GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Algorithmic Approaches to Flexible Job Shop Scheduling

Master Thesis

Fakultät für Mathematik und Informatik
Institut für Numerische und Angewandte Mathematik
Studiengang Wirtschaftsmathematik

Submitted by / on:	Morten Tiedemann / 14 March 2012
Matriculation Number:	20723108
Adress:	Kreuzbergring 56c
	37075 Göttingen
Email:	morten.tiedemann@stud.uni-goettingen.de

Primary Reviewer:	Prof. Dr. Stephan Westphal
Secondary Reviewer:	Prof. Dr. Anita Schöbel

Contents

Chapter 1

Introduction

Scheduling is the allocation of shared resources over time to competing activities. One of the applications that motivated research in this area is machine scheduling: Jobs describe activities and machines processing at most one operation at a time represent resources. Motivated by a machine scheduling problem provided by a manufacturer of vacuum chambers, this master thesis is concerned with a special case of machine scheduling, namely the flexible job shop scheduling problem. Order-specific production requires accurate scheduling in order to ensure efficient manufacturing. Furthermore, the manufacturing of a vacuum chamber consists of a sequence of individual operations, each occupying shared machines or resources, and additionally, each operation can be allocated to a subset of valid machines. This environment is appropriately modeled by the flexible job shop scheduling problem.

In the course of this thesis, the modeling and optimal solving of the flexible job shop scheduling as well as approximation algorithms for the flexible job shop scheduling problem are covered. Due to the complexity of the problem, most of the literature concerned with flexible job shop scheduling focuses on heuristic approaches such as genetic algorithms, tabu search or other local search algorithms. In contrast to this, the modeling of the flexible job shop scheduling problem plays an essential role in this thesis. Furthermore, this thesis brings LP-based heuristics grounded on the developed models into focus.

In Chapter 2, the flexible job shop scheduling problem is formally defined, its complexity is discussed, and an outline of the existing literature concerned with the flexible job shop scheduling problem is given. Thereupon, Chap-

ter 3 is devoted to the modeling of the flexible job shop scheduling problem aiming for an optimal solution. Different modeling approaches are developed and evaluated. In order to improve the performance of the models, several structural improvements are presented and compared to the basic models. In Chapter 4, approximation algorithms for the flexible job shop scheduling problem are discussed. First of all, a performance guarantee is given and subsequently, LP-based heuristics are examined. Here, two different approaches are being pursued. First, an LP-based heuristic utilizing the convex hull of feasible solutions is presented and a performance ratio is derived. Second, the mixed integer programming formulations given in Chapter 3 are modified in order to serve as a basis for another LP-based heuristic. In the latter case, empirical performance evaluations are constituted and additionally, a practical application of the LP-based heuristic to the actual scheduling problem provided by the manufacturer mentioned above is presented in Chapter 5.

Furthermore, for the reader's convenience, Appendix A lists frequently used notation.

Acknowledgments

I would like to express my gratitude to Prof. Dr. Stephan Westphal for his intensive supervision of this master thesis, an open door policy, and the resulting inspiring discussions. Furthermore, I appreciate the cooperation with Trinos Vakuum Systeme GmbH and especially, I would like to thank Timm Marienhagen and Ole Marienhagen.

Last but not least, I am indebted to Marco Bender for a helpful revision of this thesis, and Sören Kruse, Frederik Tietz, and Janneke van Hoorn for valuable suggestions and always having a sympathetic ear for my problems.

Chapter 2

Preliminaries

This chapter is devoted to the introduction and formal definition of the flexible job shop scheduling problem. Additionally, its complexity is discussed and existing literature concerned with the problem is reviewed. Initially, some basic concepts of combinatorial optimization are refurbished in order to familiarize the reader with the mathematical notions used throughout this thesis.

2.1 Combinatorial Optimization

The modeling techniques for scheduling problems involve the formulation of *mixed integer linear programming problems*. A mixed integer linear programming problem in standard form is given by vectors $c \in \mathbb{Q}^n$, $h \in \mathbb{Q}^p$, and $b \in \mathbb{Q}^m$, and matrices $A \in \mathbb{Q}^{m \times n}$ and $G \in \mathbb{Q}^{m \times p}$, where $\mathbb{Q}^n$ is the set of rational n-dimensional vectors. The data sets are assumed to be rational due to the inability of digital computers to deal with real numbers. The objective is to find vectors $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_p)$ being the optimal solution to the problem

$$\begin{aligned} \min \quad & c^T x + h^T y && \text{(MIP)} \\ \text{s.t.} \quad & Ax + Gy \leq b \\ & x \in \mathbb{Z}_+^n \\ & y \in \mathbb{R}_+^p, \end{aligned}$$

where $\mathbb{Z}_+^n$ is the set of nonnegative integral n-dimensional vectors and $\mathbb{R}_+^p$ is the set of nonnegative real-valued p-dimensional vectors. The function $c^T x + h^T y$ is called the *objective function* and the set

$$S := \{x \in \mathbb{Z}_+^n,\, y \in \mathbb{R}_+^p \mid Ax + Gy \leq b\}$$

is called the *feasible region*. Mixed integer linear programming in general is $\mathcal{NP}$-hard [NW88, p. 133]. For a detailed introduction of integer programming it is referred to [NW88].

Various solution approaches for integer programs exist and usually several methods are combined in mathematical optimization software. For a better understanding of the solution process of (mixed) integer programs the basic techniques are reviewed in the following. First of all, many techniques are based on *linear relaxations* of the (mixed) integer programs. A linear relaxation of an integer program $\min\{c^t x \mid Ax \leq b,\, x \in \mathbb{Z}_+^n\}$ is given by the relaxation of the integrality constraints, that is $\min\{c^t x \mid Ax \leq b,\, x \in \mathbb{R}_+^n\}$. Furthermore, one of the most important techniques deployed in the solution process of (mixed) integer programs is the *branch and cut procedure* combining a *branch and bound search* and *cutting plane algorithms*.

Branch and bound search is an implicit enumeration technique for solving (mixed) integer programs. The main problem is processed with the aid of suitably constructed subproblems (branches) in the form of a search tree, whereat each branch is assessed with respect to the objective function and applied for the generation of bounds for the objective value.

Cutting plane algorithms add linear inequalities to the relaxed linear program of a (mixed) integer program separating the non-integer solution of the linear relaxation from the convex hull of the original feasible set. Such inequalities are called cuts. Then, the current non-integer solution is no longer feasible to the relaxation. This process is repeated until an optimal integer solution is found.

Furthermore, the notion of conjunctive and disjunctive constraints is introduced. Mixed integer programs only permit *conjunctive constraints*, that is in a feasible solution each constraint of the system of inequalities is satisfied. As opposed to this, the choice between two alternatives is called a

disjunction. The generalization of mixed integer programming to permitting disjunctive constraints is called *disjunctive mixed integer programming*. Disjunctive constraints naturally arise in scheduling problems. Suppose that two jobs must be processed on the same machine and cannot be processed simultaneously. Let p_1 and p_2 be the processing times of the two jobs and the variables t_1 and t_2 the corresponding starting times. This leads to the disjunctive constraint

$$t_1 + p_1 \leq t_2 \quad \vee \quad t_2 + p_2 \leq t_1.$$

Assuming $0 \leq t_i \leq T$ for $i = 1, 2$, the disjunctive constraints can be reformulated with the help of binary variables y_i for $i = 1, 2$. Choose $M \geq \max\{t_1 + p_1 - t_2, t_2 + p_2 - t_1 \,|\, 0 \leq t_1, t_2 \leq T\}$ and take as constraints

$$\begin{aligned} t_1 + p_1 &\leq t_2 + M(1 - y_1) \\ t_2 + p_2 &\leq t_1 + M(1 - y_2) \\ y_1 + y_2 &= 1 \\ t_1, t_2 &\leq T \\ t_1, t_2 &\geq 0 \\ y_1, y_2 &\in \{0, 1\}. \end{aligned}$$

This reformulation can be applied to general disjunctive constraints, as long as the variables are bounded. However, these constraints significantly corrupt the sharpness of the linear relaxations if large constants M, so-called *Big-M constants*, are chosen. Since mathematical optimization software uses linear relaxations in the solution process of mixed integer programs, as seen in the description of the branch and bound search, it is quite important to choose Big-M constants as tight as possible.

2.2 The Flexible Job Shop Scheduling Problem

In this section the flexible job shop scheduling problem is now introduced formally. Before doing so, a more commonly known scheduling problem, the job shop scheduling problem, is stated and subsequently, the flexible job shop scheduling problem is presented as a generalization.

The classical *job shop scheduling problem* can be stated as follows [BK06]:

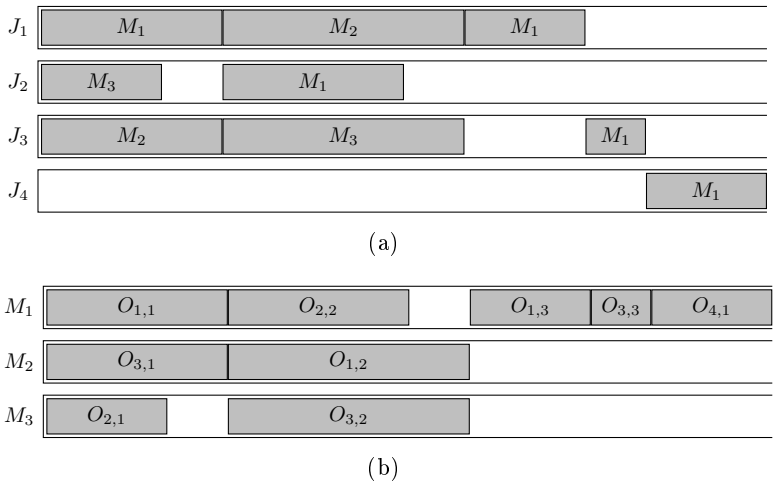


Figure 2.1: Job- and machine-oriented Gantt charts

We are given a shop environment with a set $M = \{\mu_1, \ldots, \mu_m\}$ of m machines and have to process n jobs $J_1, \ldots, J_n$. Job J_i consists of n_i operations $O_{i,j}$, $j = 1, \ldots, n_i$, which have to be processed subsequently,

$$O_{i,1} \to O_{i,2} \to \cdots \to O_{i,n_i}.$$

Operation $O_{i,j}$ must be processed for $p_{i,j} > 0$ time units on a dedicated machine $m_{i,j} \in \{\mu_1, \ldots, \mu_m\}$ without preemption, i.e., it cannot be interrupted during its execution. Furthermore, each machine can process at most one job at a time.

