAN, *The schedule-sequencing problem*, Operations Research, 7 (1959), pp. 621–624.

[8] P. BRUCKER, *Scheduling Algorithms*, Springer-Verlag, Berlin Heidelberg, 5th ed., 2007.

[9] P. BRUCKER, B. JURISCH, AND A. KRÄMER, *Complexity of scheduling problems with multi-purpose machines*, Annals of Operations Research, 70 (1997), pp. 57–73.

[10] K. BÜLBÜL AND P. KAMINSKY, *A linear programming-based method for job shop scheduling*. Available at `http://www.optimization-online.org/DB_HTML/2010/12/2867.html`, 2010.

[11] CSCMP SUPPLY CHAIN MANAGEMENT PROFESSIONALS, *CSCMP supply chain management definitions*. Available at `http://http://cscmp.org/aboutcscmp/definitions.asp`.

[12] L. DE GIOVANNI AND F. PEZZELLA, *An improved genetic algorithm for the distributed and flexible job-shop scheduling problem*, European Journal of Operational Research, 200 (2010), pp. 395–408.

[13] K. FANG, N. A. UHAN, F. ZHAO, AND J. W. SUTHERLAND, *Flow shop scheduling for sustainable manufacturing*, in Proceedings of 10th Workshop on Models and Algorithms for Planning and Scheduling Problems, Nymburk, Czech Republic, 2011.

[14] P. FATTAHI, M. SAIDI MEHRABAD, AND F. JOLAI, *Mathematical modeling and heuristic approaches to flexible job shop scheduling problems*, Journal of Intelligent Manufacturing, 18 (2007), pp. 331–342.

[15] R. FOURER, D. M. GAY, AND B. W. KERNIGHAN, *AMPL: A Modeling Language for Mathematical Programming*, Brooks/Cole Publishing Company/Cengage Learning, 2nd ed., 2002.

[16] J. GAO, L. SUN, AND M. GEN, *A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems*, Computers & Operations Research, 35 (2008), pp. 2892–2907.

[17] M. GAREY AND D. JOHNSON, *Computers and intractability*, W. H. Freeman & Co., New York, NY, USA, 1979.

[18] M. R. GAREY, D. S. JOHNSON, AND R. SETHI, *The complexity of flowshop and jobshop scheduling*, Mathematics of Operations Research, 1 (1976), pp. 117–129.

[19] M. GOMES, A. BARBOSA-PÓVOA, AND A.Q.NOVAIS, *Optimal scheduling for flexible job shop operation*, International Journal of Production Research, 43 (2005), pp. 2323–2353.

[20] R. L. GRAHAM, E. L. LAWLER, J. K. LENSTRA, AND A. H. G. RINNOOY KAN, *Optimization and approximation in deterministic sequencing and scheduling: A survey*, Annals of Discrete Mathematics, 5 (1979), pp. 287–326.

[21] J. HEINONEN AND F. PETTERSSON, *Hybrid ant colony optimization and visibility studies applied to a job-shop scheduling problem*, Applied Mathematics and Computation, 187 (2007), pp. 989–998.

[22] J. W. HERRMANN, *Improving production scheduling: Integrating organizational, decision-making, and problem-solving perspectives*, in Proceedings of the 2006 Industrial Engineering Research Conference May 20-24, Orlando, Florida, USA, 2006.

[23] A. B. HMIDA, M. HAOUARI, M.-J. HUGUET, AND P. LOPEZ, *Discrepancy search for the flexible job shop scheduling problem*, Computers & Operations Research, 37 (2010), pp. 2192–2201.

[24] O. HOLTHAUS AND C. RAJENDRAN, *Efficient dispatching rules for scheduling in a job shop*, International Journal of Production Economics, 48 (1997), pp. 87–105.

[25] W. J. HOPP AND M. L. SPEARMAN, *Factory Physics*, McGraw-Hill/Irwin, New York, NY, USA, 3^{d} ed., 2008.