A *schedule* $S = (S_{i,j})$ is defined by the starting times of all operations. A feasible schedule S has to respect the following constraints:

$$S_{i,j} + p_{i,j} \leq S_{i,j+1} \quad \text{for all jobs } i = 1, \ldots, n, \text{ operations } j = 1, \ldots, n_i - 1 \tag{2.1}$$

$$S_{i,j} + p_{i,j} \leq S_{k,l} \vee S_{k,l} + p_{k,l} \leq S_{i,j} \quad \text{for all pairs } O_{i,j}, O_{k,l} \text{ of operations with } m_{i,j} = m_{k,l} \tag{2.2}$$

Constraint (2.1) ensures that the order of operations of each job is maintained and constraint (2.2) assures that each machine processes at most one job at a time. Schedules may be visualized by means of *Gantt charts* as shown in Figure 2.1. In a job-oriented Gantt chart, see Figure 2.1(a), each job is represented by one row and the operations belong to that job are posi-

Objective Function	Description
$C_{\max} := \max_{i=1,\dots,n} C_i$	makespan
$\sum_{i=1}^{n} C_i$	total flow time
$\sum_{i=1}^{n} w_i C_i$	weighted (total) flow time

Table 2.1: Objective functions for the job shop scheduling problem

tioned according to their starting time and labeled by the assigned machine. In a machine-oriented Gantt chart, see Figure 2.1(b), each machine is represented by one row and each operation is positioned according to its starting time in the row corresponding its assigned machine.

The objective of the job shop scheduling problem is to determine a feasible schedule S minimizing a given objective function. A wide class of objective functions can be elaborated based on the completion times of the jobs. Let $C_i := S_{i,n_i} + p_{i,n_i}$ be the completion time of job J_i and denote by d_i the due date of job J_i. In Table 2.1 some common objective functions of practical relevance are specified [Bru04]. Further objective functions can be defined using other special functions such as

$$
\begin{aligned}
E_i &:= \max\{0, d_i - C_i\} && \text{(earliness)},\\
T_i &:= \max\{0, C_i - d_i\} && \text{(tardiness)},\\
D_i &:= |C_i - d_i| && \text{(absolute deviation)},\\
S_i &:= (C_i - d_i)^2 && \text{(squared deviation)},\\
U_i &:= \begin{cases} 0 & \text{if } C_i \le d_i \\ 1 & \text{otherwise} \end{cases} && \text{(unit penalty)}.
\end{aligned}
$$

The selection of the objective function is obviously dependent on the practical application of the scheduling problem as outlined by the following examples:

Just-In-Time Just-in-time production refers to a production strategy aiming for continuous material flows along the supply chain. One of the crucial factors is to complete the jobs according to the given deadlines,

neither sooner nor later. Consequently, an objective function based on the absolute deviation D_i or the squared deviation S_i is advisable.

High Penalty Costs If a company is obligated to pay high penalties in case of an exceeded deadline, it is appropriate to choose an objective function based on the lateness L_i.

Hospital Another typical example for a job shop is a hospital. The patients are modeled as jobs, that have to run through different places of the hospital such as front desk, doctor's room, X-ray room, operation room, etc. In order to minimize the overall duration of the treatments and to distinguish patients in their significance, the weighted (total) flow time $\sum w_i C_i$ is an adequate choice for the objective function.

Still, since the most predominant objective for scheduling is the minimization of the makespan, our considerations are restricted to the makespan $C_{\max}$ until further notice.

The *flexible job shop scheduling problem* is a generalization of the job shop scheduling problem [Bra93]. In the flexible job shop scheduling problem, an operation $O_{i,j}$ can be processed by a set of machines $M_{i,j} \subseteq \{\mu_1, \ldots, \mu_m\}$ in contrast to the job shop scheduling problem, where each operation is dedicated to a single machine. The processing time of operation $O_{i,j}$ on machine $k \in M_{i,j}$ is denoted by $p_{i,j,k}$. In order to simplify the notations for the flexible job shop scheduling problem it is convenient to identify the operations $O_{i,j}$ by numbers $1, \ldots, N$ where $N := \sum_{i=1}^{n} n_i$. Consequently, the set of machines operation $i \in \{1, \ldots, N\}$ can be assigned to, is now denoted by $M_i \subseteq \{\mu_1, \ldots, \mu_m\}$ and the processing time of operation i on machine $k \in M_i$ is denoted by $p_{i,k}$.

In order to be able to classify the (flexible) job shop scheduling problem into the wide field of scheduling it is made use of the classification scheme introduced by Graham et al. [GLLR79]. Accordingly, scheduling problems are classified in terms of a three-field classification $\alpha|\beta|\gamma$ where α specifies the machine environment, β denotes the job characteristics and γ specifies the optimality criterion. Examples 2.1 and 2.2 give an illustration of the three-field notation for two different scheduling problems.

Example 2.1. *$R \,|\, chains,\ p_i = 1 \,|\, C_{\max}$ is the problem of scheduling jobs with chain precedence constraints (chains) and uniform processing times ($p_i = 1$)*

on unrelated parallel machines (R) such that the makespan ($C_{\max}$) is minimized.

Example 2.2. *$J2\,|\,pmtm\,|\,\sum w_iC_i$ is the problem of scheduling jobs preemptively (pmtm) in a two-machine job shop (J2) such that the weighted total flow time ($\sum w_iC_i$) is minimized.*

Furthermore, in Table 2.2 important characteristics of scheduling problems considered in the course of this thesis are specified. The abbreviation FJ is introduced for the machine environment of the flexible job shop scheduling problem as described above, the remaining notation is adopted from [Bru04]. If the processing times for each operation on the valid machines do not differ from each other, i.e., $p_{i,k} = p_{i,l}$ for all machines $k, l \in M_i$, the problem is denoted as job shop scheduling problem with *multi-purpose machines* [Bru04] and labeled as $JMPM$. The job shop scheduling problem with multi-purpose machines is a special case of the flexible job shop scheduling problem. Finally, two equivalences are pointed out. First, $JMPM\,||\,C_{\max}$ is, according to the notation used throughout this thesis, equivalent to $FJ\,|\,p_{i,k} = p_i\,|\,C_{\max}$. Secondly, $FJ\,||\,C_{\max}$ is equivalent to the problem of scheduling jobs with chain precedence constraints on unrelated parallel machines, which is denoted by $R\,|\,\text{chains}\,|\,C_{\max}$. For the latter case, each instance of $R\,|\,\text{chains}\,|\,C_{\max}$ corresponds to an instance of $FJ\,||C_{\max}$, whereby the complete set of machines is valid for each operation. Equally, each instance of $FJ\,||\,C_{\max}$ corresponds to an instance of $R\,|\,\text{chains}\,|\,C_{\max}$, whereby the processing time of operations on invalid machines is set to infinity.

2.3 Complexity

In this section the computational complexity of the flexible job shop scheduling problem is discussed. The complexity theory provides a mathematical framework for the classification of computational problems with respect to their complexity. In the following, some basic definitions of computational complexity used in this section are given. For a more detailed introduction it is referred to [GJ79].

The complexity theory is designed to be applied to *decision problems*. A problem is called a decision problem if the output range is $\{\text{yes}, \text{no}\}$. Each

Field	Option	Description
α	J	job shop machine environment
α	FJ	flexible job shop machine environment
α	$JMPM$	job shop machine environment with multi-purpose machines
β	d_i	job deadlines are specified
β	r_i	job release dates are specified
β	$p_{i,k} = p_i$	$p_{i,k} = p_{i,l}$ for all machines $k, l \in M_i$
β	$n_i = k$	maximum number of operations per job
γ	$C_{\max}$	minimize the makespan
γ	$\sum_{i=1}^{n} w_i C_i$	minimize the total weighted completion time

Table 2.2: Classification characteristics of the (flexible) job shop scheduling problem used in this thesis

optimization problem may be associated with a decision problem by defining a threshold k for the corresponding objective function f. For a minimization problem the decision problem is then given by: Does there exist a feasible solution S such that $f(S) \leq k$?

Complexity theory commonly distinguishes between two classes of decision problems. First, the class $\mathcal{P}$ contains all decision problems for which a polynomial-time deterministic algorithm exists. Secondly, the class $\mathcal{NP}$ is defined to be the class of all decision problems that can be solved by polynomial-time nondeterministic algorithms. Obviously, $\mathcal{P} \subseteq \mathcal{NP}$. Furthermore, it is generally conjectured that $\mathcal{P} \neq \mathcal{NP}$, but the "$\mathcal{P}$ versus $\mathcal{NP}$" problem is one of the major open problems of modern mathematics.

A decision problem Π is called *polynomial-time reducible* to a decision problem Π' if the following two conditions hold:

1. There exists a polynomial-time computable function f transforming inputs for Π to inputs for Π'.

2. For all inputs I, the output of Π for instance I is yes if and only if the output of Π' for instance $f(I)$ is yes.

If Π is polynomial-time reducible to Π', it is denoted by $\Pi \propto \Pi'$. A decision problem Π is $\mathcal{NP}$-*hard* if $\Pi' \propto \Pi$ for all other decision problems $\Pi' \in \mathcal{NP}$. If additionally $\Pi \in \mathcal{NP}$ holds, then Π is called $\mathcal{NP}$-*complete*.

By means of these basic definitions, the complexity of the flexible job shop scheduling problem is now surveyed. Since the flexible job shop scheduling problem is comprised of an assignment problem, that is each operation has to be assigned to a machine from its set of valid machines, and a classical job shop scheduling problem, the flexible job shop scheduling problem is more complex than the job shop scheduling problem. The complexity of the job shop scheduling problem has been studied intensively. In [SS95] Sotskov et al. proved that the job shop scheduling problem with three jobs and three machines $J3\,|\,n = 3\,|\,C_{\max}$ is $\mathcal{NP}$-hard. In [GJS76] Garey et al. proved that the job shop scheduling problem with two jobs $J2\,|\,|\,C_{\max}$ is $\mathcal{NP}$-hard. Since the job shop scheduling problem is a special case of the flexible job shop scheduling problem, these results hold for the flexible job shop scheduling problem.

The complexity of the flexible job shop scheduling problem is further characterized by the following results. In [Ake56] Akers Jr. presented a reduction of the job shop scheduling problem with two jobs $J\,|\,n = 2\,|\,C_{\max}$ to a restricted shortest path problem in the two-dimensional plane. This reduction is briefly outlined in the following. In Figure 2.2 a restricted shortest path problem corresponding to a job shop scheduling problem with two jobs and $n_1 = 3$ and $n_2 = 2$ is depicted. The processing times of the operations of job J_1 and job J_2 are presented as intervals on the horizontal axis and vertical axis respectively. The order of intervals matches the order of operations in job J_1 and job J_2. Additionally, the intervals are labeled by the machine the corresponding operation is assigned to. The region $I_1 \times I_2$, where I_1 is an interval on the horizontal axis and I_2 is an interval on the vertical axis, is marked as an obstacle if the intervals correspond to the same machine. A feasible schedule for the job shop scheduling problem equals a path from O to F with the following properties:

1. The path consists of segments which are either parallel to one of the axes or at a 45-degree angle.

2. The path avoids the interior of any rectangular obstacle.

During an axis-parallel segment of the path an operation of only one job is

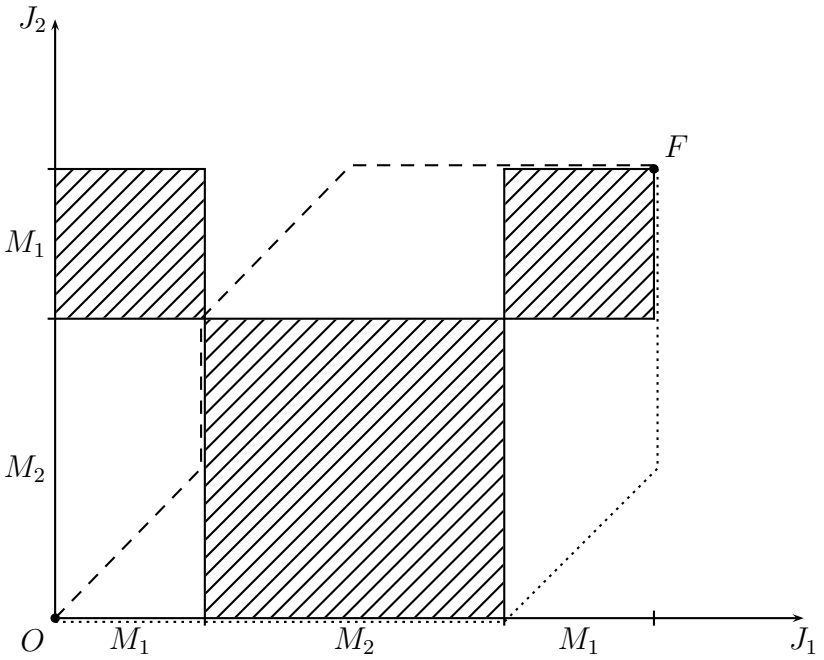


Figure 2.2: Graphical representation of two feasible solutions for a job shop scheduling problem with two jobs

processed and during a 45-degree segment of the path operations of both jobs are processed in parallel. The avoidance of obstacles corresponds to the fact that at most one operation at a time can be processed on each machine. In order to determine the length of the path in conformity with the length of the associated schedule, the projections on an axis of the 45-degree segments are considered. Thus, the length of the path is given by

$$\sum \text{horizontal segments} + \sum \text{vertical segments} + \frac{1}{\sqrt{2}} \sum \text{45-degree segments}.$$

Furthermore, Brucker et al. gave a reduction of the restricted shortest path problem in the two-dimensional plane to an unrestricted shortest path problem in the network $N = (V, A, d)$ [Bru04]. An optimal schedule for the job shop scheduling problem corresponds with a shortest O-F-path in N. Additionally, it is proved that the construction of the network and the calculation of the shortest path has complexity $\mathcal{O}(N \log N)$. Consequently, the job shop scheduling problem with two jobs can be solved polynomially. In [BS90] Brucker et al. presented a generalization of this approach yielding a polynomial-time algorithm for the problem $FJ \,|\, p_{i,k} = p_i \,|\, C_{\max}$. More

precisely, it is shown that for $FJ\,|\,n = 2, p_{i,k} = p_i\,|\,C_{\max}$ a schedule with minimum makespan can be found in $\mathcal{O}\left(\max\{n_1, n_2\}^3\right)$ time.

Secondly, in [BJK97] Brucker et al. proved that the job shop scheduling problem with multi-purpose machines, three jobs and two machines $JMPM2\,|\,n = 3\,|\,C_{\max}$ is $\mathcal{NP}$-hard. Since this problem is a special case of the flexible job shop scheduling problem, $JMPM2\,|\,n = 3\,|\,C_{\max}$ is $\mathcal{NP}$-hard, too. The proof presented in [BJK97] is based upon a reduction of the problem PARTITION which is known to be $\mathcal{NP}$-complete [GJ79, p. 223], to an instance of the job shop scheduling problem with multi-purpose machines.

2.4 Literature Review

This section provides a review of the literature relevant for the flexible job shop scheduling problem. Literature concerning the complexity of the flexible job shop scheduling problem is already covered in Section 2.3, consequently it is not mentioned here.

Due to the complexity of the flexible job shop scheduling problem, cf. Section 2.3, mixed integer programming formulations are only sparsely covered in the literature. In [EAS+11] Elazeem et al. introduced some optimality conditions for the solution of the flexible job shop scheduling problem and a mathematical model is presented. In [CC02] Choi et al. presented a mixed integer programming formulation for the flexible job shop scheduling problem. In [SMF06] Saidi-Mehradbad et al. and in [FSMJ07] Fattahi et al. extended this formulation and specified results for different instances obtained with a branch and bound method. Still, most of the literature related to the flexible job shop scheduling problem is devoted to heuristics searching for a "good" solution of the problem, instead of solving it to optimality. Here, a wide range of metaheuristics for combinatorial problems is applied to the flexible job shop scheduling problem. In order to give a well-arranged outline, three different metaheuristics are briefly illustrated and some of the literature applying this kind of metaheuristic is cited.

Genetic algorithms are search heuristics introduced by John Holland in the early 1970's that mimic the process of natural evolution [Hol75]. There are two mechanisms that link a genetic algorithm to the problem it is solving. First, evolution takes place on chromosomes, which are represented by solutions of a combinatorial problem. One of the

determining characteristics of a genetic algorithm is the way of encoding feasible solutions to the problem on chromosomes. Secondly, in order to mimic the process of natural selection, an evaluation function is essential, returning a measurement of the worth of any chromosome in the context of the problem. In the following a description of the execution of a genetic algorithm based on [Dav91, p. 5] is given.

1. Initialize a population of chromosomes, i.e., an initial set of feasible solutions to the problem.
2. Evaluate each chromosome in the population.
3. Create new chromosomes by mating current chromosomes. Apply mutation and recombination as the parent chromosomes mate.
4. Delete members of the population to make room for the new chromosomes.
5. Evaluate the new chromosomes and insert them into the population.

A genetic algorithm is applied to the flexible job shop scheduling problem for instance by Chen et al. in [CIL99]. Here, a feasible solution to the problem is represented by an individual consisting of two chromosomes, chromosome A and chromosome B. The first one defines the routing policy, the latter one defines the sequence of operations on each machine. Furthermore, the evaluation function is to compute the makespan for each solution represented by chromosomes. The value of the evaluation function is used to determine the survival probability of this individual compared to the others.

Different variants of genetic algorithms are for example applied by Pezzella et al. in [PMC08] and Zhang et al. in [ZGS11]. Numerical experiments published in the literature cited above substantiate the effectiveness and efficiency of genetic algorithms for solving the flexible job shop scheduling problem approximately.

Tabu search is a local search technique used for combinatorial optimization, initially formalized by Glover in [Glo89]. Consider a problem of the form

$$\min c(x) : x \in X. \tag{P}$$

Tabu search is a procedure that is characterized by a sequence of moves that lead from one trial solution (selected $x \in X$) to another. A move s consists of a mapping defined on a subset $X(s) \subseteq X$. Let S be the set of all moves. Associated with $x \in X$ is the set $S(x) := \{s \in S : x \in X(s)\}$ of those moves $s \in S$ that can be applied to x. The set $S(x)$ can be viewed as a neighborhood function. Tabu search is characterized by two key elements. First, the search is constrained by classifying certain of its moves as forbidden, i.e., tabu, and secondly, the search is freed by a short term memory function that provides strategic forgetting.

Tabu search is applied to the flexible job shop scheduling for example by Saidi-Mehrabad et al. in [SMF06], by Brandimarte in [Bra93], and by Mastrolilli et al. in [MG00]. Again, computational studies prove that tabu search is effectively applicable for the approximate solution of the flexible job shop scheduling problem.

Particle swarm optimization is in the first place attributed to Kennedy and Eberhart [KE95], and Shi [SE98]. Particle swarm optimization is based on the idea of resembling a school of flying birds. Instead of using genetic operators as described above for genetic algorithms, the individuals are evolved by cooperation and competition among the individuals themselves through generations. Each individual is named as a particle, representing a feasible solution to an optimization problem. Furthermore, each particle adjusts its flying according to its own flying experience and its companions' flying experience. According to [SE98], each particle i is characterized by a point $X_i = (x_{i,1}, \ldots, x_{i,D})$ in the D-dimensional space, the best previous position $P_i = (p_{i,1}, \ldots, p_{i,D})$ of the particle with respect to a predefined fitness function related to the optimization problem, and a velocity $V_i = (v_{i,1}, \ldots, v_{i,D})$, that is the rate of position change for particle i. The basic manipulation of particles introduced by Shi in [SE98] is given by

$$\begin{aligned} v_{i,d} &= v_{i,d} + c_1\mathcal{R}_1(p_{i,d} - x_{i,d}) + c_2\mathcal{R}_2(p_{g,d} - x_{i,d}) \\ x_{i,d} &= x_{i,d} + v_{i,d} \end{aligned} \tag{V}$$

for $d = 1, \ldots, D$, where g is the index of the best particle in the population with respect to the fitness function. Furthermore, $\mathcal{R}_1$ and $\mathcal{R}_2$

are two random functions with values in $[0, 1]$ and c_1 and c_2 are two positive constants. The second addend of equation (V) represents the egoistic thinking of each individual – flying towards the position of its own best experience. The third addend of equation (V) represents the companionable thinking of each individual – flying towards the position of the group's best experience.

Particle swarm optimization is applied to the flexible job shop scheduling problem for example by Girish et al. in [GJ09] and by Zhang et al. in [ZSLG09]. In [FYL+08] Feng et al. proposed a particle swarm optimization algorithm based on a swarm grouping mechanism for the flexible job shop scheduling problem. The algorithm partitions the swarm into many groups, and each group flies toward its own group's best particle.