[26] IBM CORP., *IBM ILOG CPLEX V12.1 User's Manual for CPLEX*, 2009.

[27] A. JAIN AND S. MEERAN, *Deterministic job-shop scheduling: Past, present and future*, European Journal of Operational Research, 113 (1999), pp. 390–434.

[28] T. JANSSON, *Resource utilization in a multitask cell*, master's thesis, Department of Mathematical Sciences, Chalmers University of Technology, Göteborg, Sweden, 2006.

[29] S. KEDAD-SIDHOUM, Y. R. SOLIS, AND F. SOURD, *Lower bounds for the earliness-tardiness scheduling problem on single and parallel machines*, European Journal of Operational Research, 189 (2008), pp. 1305–1316.

[30] A. KONONOV, *Combinatorial complexity of scheduling jobs with simple linear deterioration*, Discrete Analysis and Operations Research, 3 (1996), pp. 15–32.

[31] S. KREIPL AND J. DICKERSBACH, *Scheduling coordination problems in supply chain planning*, Annals of Operations Research, 161 (2008), pp. 103–122.

[32] L.LIBERTI AND J. OSTROWSKI, *On the generation of symmetry breaking constraints for mathematical programs*, tech. report, Argonne National Laboratory, 9700 S. Cass Avenue Argonne, IL 60439, France, July 2011.

[33] A. S. MANNE, *On the job-shop scheduling problem*, Operations Research, 8 (1960), pp. 219–223.

[34] F. MARGOT, *Symmetry in integer linear programming*, in 50 Years of Integer Programming 1958-2008, M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, eds., Springer Berlin, 2010, pp. 647–686.

[35] D. MATTFELD AND C. BIERWIRTH, *An efficient genetic algorithm for job shop scheduling with tardiness objectives*, European Journal of Operational Research, 155 (2004), pp. 616–630.

[36] S. MEERAN AND M. MORSHED, *A hybrid genetic tabu search algorithm for solving job shop scheduling problems: a case study*, Journal of Intelligent Manufacturing, (2011). Available at `http://dx.doi.org/10.1007/s10845-011-0520-x`.

[37] L. MÖNCH AND R. DRIESSEL, *A distributed shifting bottleneck heuristic for complex job shops*, Computers & Industrial Engineering, 49 (2005), pp. 363–380.

[38] G. MOSHEIOV, *Complexity analysis of job-shop scheduling with deteriorating jobs*, Discrete Applied Mathematics, 117 (2002), pp. 195–209.

[39] G. L. NEMHAUSER AND L. A. WOLSEY, *Integer and Combinatorial Optimization*, John Wiley & Sons, Inc., New York, NY, USA, 1988.

[40] C. ÖZGÜVEN, L. ÖZBAKIR, AND Y. YAVUZ, *Mathematical models for job-shop scheduling problems with routing and process plan flexibility*, Applied Mathematical Modelling, 34 (2010), pp. 1539–1548.

[41] S. PETTERSSON, *The interaction between optimization model and production at Volvo Aero*, master's thesis, Department of Technology Management and Economics, Chalmers University of Technology, Göteborg, Sweden, 2010.

[42] M. E. POSNER, *Job shop scheduling*, in Wiley Encyclopedia of Operations Research and Management Science, 2011.

[43] D. RONCONI AND E. BIRGIN, *Mixed-integer programming models for flowshop scheduling problems minimizing the total earliness and tardiness*, in Just-in-Time Systems, Springer Series on Optimization and Its Applications, vol. 60, 2011. Available at `www.ime.usp.br/~egbirgin/publications/rob.pdf`.

[44] J. ROSLÖF, I. HARJUNKOSKI, T. WESTERLUND, AND J. ISAKSSON, *Solving a large-scale industrial scheduling problem using MILP combined with a heuristic procedure*, European Journal of Operational Research, 138 (2002), pp. 29–42.

[45] J. SOUSA AND L. WOLSEY, *A time indexed formulation of non-preemptive single machine scheduling problems*, Mathematical Programming, 54 (1992), pp. 353–367.