The effectiveness of particle swarm optimization for solving the flexible job shop scheduling problem is again verified by computational studies presented in the literature cited above.

The metaheuristics described in the course of this section can also be combined and applied to flexible job shop scheduling as a hybrid algorithm. For example, in [GSG08] Gao et al. combined a genetic algorithm with a variable neighborhood descent, involving two local search procedures improving the individuals before the natural selection.

Furthermore, the algorithmic approaches to the flexible job shop scheduling problem can be subdivided into one level approaches and two level approaches solving the routing problem and the scheduling problem either simultaneously or consecutively. In [Bra93] Brandimarte applied a two level approach based on the decomposition of the flexible job shop scheduling problem in an assignment subproblem and a job shop scheduling subproblem. Both problems are treated by tabu search heuristics. Opposed to this two level approach, Jurisch considered the assignment problem and the scheduling problem simultaneously in [Jur92], and proposed a tabu search heuristic to solve it.

The literature review given in this section raises no claim to completeness, but it gives an outline of the variety and complexity of algorithmic approaches applied to the flexible job shop scheduling problem. As opposed to the predominance of metaheuristics in the literature of the flexible job

shop scheduling problem, in the remainder of this thesis different mixed integer programming formulations for the problem are featured and LP-based heuristics relying on the intelligence inherited from the original models to the linear relaxations are presented.

Chapter 3

Modeling Approaches

In the course of this chapter, different models for the flexible job shop scheduling problem $FJ\,|\,|\,C_{\max}$ are developed that allow for an efficient computation of optimal solutions. First of all, the problem $FJ\,|\,|\,C_{\max}$ is expressed as a mixed integer program with additional disjunctive constraints which are dependent on the machine assignment in Section 3.1. Furthermore, several reformulations of this disjunctive mixed integer program are presented in Sections 3.2 - 3.5 in order to achieve mixed integer programs without disjunctive constraints. Additionally, several performance improvements for the different models are developed in Section 3.6 and finally, computational results for the different models are provided and evaluated in Section 3.7.

Whereas heuristics for the flexible job shop scheduling problem have been studied in various forms, models that allow for a computation of optimal solutions are only sparsely covered in the literature. Since the flexible job shop scheduling problem is an $\mathcal{NP}$-hard problem, there is no efficient technique to solve it to optimality, unless $\mathcal{P} = \mathcal{NP}$. Nevertheless, it is interesting to follow different modeling approaches and compare the performance of the resulting models in interaction with state-of-the-art optimization software. By doing this, the impact of various formulations on the performance of the model is determined. Additionally, by means of the results of this chapter an efficient LP-based heuristic is developed, see Section 4.2.

3.1 Introduction

In this section, a first formulation of the problem $FJ\,||\,C_{\max}$ will be presented, as proposed by Brucker in [BK06]. Let $J(i)$ denote the job to which operation i belongs and let $P(i)$ be the position of operation i in the sequence of operations belonging to job $J(i)$ starting with one, i.e., $P(i) = 1$ if operation i is the first operation of a job. First of all, in order to model the assignment of operations to machines, assignment variables $x_{i,k} \in \{0, 1\}$ for all $k \in M_i$, $i \in O$ are introduced, where

$$x_{i,k} = \begin{cases} 1, & \text{if operation } i \text{ is assigned to machine } k \\ 0, & \text{otherwise.} \end{cases}$$

Furthermore, S_i is defined as the starting time for operation i, see Section 2.2. Thus, the makespan $C_{\max}$ is now defined by the constraints

$$C_{\max} \geq S_i + \sum_{k \in M_i} x_{i,k} p_{i,k} \quad \text{for all } i \in O \colon P(i) = n_{J(i)}. \tag{3.1}$$

In order to ensure that each operation is assigned to exactly one machine, constraints

$$\sum_{k \in M_i} x_{i,k} = 1 \quad \text{for all } i \in O \tag{3.2}$$

are introduced. Moreover, for each job the corresponding operations have to be processed in the given order, that is, the starting time of an operation must not be earlier than the point at which the preceding operation in the sequence of operations of the respective job is completed. This constraint is imposed simultaneously on all appropriate pairs of operations, aggregated in the set of conjunctions C given by

$$C := \{(i, j) \,|\, i, j \in O \colon J(i) = J(j) \,\wedge\, P(j) = P(i) + 1\}.$$

Consequently, the *precedence constraints* are given by

$$S_i + \sum_{k \in M_i} x_{i,k} p_{i,k} \;\leq\; S_j \quad \text{for all } (i, j) \in C. \tag{3.3}$$

Furthermore, for each assignment $x = (x_{i,k})$ the set $D(x)$ of all pairs of operations assigned to the same machine is defined as

$$D(x) := \{(i,j) \,|\, x_{i,k} = x_{j,k} = 1 \text{ for some } k \in M_i \cap M_j\}.$$

For each pair of operations $(i,j) \in D(x)$, either operation i has to be processed before operation j or vice versa, since a machine can process at most one operation at a time. Consequently, $D(x)$ is also called the set of disjunctions [BK06]. The disjunctive constraints arising from these requirements are provided by

$$\begin{aligned} S_i + \sum_{k \in M_i} x_{i,k} p_{i,k} \leq S_j \quad \vee \\ S_j + \sum_{k \in M_j} x_{j,k} p_{j,k} \leq S_i \text{ for all } (i,j) \in D(x). \end{aligned} \tag{3.4}$$

Altogether, the formulation for the problem $FJ\,||\,C_{\max}$ is thus given by

$$\begin{aligned}
\min \quad & C_{\max} & & \\
\text{s.t.} \quad & & & \\
& C_{\max} - \sum_{k \in M_i} x_{i,k} p_{i,k} \geq S_i & & \text{for all } i \in O\colon P(i) = n_{J(i)} \\
& \sum_{k \in M_i} x_{i,k} = 1 & & \text{for all } i \in O \\
& S_i + \sum_{k \in M_i} x_{i,k} p_{i,k} \leq S_j & & \text{for all } (i,j) \in C \\
& S_i + \sum_{k \in M_i} x_{i,k} p_{i,k} \leq S_j \quad \vee & & \\
& S_j + \sum_{k \in M_j} x_{j,k} p_{j,k} \leq S_i & & \text{for all } (i,j) \in D(x) \\
& S_i \geq 0 & & \text{for all } i \in O \\
& x_{i,k} \in \{0,1\} & & \text{for all } k \in M_i,\ i \in O.
\end{aligned}$$

The model for the problem $FJ\,||\,C_{\max}$ provided in this section does not satisfy the conditions for a mixed integer program formulation. First, constraints (3.4) are disjunctive constraints and secondly, the set of disjunctive constraints is dependent on the machine assignment $x = (x_{i,k})$. In Sections 3.2, 3.3, 3.4, and 3.5, different approaches to the transformation of the formulation provided in this section into a mixed integer program formulation are presented. It is noted, that a linear program formulation without inte-

ger variables cannot be found for the problem $FJ\,||\,C_{\max}$, unless $\mathcal{P} = \mathcal{NP}$. This results from the facts that on the one hand the problem $FJ\,||\,C_{\max}$ is $\mathcal{NP}$-hard and on the other hand linear programming is solvable in polynomial time, for example by means of the interior-point method introduced by Karmarkar in [Kar84].

3.2 Model IPF: Immediate Precedence Formulation

The formulation presented in this section is based on the model of Choi et al. [CC02]. Saidi-Mehrabad et al. [SMF06] and Fattahi et al. [FSMJ07] used a similar formulation. Both models include sequence-dependent set up times, which will be left out in the model presented in this section.

First of all, a dummy job J_0 consisting of m operations with zero processing time is introduced. Each dummy operation marks the initial operation for one of the m machines. The dummy operations do not have any ordering, thus $P(i) = 0$ for all dummy operations i. Let $\tilde{O} := \{1, \ldots, \tilde{N}\}$ be the set of all operations including the m operations of the dummy job J_0, i.e.,

$$\tilde{N} := \sum_{i=0}^{n} n_i,$$

where by definition $n_0 = m$. Furthermore, the index set I_k defined by

$$I_k := \{i \in O \,|\, k \in M_i\}$$

denotes the indices of operations $i \in O$ that can be processed on machine k. Analogously, the index set $\tilde{I}_k := \{i \in \tilde{O} \,|\, k \in M_i\}$ is defined for the extended operation set $\tilde{O}$. In order to transform the formulation for the problem $FJ\,||\,C_{\max}$ developed in Section 3.1 into a mixed integer program, the disjunctive constraints and their dependence on the machine assignment have to be eliminated. A possible approach is to establish an order on each machine by binary variables $y_{i,j,k}$ for all $i \neq j \in I_k$, $k = 1, \ldots, m$, where

$$y_{i,j,k} = \begin{cases} 1, & \text{if operation } i \text{ precedes operation } j \text{ immediately on machine } k \\ 0, & \text{otherwise.} \end{cases}$$

If operation i is assigned to machine k, exactly one operation j is the imme-

diate successor of operation i on machine k. This constraint is given by

$$\sum_{j \in I_k,\, j \neq i} y_{i,j,k} = x_{i,k} \quad \text{for all } k \in M_i,\ i \in \tilde{O}. \tag{3.5}$$

Similarly, if operation j is assigned to machine k, exactly one operation i is the immediate predecessor of operation j on machine k. This constraint is given by

$$\sum_{i \in I_k,\, i \neq j} y_{i,j,k} = x_{j,k} \quad \text{for all } k \in M_j,\ j \in \tilde{O}. \tag{3.6}$$

Jointly, constraints (3.5) and (3.6) define circular orderings of operations on each machine. Thus, the following constraints ensure that each machine processes not more than one operation at a time:

$$S_i + p_{i,k} - M(1 - y_{i,j,k}) \leq S_j \quad \text{for all } i \neq j \in I_k :\ J(j) \neq 0, \\ k = 1, \ldots, m. \tag{3.7}$$

M is a Big-M constant chosen sufficiently large in order to guarantee constraints (3.7) to be valid for arbitrary values of S_i and S_j if $y_{i,j,k} = 0$. A suitable choice of the Big-M constants is discussed in Section 3.6.1. In order to define the dummy operations as starting and ending point for the circular arrangement of operations on each machine, constraints (3.7) are only introduced partially for the dummy operations. Consequently, a feasible sequence of operations on each machine starting with the dummy operation is obtained by constraints (3.7). The remaining constraints are defined analogously to Section 3.1. If necessary, the set of operations O has been replaced by the extended set of operations $\tilde{O}$. Then, the mixed integer program formulation IPF (Immediate Precedence Formulation) for the problem $FJ\,||\,C_{\max}$ is denoted by

$$\begin{aligned} \min \quad & C_{\max} && && \text{(IPF)} \\ \text{s.t.} \quad & \\ & C_{\max} - \sum_{k \in M_i} x_{i,k} p_{i,k} \geq S_i && \text{for all } i \in \tilde{O} : P(i) = n_{J(i)} \\ & S_i + \sum_{k \in M_i} x_{i,k} p_{i,k} \leq S_j && \text{for all } (i,j) \in C \end{aligned}$$

$$
\begin{aligned}
\sum_{k \in M_i} x_{i,k} &= 1 && \text{for all } i \in \tilde{O} \\
S_i + p_{i,k} - M(1 - y_{i,j,k}) &\leq S_j && \text{for all } i \neq j \in \tilde{I}_k : \ J(j) \neq 0, \\
& && \quad k = 1, \ldots, m \\
\sum_{j \in I_k, j \neq i} y_{i,j,k} &= x_{i,k} && \text{for all } k \in M_i, \ i \in \tilde{O} \\
\sum_{i \in I_k, i \neq j} y_{i,j,k} &= x_{j,k} && \text{for all } k \in M_j, \ j \in \tilde{O} \\
S_i &\geq 0 && \text{for all } i \in \tilde{O} \\
y_{i,j,k} &\in \{0,1\} && \text{for all } i \neq j \in \tilde{I}_k, \ k = 1, \ldots, m \\
x_{i,k} &\in \{0,1\} && \text{for all } k \in M_i, \ i \in \tilde{O}.
\end{aligned}
$$

The size of a mixed integer program is determined by the number of variables denoted by $\mathcal{V}$ and the number of constraints denoted by $\mathcal{C}$, whereas the complexity is among other things dependent on the number of binary variables denoted by $\mathcal{V}_b$ and the number of constraints with Big-M constraints denoted by $\mathcal{C}_M$. For the model IPF we have

$$
\begin{aligned}
\mathcal{V}(IPF) &= 1 + |\tilde{O}| + \sum_{k=1}^{m} |\tilde{I}_k| \left(|\tilde{I}_k| - 1 \right) + \sum_{i \in \tilde{O}} |M_i| \\
&= 1 + |O| + m + \sum_{k=1}^{m} |I_k| \left(|I_k| + 1 \right) + \sum_{i \in O} |M_i| + m \\
&\leq 1 + N + m + N(N+1)m + m + Nm \\
&= 1 + 2m + N + 2Nm + N^2 m,
\end{aligned}
$$

and consequently,

$$
\mathcal{V}_b(IPF) \leq 2m + 2Nm + N^2 m.
$$

Furthermore,

$$
\begin{aligned}
\mathcal{C}(IPF) &= n + \sum_{j=1}^{n} (n_j - 1) + |\tilde{O}| + \sum_{k=1}^{m} |I_k| \left(|I_k| - 1 \right) + 2 \sum_{i \in \tilde{O}} |M_i| \\
&\leq n + N - n + |O| + m + N(N-1)m + 2 \sum_{i \in O} |M_i| + 2m \\
&= 2m + 2N + Nm + N^2 m,
\end{aligned}
$$

and

$$\begin{aligned}\mathcal{C}_M(IPF) &= \sum_{k=1}^{m} |I_k| \left(|I_k| - 1\right) \\ &\leq N^2 m - Nm.\end{aligned}$$

Therefore, $\mathcal{V}(IPF)$, $\mathcal{V}_b(IPF)$, $\mathcal{C}(IPF)$, $\mathcal{C}_M(IPF) \in \mathcal{O}(N^2 m)$.

3.3 Model GPF: General Precedence Formulation

In this section, another approach to deal with the disjunctive constraints and their dependence on the machine assignment is presented. Here, the binary variables $y_{i,j,k}$ for all $i \neq j \in I_k,\ k = 1, \ldots, m$ are defined by

$$y_{i,j,k} = \begin{cases} 1, & \text{if operation } i \text{ precedes operation } j \text{ on machine } k \\ 0, & \text{otherwise.} \end{cases}$$

Thus, the notion of immediate precedence is relaxed and instead, a general precedence order is imposed. For each pair i, j of operations assigned to the same machine k, either operation i has to be completed before operation j starts or operation j has to be completed before operation i starts. These constraints are given by

$$x_{i,k} + x_{j,k} \leq 1 + y_{j,i,k} + y_{i,j,k} \quad \text{for all } i \neq j \in I_k,\ k = 1, \ldots, m. \tag{3.8}$$

If operation i and operation j are scheduled on the same machine, then $x_{i,k} = x_{j,k} = 1$ for some k, and thus, $y_{i,j,k} = 1$ or $y_{j,i,k} = 1$ in order to obtain a valid constraint. The additional constraints

$$S_i + p_{i,k} - M\left(1 - y_{i,j,k}\right) \leq S_j \quad \text{for all } i \neq j \in I_k,\ k = 1, \ldots, m \tag{3.9}$$

ensure that each machine processes at most one job at a time. Again, M is a Big-M constant taken sufficiently large in order to guarantee constraint (3.9) to be satisfied for arbitrary values of S_i and S_j if $y_{i,j,k} = 0$. For a discussion concerning the suitable choice of the Big-M constants it is referred to Section 3.6.1. The remaining constraints are defined analogously to model IPF. Then, the mixed integer program formulation GPF (General Precedence

Formulation) for the problem $FJ\,|\,|\,C_{\max}$ is given by

$$
\begin{aligned}
\min \quad & C_{\max} && && \text{(GPF)}\\
\text{s.t.} \quad & \\
C_{\max} - \sum_{k\in M_i} x_{i,k}p_{i,k} &\geq S_i && \text{for all } i \in O\colon P(i) = n_{J(i)}\\
S_i + \sum_{k\in M_i} x_{i,k}p_{i,k} &\leq S_j && \text{for all } i,j \in C\\
\sum_{k\in M_i} x_{i,k} &= 1 && \text{for all } i \in O\\
S_i + p_{i,k} - M\,(1 - y_{i,j,k}) &\leq S_j && \text{for all } i \neq j \in I_k,\ k = 1,\ldots,m\\
x_{i,k} + x_{j,k} - y_{j,i,k} - y_{i,j,k} &\leq 1 && \text{for all } i \neq j \in I_k,\ k = 1,\ldots,m\\
S_i &\geq 0 && \text{for all } i \in O\\
y_{i,j,k} &\in \{0,1\} && \text{for all } i \neq j \in I_k,\ k = 1,\ldots,m\\
x_{i,k} &\in \{0,1\} && \text{for all } k \in M_i,\ i \in O.
\end{aligned}
$$

Since there is no need for m dummy operations in the model GPF, the number of (binary) variables is slightly reduced in relation to the model IPF. However, $\mathcal{V}(GPF)$, $\mathcal{V}_b(GPF) \in \mathcal{O}(N^2m)$. The number of constraints $\mathcal{C}$ and the number of constraints with Big-M constants $\mathcal{C}_M$ of the model GPF is given by

$$
\begin{aligned}
\mathcal{C}(GPF) &= n + \sum_{j=1}^{n}(n_j - 1) + |O| + 2\sum_{k=1}^{m}|I_k|\,(|I_k| - 1)\\
&\leq n + N - n + |O| + m + 2N(N-1)m\\
&= m + 2N + 2N^2m - 2Nm,
\end{aligned}
$$

and

$$
\begin{aligned}
\mathcal{C}_M(GPF) &= \sum_{k=1}^{m}|I_k|\,(|I_k| - 1)\\
&\leq N^2m - Nm.
\end{aligned}
$$

Consequently, $\mathcal{C}(GPF)$, $\mathcal{C}_M(GPF) \in \mathcal{O}(N^2m)$, too. Regarding the number of (binary) variables and the number of (Big-M) constraints there is

no significant difference between model IPF and model GPF. Nevertheless, the structure of constraints (3.8) and (3.9) of model GPF is different to the structure of constraints (3.5), (3.6), and (3.7) of model IPF. The impact of these structural differences on the performance of both models is evaluated in Section 3.7.

3.4 Model TEF: Time-Expanded Formulation

Computer-based mixed integer program solvers consider linear relaxations of a given mixed integer program in the process of solving it to optimality (see Section 2.1). In case of modeling disjunctive constraints by means of Big-M constants, the Big-M constraints are not completely effective for a feasible solution of the linear relaxation. Due to a large value of the Big-M constant, fractional values for the binary variables contained in the Big-M constraints potentially lead to a valid constraint for both options of the disjunction. Consequently, these Big-M constraints do not matter at all in the linear relaxation.

According to the disjunctive constraints of the flexible job shop scheduling problem, several operations are potentially scheduled at the same time on the same machine. Thus, the quality of the linear relaxation is significantly decreased by the use of Big-M constraints until a feasible integer solution to the binary variables incorporated in the Big-M constraint is found. Furthermore, a suitable choice of the Big-M constants, that is, as small as possible, is crucial for the quality of the model. Both model IPF and model GPF require the application of Big-M constants in order to model the disjunctive constraints. In this section, a time-expanded model for the flexible job shop scheduling problem is presented, that manages without any Big-M constants and consequently avoids the difficulties described above.

The model presented in this section is based on the discretization of time. It is always possible to transform the processing times and consequently the starting times to integer values. Then, the size of a time unit has to be chosen depending on the required accuracy of the resulting schedule. The choice of the size of a time unit as the greatest common divisor of the processing times of all operations always leads to a schedule without any loss in accuracy. Consequently, the discretization is a reasonable assumption and does not lead to further restrictions. In the model the time horizon $0, \ldots, T$

is considered, where T is an upper bound for the makespan. Obviously, a feasible, but in most cases not optimal choice for T is the sum of the maximal processing times of all operations,

$$T := \sum_{i=1,\dots,N} \max_{k \in M_i} \{p_{i,k}\}.$$

A better choice for T will be discussed in Section 3.6.2.