[46] E. STAFFORD, JR., F. T. TSENG, AND J. N. D. GUPTA, *Comparative evaluation of MILP flowshop models*, The Journal of the Operational Research Society, 56 (2005), pp. 88–101.

[47] P. STOOP AND V. WIERS, *The complexity of scheduling in practice*, International Journal of Operations & Production Management, 16 (1996), pp. 37–53.

[48] A. SYBERFELDT, *A multi-objective evolutionary approach to simulation-based optimisation of real-world problems*, PhD thesis, DeMontfort University, Leicester, UK, 2009.

[49] A. SYBERFELDT, A. NG, R. I. JOHN, AND P. MOORE, *Multi-objective evolutionary simulation-optimisation of a real-world manufacturing problem*, Robotics and Computer-Integrated Manufacturing, 25 (2009), pp. 926–931.

[50] H. A. TAHA, *Operations research: An introduction*, Pearson Prentice Hall, Upper Saddle River, NJ, USA, 8th ed., 2007.

[51] K. THÖRNBLAD, T. ALMGREN, A.-B. STRÖMBERG, AND M. PATRIKSSON, *Optimering av scheman för en verklig produktionscell: tidsdiskretisering reducerar lösningstiden utan att lösningarnas kvalitet försämras*, in PLANs forsknings- och tillämpningskonferens, Norrköping, Sweden, 2011.

[52] K. THÖRNBLAD AND L. KJELLSDOTTER IVERT, *A comparison of schedules resulting from priority rules and mathematical optimization for a real production cell*, in PLANs forsknings- och tillämpningskonferens, Skövde, Sweden, 2010.

[53] K. THÖRNBLAD AND M. PATRIKSSON, *A note on the complexity of flow-shop scheduling with deteriorating jobs*, Discrete Applied Mathematics, 159 (2011), pp. 251–253.

[54] K. THÖRNBLAD, M. PATRIKSSON, A.-B. STRÖMBERG, AND T. ALMGREN, *Mathematical modelling of a real flexible job shop in aero engine component manufacturing*, in Proceedings of 10th Workshop on Models and Algorithms for Planning and Scheduling Problems, Nymburk, Czech Republic, 2011.

[55] K. THÖRNBLAD, A.-B. STRÖMBERG, T. ALMGREN, AND M. PATRIKSSON, *Optimization of schedules for a multitask production cell*, in 22nd Nofoma conference proceedings, Kolding, Denmark, 2010.

[56] J. VAN DEN AKKER, C. HURKENS, AND M. SAVELSBERG, *Time-indexed formulations for machine scheduling problems: Column generation*, INFORMS Journal on Computing, 12 (2000), pp. 111–124.

[57] T. VOLLMANN, W. L. BERRY, AND D. WHYBANK, *Manufacturing Planning and Control Systems*, Irwin, Homewood IL, USA, 3d ed.

[58] H. M. WAGNER, *An integer linear-programming model for machine scheduling*, Naval Research Logistics Quarterly, 6 (1959), pp. 131–140.

[59] S. WANG AND J. YU, *An effective heuristic for flexible job-shop scheduling problem with maintenance activities*, Computers & Industrial Engineering, 59 (2010), pp. 436–447.

[60] L. A. WOLSEY, *MIP modelling of changeovers in production planning and scheduling problems*, European Journal of Operational Research, 99 (1997), pp. 154–165.

[61] L. A. WOLSEY, *Integer Programming*, John Wiley & Sons, Inc., New York, NY, USA, 1998.

[62] R. ZHANG AND W. CHENG, *A hybrid approach to large-scale job shop scheduling*, Applied Intelligence, 32 (2010), pp. 47–59.

[63] Z. ZHU AND R. HEADY, *Minimizing the sum of earliness/tardiness in multi-machine scheduling: a mixed integer programming approach*, Computers & Industrial Engineering, 38 (2000), pp. 297–305.