First of all, binary variables $x_{i,k,t}$ for all $k \in M_i$, $i \in O$, $t = 0, \dots, T$ are introduced marking the beginning of the processing of an operation on a dedicated machine:

$$x_{i,k,t} = \begin{cases} 1, & \text{if operation } i \text{ starts at time } t \text{ on machine } k \\ 0, & \text{otherwise.} \end{cases}$$

Thereupon, the makespan is now defined by

$$\sum_{k \in M_i} \sum_{t=0}^{T} x_{i,k,t}(t + p_{i,k}) \leq C_{\max} \quad \text{for all } i \in O \colon P(i) = n_{J(i)}. \tag{3.10}$$

For a pair of operations $(i, j) \in C$, that is, two consecutive operations of a job, it has to be ensured that the starting time of operation j, given by

$$\sum_{k \in M_j} \sum_{t=0}^{T} x_{j,k,t} \cdot t,$$

is not earlier than the completion time of operation i, given by

$$\sum_{k \in M_i} \sum_{t=0}^{T} x_{i,k,t}(t + p_{i,k}).$$

Consequently, the precedence constraints are given by

$$\sum_{k \in M_i} \sum_{t=0}^{T} x_{i,k,t}(t + p_{i,k}) \leq \sum_{k \in M_j} \sum_{t=0}^{T} x_{j,k,t} \cdot t \quad \text{for all } i, j \in C. \tag{3.11}$$

Furthermore, each operation has to be scheduled on exactly one machine at exactly one point in time, which is assured by

$$\sum_{k \in M_i} \sum_{t=0}^{t} x_{i,k,t} = 1 \quad \text{for all } i \in O. \tag{3.12}$$

In contrast to model IPF and model GPF, a formulation of the disjunctive constraints without Big-M constants is possible. On each machine at every point in time at most one operation is allowed to be scheduled. Since the binary variables $x_{i,k,t}$ represent only the starting time t of operation i on machine k, it has to be ensured that at most one operation i starts in the time period $[t - p_{i,k} + 1, t]$ on machine k. The disjunctive constraints are therefore given by

$$\sum_{i \in I_k} \sum_{\tau=t-p_{i,k}+1}^{t} x_{i,k,\tau} \leq 1 \quad \text{for all } t = 0, \ldots, T,\ k = 1, \ldots, m. \tag{3.13}$$

Finally, the mixed integer program formulation TEF (Time-Expanded Formulation) for the problem $FJ\,|\,|\,C_{\max}$ is given by

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

s.t.

$$\sum_{k \in M_i} \sum_{t=0}^{T} x_{i,k,t}(t + p_{i,k}) \leq C_{\max} \qquad \text{for all } i \in O\colon P(i) = n_{J(i)}$$

$$\sum_{k \in M_i} \sum_{t=0}^{T} x_{i,k,t}(t + p_{i,k}) \leq \sum_{k \in M_j} \sum_{t=0}^{T} x_{j,k,t} \cdot t \quad \text{for all } i, j \in C$$

$$\sum_{k \in M_i} \sum_{t=0}^{T} x_{i,k,t} = 1 \qquad \text{for all } i \in O$$

$$\sum_{i \in I_k} \sum_{\tau=t-p_{i,k}+1}^{t} x_{i,k,\tau} \leq 1 \qquad \text{for all } t = 0, \ldots, T,\ k = 1, \ldots, m$$

$$x_{i,k,t} \in \{0, 1\} \qquad \text{for all } k \in M_i,\ i \in O,\ t = 0, \ldots, T.$$

Obviously, there are no Big-M constants in the model TEF, which is a significant advantage in comparison to the models IPF and GPF. Furthermore, except for the variable $C_{\max}$ the model TEF contains solely binary

variables. More precisely, the number of binary variables is given by

$$\mathcal{V}_b(TEF) = (T+1)\sum_{i\in O} M_i \leq T|O|m = TNm.$$

Therefore, $\mathcal{V}_b(TEF)$, $\mathcal{V}(TEF) \in \mathcal{O}(TNm)$ and it depends on the ratio of the number of operations N and the number of points in time T whether the number of (binary) variables of the model TEF is smaller than the number of (binary) variables of the models IPF and GPF. For $T < N$, the model TEF constitutes an improvement in the number of (binary) variables. Furthermore, the number of constraints is given by

$$\begin{aligned}\mathcal{C}(TEF) &= n + \sum_{j=1}^{n}(n_j - 1) + |O| + (T+1)m \\ &\leq n + N - n + N + Tm \\ &= 2N + Tm.\end{aligned}$$

Consequently, $\mathcal{C}(TEF) \in \mathcal{O}(N + Tm)$ and again the number of constraints in the model TEF depends on the size of T.

In Section 3.6.2 the choice of the upper bound T will be commented further. It is evident, that the size of T is decisive for the efficiency of the model TEF.

3.5 Model MPF: Machine-Position Formulation

In this section an additional formulation for the flexible job shop scheduling problem is suggested. This approach is based on the idea of weakening the dependency of model TEF on the upper bound T. Instances of the flexible job shop scheduling problem consisting of operations with strongly varying processing times lead to a large model size, since each point in time is modeled.

In order to cope with this difficulty, no longer binary variables representing the starting time of an operation on a machine are modeled, but binary variables representing the relative position of an operation on a machine with respect to the other operations assigned to that machine. Recall that the index set I_k contains the indices of all operations that can be assigned to machine k. Consequently, there are $|I_k|$ positions on machine k. Obviously,

on each machine at most N operations can be scheduled, and the number of positions on each machine is therefore bounded by N. Thus, strongly varying processing times have no influence on the model size.

In order to model the machine positions, binary variables $x_{i,k,p}$ for all $p_k = 1, \ldots, |I_k|$, $k = 1, \ldots, m$, $i \in O$ are introduced:

$$x_{i,k,p} = \begin{cases} 1, & \text{if operation } i \text{ is scheduled for position } p \text{ on machine } k \\ 0, & \text{otherwise.} \end{cases}$$

The definition of the makespan and the precedence constraints is analogue to the models IPF and GPF:

$$C_{\max} \geq S_i + \sum_{k \in M_i} \sum_{p=1}^{|I_k|} x_{i,k,p} p_{i,k} \quad \text{for all } i \in O\colon P(i) = n_{J(i)}, \tag{3.14}$$

$$S_i + \sum_{k \in M_i} \sum_{p=1}^{|I_k|} x_{i,k,p} p_{i,k} \leq S_j \quad \text{for all } (i,j) \in C. \tag{3.15}$$

Furthermore, each operation has to be assigned to exactly one position, which is ensured by

$$\sum_{k=1}^{m} \sum_{p=1}^{|I_k|} x_{i,k,p} = 1 \quad \text{for all } i \in O. \tag{3.16}$$

Additionally, at most one operation can be assigned to each position, given by the constraints

$$\sum_{i \in O} x_{i,k,p} \leq 1 \quad \text{for all } p = 1, \ldots, |I_k|,\ k = 1, \ldots, m. \tag{3.17}$$

The positions on each machine have to be filled subsequently, that is, an operation is only allowed to be assigned to a position on a machine if the preceding position is already filled. This condition is ensured by

$$\sum_{i \in O} x_{i,k,p} \leq \sum_{i \in O} x_{i,k,p-1} \quad \text{for all } p = 2, \ldots, |I_k|,\ k = 1, \ldots, m. \tag{3.18}$$

Finally, in order to interconnect the machine position variables with the starting time variables and to enforce a feasible schedule, non-overlapping

constraints are defined by

$$S_i + p_{i,k} - M(2 - x_{i,k,p-1} - x_{j,k,p}) \leq S_j \tag{3.19}$$

for all $p = 2, \ldots, |I_k|$, $i \neq j \in I_k$, and $k = 1, \ldots, m$. If the operations i and j are assigned to the same machine k for consecutive positions $p-1$ and p, then the starting time S_j of operation j must not be earlier than the completion time $S_i + p_{i,k}$ of operation i. Again, M is a Big-M constant taken sufficiently large in order to guarantee constraints (3.19) to be valid if at least one of the machine position variables $x_{i,k,p}$ and $x_{j,k,p-1}$ is zero, that is, operations i and j are not assigned to consecutive positions on the same machine and consequently, a non-overlapping constraint does not have to be taken into account. Thus, the mixed integer program formulation MPF (Machine-Position Formulation) for the problem $FJ\,||\,C_{\max}$ is given by

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

s.t.

$$\begin{aligned}
S_i + \sum_{k \in M_i} \sum_{p=1}^{|I_k|} x_{i,k,p} p_{i,k} &\leq C_{\max} && \text{for all } i \in O\colon P(i) = n_{J(i)} \\
S_i + \sum_{k \in M_i} \sum_{p=1}^{|I_k|} x_{i,k,p} p_{i,k} &\leq S_j && \text{for all } (i,j) \in C \\
\sum_{k \in M_i} \sum_{p=1}^{|I_k|} x_{i,k,p} &= 1 && \text{for all } i \in O \\
\sum_{i \in O} x_{i,k,p} &\leq 1 && \text{for all } p = 1, \ldots, |I_k|, \\
& && \quad k = 1, \ldots, m \\
\sum_{i \in O} x_{i,k,p} - \sum_{i \in O} x_{i,k,p-1} &\leq 0 && \text{for all } p = 2, \ldots, |I_k|, \\
& && \quad k = 1, \ldots, m \\
M(2 - x_{i,k,p-1} - x_{j,k,p}) + S_j &\geq S_i + p_{i,k} && \text{for all } p = 2, \ldots, |I_k|, \\
& && \quad i \neq j \in I_k, \\
& && \quad k = 1, \ldots, m \\
S_i &\geq 0 && \text{for all } i \in O \\
x_{i,k,p} &\in \{0,1\} && \text{for all } p = 1, \ldots, |I_k|, \\
& && \quad k \in M_i,\ i \in O.
\end{aligned}$$

The number of variables $\mathcal{V}(MPF)$ is given by

$$\mathcal{V}(MPF) = 1 + |O| + \sum_{i \in O} \sum_{k \in M_i} |I_k| \leq 1 + N + N^2 m,$$

and the number of binary variables $\mathcal{V}_b(MPF)$ is given by

$$\mathcal{V}_b(MPF) = \sum_{i \in O} \sum_{k \in M_i} |I_k| \leq N^2 m.$$

Thus, $\mathcal{V}(MPF)$, $\mathcal{V}_b(MPF) \in \mathcal{O}(N^2 m)$. Furthermore, the number of constraints $\mathcal{C}(MPF)$ is given by

$$\begin{aligned}
\mathcal{C}(MPF) &= n + \sum_{j=1}^{n} (n_j - 1) + |O| + \sum_{k=1}^{m} |I_k| + \sum_{k=1}^{m} (|I_k| - 1) \\
&\quad + \sum_{k=1}^{m} \left((|I_k| - 1) \left(|I_k| \left(|I_k| - 1 \right) \right) \right) \\
&= n + N - n + N + 2 \sum_{k=1}^{m} |I_k| - m + \sum_{k=1}^{m} \left(|I_k|^3 - 2|I_k|^2 + |I_k| \right) \\
&\leq 2N + 3mN - m + mN^3 - 2mN^2,
\end{aligned}$$

and the number of constraints with Big-M constants $\mathcal{C}_b(MPF)$ is given by

$$\mathcal{C}_M(MPF) = \sum_{k=1}^{m} \left((|I_k| - 1) \left(|I_k| \left(|I_k| - 1 \right) \right) \right) = mN^3 - 2mN^2 + mN.$$

Therefore, $\mathcal{C}(MPF)$,$\mathcal{C}_M(MPF) \in \mathcal{O}(mN^3)$. The transformation from points in time to positions on machines detaches the model size from the size of processing times. However, this is already achieved for models IPF and GPF and additionally, the number of constraints with Big-M constants for model MPF is by a factor of N higher than the number of constraints with Big-M constants of models IPF and GPF.

3.6 Performance Improvements

In this section the models from Sections 3.2 - 3.5 are considered again with the intention of performance improvement. In the literature, models for the flexible job shop scheduling problem are only sparsely covered, and further-

	$\mathcal{V}$	$\mathcal{V}_b$	$\mathcal{C}$	$\mathcal{C}_M$
IPF	$\mathcal{O}(N^2m)$	$\mathcal{O}(N^2m)$	$\mathcal{O}(N^2m)$	$\mathcal{O}(N^2m)$
GPF	$\mathcal{O}(N^2m)$	$\mathcal{O}(N^2m)$	$\mathcal{O}(N^2m)$	$\mathcal{O}(N^2m)$
TEF	$\mathcal{O}(TNm)$	$\mathcal{O}(TNm)$	$\mathcal{O}(n+Tm)$	0
MPF	$\mathcal{O}(N^2m)$	$\mathcal{O}(N^2m)$	$\mathcal{O}(N^3m)$	$\mathcal{O}(N^3m)$

Table 3.1: Model sizes of the models IPF, GPF, TEF, and MPF

more, detailed discussions of different models and possible approaches for performance improvement are up to our knowledge not considered at all. On this account and with the intention to employ one of the models as a basis for an efficient approximation algorithm, a detailed analysis seems worthwhile.

In Section 3.6.3 and Section 3.6.4, structural improvements are discussed for the model IPF and the model GPF, respectively. Furthermore, in Section 3.6.1, the choice of Big-M constants for the models IPF, GPF, and MPF is considered in detail and in Section 3.6.2, the choice of an upper bound T for the number of points in time is analyzed explicitly. In order to summarize the results from the previous sections, an overview of the model sizes of the models IPF, GPF, TEF, and MPF is given in Table 3.1. The essentially different structure of model TEF becomes obvious in the model size, too. Furthermore, with respect to the size, model MPF is dominated by the other models. A detailed comparison of the performance of the different models is given in Section 3.7.

3.6.1 Suitable Choice of Big-M Constants

Constraints with Big-M constants occur in the models IPF, GPF, and MPF. As pointed out earlier, it is desirable to choose the Big-M constants as small as possible in order to strengthen the linear relaxation. Obviously, a feasible choice of a Big-M constant is given by

$$M := \sum_{i \in O} \max_{k \in M_i} p_{i,k}. \tag{3.20}$$

Since the dummy operations used in model IPF have zero processing time, this holds for all three models IPF, GPF, and MPF. Smaller Big-M constants can be achieved by assigning an individual Big-M constant to each constraint instead of using a single Big-M constant for all constraints. Let $M_{\max}$ be an upper bound for the makespan of the flexible job shop scheduling problem. A possible upper bound for the makespan is given by the sum of the maximal processing times of all operations as seen in equation (3.20). Additionally, an upper bound for the makespan can be achieved by a list scheduling heuristic, which is discussed in detail in Section 3.6.2. Due to the job structure, each operation is in a sequence of operations that have to be processed before and after the operation. By means of this basic observation, the range of possible starting times for an operation can be narrowed down. Let $M_{i,j,k}$ denote the Big-M constant for the corresponding constraint in (3.7), (3.9), and (3.19). After operation i is completed the operations in the sequence of job $J(i)$ with a position greater than $P(i)$ still have to be processed (cf. Example 3.1). Therefore,

$$S_i + p_{i,k} \leq M - \sum_{j \in A_i} \min_{l \in M_j} p_{j,l} + p_{i,k}, \tag{3.21}$$

where $A_i := \{j \in O \,|\, J(j) = J(i) \wedge P(j) \geq P(i)\}$. Additionally, before operation j is started, the operations in the sequence of job $J(j)$ with a position smaller than $P(j)$ first have to be processed. Consequently,

$$S_j \geq \sum_{i \in B_j} \min_{l \in M_i} p_{i,l}, \tag{3.22}$$

where $B_j := \{i \in O \,|\, J(i) = J(j) \wedge P(i) < P(j)\}$.

Example 3.1. *Consider two jobs J_1 and J_2 with three operations each, whereby each operation has unit processing time. After six time units at the latest all six operations are completed regardless of the machines the operations have to be processed on. Consequently, $M_{\max} = 6$ is a suitable choice for an upper bound for the makespan. Furthermore, operation $O_{1,1}$ has to be processed in the interval $[0, 4]$, otherwise the processing times of the subsequent operations $O_{1,2}$ and $O_{1,3}$ would lead to an overall processing time greater than $M_{\max}$. Consequently, the individual choice of a Big-M constant $M_{1,1,k} = 4$ improves upon the initial choice of an overall Big-M constant of*

$M = 6$, leading to a stronger linear relaxation. In Figure 3.1, the possible reduction of Big-M constants for job J_1 is depicted.

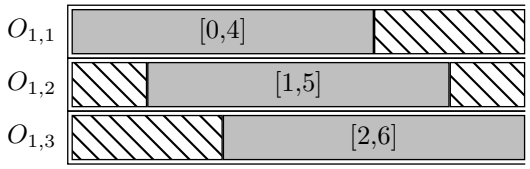


Figure 3.1: Reduction of Big-M constants

With the help of equations (3.21) and (3.22), better Big-M constants can be achieved.

$$S_i + p_{i,k} - S_j \leq M - \sum_{g \in A_i} \min_{l \in M_g} p_{g,l} + p_{i,k} - \sum_{h \in B_j} \min_{l \in M_h} p_{h,l},$$

and therefore the Big-M constants $M_{i,j,k}$ for the constraints (3.7) and (3.9) can be defined by

$$M_{i,j,k} := M - \sum_{g \in A_i} \min_{l \in M_g} p_{g,l} + p_{i,k} - \sum_{h \in B_j} \min_{l \in M_h} p_{h,l}.$$

For model MPF there has to be made a minor differentiation, since constraints (3.19) are given by

$$S_i + p_{i,k} - M(2 - x_{i,k,p} - x_{j,k,p-1}) \leq S_j.$$

Denote the individual Big-M constants for the model MPF by $M'_{i,j,k}$ and define them by

$$M'_{i,j,k} := \begin{cases} \frac{1}{2} M_{i,j,k} & \text{if } M_{i,j,k} < 0 \\ M_{i,j,k} & \text{otherwise.} \end{cases}$$

If $M_{i,j,k} < 0$, further restrictions can be imposed. If $M_{i,j,k} < 0$, it is ensured that operation j cannot be scheduled (immediately) before operation i on machine k in a feasible solution. Consequently, $y_{j,i,k}$ is initially set to zero for models IPF and GPF. The initial fixing of variables can lead to a further performance improvement.

For model MPF, additional constraints given by

$$x_{i,k,p'} + x_{j,k,p} \leq 1 \quad \text{for all } 1 \leq p' < p \leq |I_k|$$

can be introduced in case $M'_{i,j,k} < 0$, since operation i cannot be scheduled before operation j on machine k.

3.6.2 Suitable Choice of an Upper Bound T

The time-expanded formulation of model TEF requires the choice of a fixed time period $[0, T]$ in which all operations are processed. Obviously, T is equivalent to an upper bound for the makespan and therefore

$$T := \sum_{i \in O} \max_{k \in M_i} p_{i,k}$$

is a feasible choice for T. Still, it is desirable to find an upper bound as small as possible leading to a reduction in the number of variables, since $\mathcal{V}(TEF) \in \mathcal{O}(TNm)$. A possible approach to reduce the upper bound for the makespan is to apply a list scheduling heuristic. The basic idea of list scheduling is to prepare an ordered list of operations, and then schedule the operations in this order according to a given rule.

An ordered list can be generated by solving the linear relaxation of the model IPF or the model GPF, and then listing the operations according to their starting times. The solution of the linear relaxation of model IPF or model GPF provides at least a feasible ordering of the operations according to their job sequences, since constraints (3.6) ensure that

$$S_i \leq S_j \quad \text{for all } (i, j) \in C.$$

The operations are now one after another selected from the ordered list and scheduled by the following rule. Consider all machines from the set of available machines of the selected operation and choose the machine on which the operation can be scheduled as early as possible. The makespan $C_{\max}$ of the resulting schedule is then assigned to T. The list scheduling heuristic is formally summarized in Algorithm 1.

Lemma 3.2. *Algorithm 1 is a polynomial-time algorithm and produces a feasible solution to the flexible job shop scheduling problem.*

Proof. Since there exist polynomial-time algorithms for linear programming [NW88], polynomial-time algorithms for sorting (for example bubble sort with worst case performance $\mathcal{O}(N^2)$) and steps 3 - 12 have complexity

Algorithm 1 List Scheduling Heuristic for $FJ\,||\,C_{\max}$

1: Find an optimal solution S^{LP} to the linear relaxation of model GPF.
2: Index the operations such that $S_1^{LP} \leq S_2^{LP} \leq \cdots \leq S_N^{LP}$.
3: **for all** $i = 1, \ldots, N$ **do**
4: $\quad S_i = \sum_{i \in O} \max_{k \in M_i} \{p_{i,k}\}$
5: $\quad$ **for all** $m \in M_i$ **do**
6: $\quad\quad$ Set s to the maximum of the availability of machine m and the completion time of the operation with position $P(i)-1$ of job $J(i)$.
7: $\quad\quad$ **if** $s < S_i$ **then**
8: $\quad\quad\quad S_i = s$;
9: $\quad\quad$ **end if**
10: $\quad$ **end for**
11: **end for**
12: Determine the makespan $C_{\max}$ and set $T = C_{\max}$.

$\mathcal{O}(Nm)$, the list scheduling heuristic presented in Algorithm 1 is a polynomial-time algorithm.

By constraints (3.6), the ordering of operations obtained from the linear relaxation of model GPF respects the imposed job sequence for each job. According to step 6, each operation i is not scheduled before the preceding operation of job $J(i)$ is completed and a machine is available. Consequently, each machine processes at most one operation at a time and the operations are scheduled according to the precedence constraints. The choice of S_i in step 4 guarantees that the condition in step 7 is satisfied at least once and therefore, each operation is assigned to exactly one machine. As a result, Algorithm 1 generates a feasible solution to the flexible job shop scheduling problem. □

By Lemma 3.2, Algorithm 1 is applicable as preprocessing step in order to determine an upper bound for the makespan. Additionally, with the help of the solution from the list scheduling heuristic, a warmstart can be performed, that is, an existing solution of a similar problem is used as a start basis [Kal02].

Furthermore, the number of variables of the model TEF can be reduced with the help of the considerations from Section 3.6.1. For each operation i

the earliest possible starting time α_i is given by

$$\alpha_i := \sum_{j \in B_i} \min_{l \in M_j} p_{j,l}.$$

In addition, the subsequent operations of job $J(i)$ cannot be processed earlier than operation i is completed. Since T is an upper bound for the maximal completion time, the latest starting time β_i of operation i is defined by

$$\beta_i := T - \sum_{j \in A_i} \min_{l \in M_j} p_{j,l} + p_{i,k}.$$

Therefore, the definition of the binary variables $x_{i,k,t}$ can be reduced to

$$x_{i,k,t} \in \{0,1\} \quad \text{for all } k \in M_i,\ i \in O,\ t = \alpha_i, \ldots, \beta_i.$$

If each job consists of only one operation, this is the same definition as in the basic model TEF.

3.6.3 Additional Constraints for IPF

Constraints (3.5), (3.6), and (3.7) of model IPF assure that on each machine at most one operation is scheduled at a time. Since constraints (3.7) incorporate Big-M constants, these constraints are potentially not completely effective until the binary variables $y_{i,j,k}$ take integer values, as described in Section 3.4. Consequently, it is not ensured, that each machine processes at most one operation at a time, although the constraints (3.5), (3.6), and (3.7) are satisfied in the linear relaxation. In the following, this situation is illustrated by means of graph theory. For an arbitrarily chosen solution $x = (x_{i,k})$ with respect to the mixed integer programming formulation IPF, consider for each machine k a weighted digraph $G_k(x) = (V_k(x), E_k(x))$, dependent on the machine assignment $x = (x_{i,k})$. The node set $V_k(x)$ is given by

$$V_k(x) = \{i \in \tilde{O} \,|\, x_{i,k} = 1\},$$

that is, each operation assigned to machine k is identified by a node. The edge set $E_k(x)$ is given by

$$E_k(x) = \{(i,j) \,|\, i,j \in V_k(x)\},$$

and the weight function $f_k : E \to \mathbb{R}$ is given by

$$f_k((i,j)) = y_{i,j,k}.$$

Constraints (3.5) and (3.6) correspond to the fact that for each node the sum of incoming edges and the sum of outgoing edges equals one. Furthermore, constraints (3.7) ensure a closed path using only edges with weight $y_{i,j,k} > 0$, visiting each node exactly once, starting and ending at the dummy node. With the help of this visualization, Example 3.3 illustrates the problem arising from the constraints with Big-M constants.

Example 3.3. *Consider a setting with $n = 6$, $n_i = 1$, $i = 1, \ldots, n$ and thus, each of the six jobs has exactly one operation and $N = 6$. Denote the operations by $O_1, \ldots, O_6$. Furthermore, $m = 2$ and $M_1 = \cdots = M_5 = \{\mu_1\}$ and $M_6 = \{\mu_2\}$. Each operation has unit processing time, $p_i = 1$, $i = 1, \ldots, N$ and for each machine there is an additional dummy operation O_0 and O_7, respectively, with zero processing time. Consequently, $M = \sum_{i=1}^{N} p_i = 6$. Consider machine μ_1. A feasible solution of the linear relaxation of the model IPF with respect to the ordering of operations on machine μ_1 is given by*

$$y_{i,j,1} = \begin{cases} \frac{1}{2}, & \text{for all } (i,j) : \ (0 \leq i,j \leq 2 \ \vee \ 3 \leq i,j \leq 5) \wedge i \neq j, \\ 0, & \text{otherwise.} \end{cases}$$

Constraints (3.5) and (3.6) are satisfied, that is, the sum of outgoing edges and the sum of incoming edges is equal to one for each node, since each node has two outgoing edges with weight $\frac{1}{2}$ and two incoming edges with weight $\frac{1}{2}$. Furthermore, for $S_0 = 0$, $S_1 = 0$, $S_2 = 1$, $S_3 = 0$, $S_4 = 1$, and $S_5 = 2$, constraints (3.7),

$$S_i + p_{i,k} - M(1 - y_{i,j,k}) \leq S_j,$$

are satisfied in the linear relaxation, since the constraints are weakened by $\frac{1}{2}M = 3$ for each edge with $y_{i,j,k} = \frac{1}{2}$ and by $M = 6$ for each edge with $y_{i,j,k} = 0$. Figure 3.3 depicts the graph $V_1(x)$.

The bottom line is that potentially the Big-M constraints are not effective at all, as seen in Example 3.3. Such examples can be found for many applications of Big-M constraints. Therefore, in order to strengthen the model IPF, additional constraints are introduced, prohibiting situations as presented in

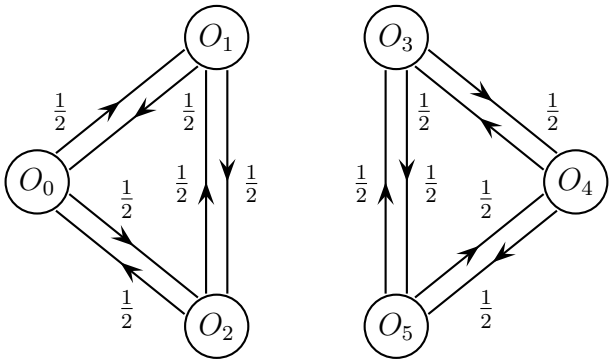


Figure 3.2: Feasible solution to the linear relaxation

Example 3.3. Consider an arbitrary subset $A \subseteq \tilde{O}$ of operations and any machine $k \in M$. If at least one operation $i \in A$ is assigned to machine k, that is, $x_{i,k} = 1$, and at least one operation $j \in \tilde{O} \setminus A$ is assigned to machine k, then the sum of outgoing edges from the node set A to the node set $\tilde{O} \setminus A$ is required to be greater than one, that is,

$$\sum_{i \in A,\, j \in \tilde{O} \setminus A} y_{i,j,k} \geq z_{A,k} \quad \text{for all } A \subseteq \tilde{O},\, k = 1, \ldots, m, \tag{3.23}$$

where $z_{A,k} \in \{0, 1\}$ is a binary variable and

$$1 + z_{A,k} \geq x_{i,k} + x_{j,k} \quad \text{for all } i \in A,\, j \in \tilde{O} \setminus A,\, A \subseteq \tilde{O},\, k = 1, \ldots, m. \tag{3.24}$$

By means of the additional constraints (3.23) and (3.24), formulation IPF is strengthened. In order to show that these constraints indeed present valid inequalities tightening the formulation, Example 3.3 is considered again. The feasible solution to the linear relaxation given in Example 3.3 is obviously no longer feasible with the additional constraints (3.23) and (3.24). Consider the subset of operations $A = \{O_0, O_1, O_2\}$. Then there is at least one operation in A assigned to machine μ_1 and one operation in $\tilde{O} \setminus A$ assigned to machine μ_1. Therefore, by constraints (3.24) $z_{A,\mu_1} = 1$ and further, constraint (3.23),

$$\sum_{i \in A, j \in \tilde{O} \setminus A} y_{i,j,k} \geq z_{A,k} = 1,$$

ensures that at least one connecting edge between the nodes from set A

and the nodes from set $\tilde{O} \setminus A$ is used. Consequently, the solution given in Example 3.3 is no longer a feasible solution to the linear relaxation and therefore, constraints (3.23) and (3.24) present valid inequalities tightening the formulation.

Since these constraints are introduced for each subset $A \subseteq \tilde{O}$ and for each machine the number of additional constraints (3.23) and (3.24) is in $\mathcal{O}\left(mN^2 2^N\right)$ and the number of additional variables $z_{A,k}$ is given by $m2^N$. Consequently, in contrast to the structural improvement achieved by the additional constraints, the model size is now growing exponentially with respect to the number of operations N and it may be expected that the benefit of the additional valid inequalities gets eaten up by the loss of speed due to the increasing number of constraints and variables.

The linear relaxation of model IPF with constraints (3.23) and (3.24) can be solved as follows. First, the linear relaxation is solved without constraints (3.23) and (3.24). Secondly, it is searched for a constraint from the set of constraints (3.23) and (3.24) that is violated for this solution. If such a constraint is found, it is added to the linear relaxation. Now, the linear relaxation is solved again and the process repeats until it is proved that no constraint is violated by the current solution of the linear relaxation. Consequently, the last solution is an optimal solution of the linear relaxation of model IPF with constraints (3.23) and (3.24). The problem of finding a violated constraint or proving that no such constraint exists is also known as the separation problem. For a formal definition see [GLS88, page 48]. Furthermore, Grötschel et al. proved the equivalence of optimization and separation with respect to the polynomial time solvability [GLS88, p. 174]. Consequently, the optimal solution of the linear relaxation of model IPF with constraints (3.23) and (3.24) can be found in polynomial time, if the separation problem is solvable in polynomial time.

In the following, another approach for an improved formulation of model IPF is presented with the intention of avoiding the ramification of the exponentially many inequalities. This approach is based on the adaption of the equivalence of optimization and separation described above to the traveling salesman problem, cf. for example [DFJ54]. When a feasible solution for the linear relaxation of model IPF is found, for each machine the corresponding graph introduced at the beginning of this section is checked for cycles. At this, only edges (i, j) with $y_{i,j,k} > 0$ and operations i that are already

assigned to a certain machine k, that is, $x_{i,k} = 1$, are considered. Consequently, a branching strategy branching first on the machine assignment variables $x_{i,k}$ supports this approach in a positive manner. For a detailed description of an improved branching strategy see Section 3.6.5. If for machine k a cycle exists, that does not contain all operations assigned so far to machine k, and the sum of outgoing edges is smaller than one, an additional constraint is introduced locally in the subtree of the branch and cut search. More precisely, assume that on machine k there is a cycle C given by

$$C = \{i_1, i_2, \ldots, i_{c-1}, i_c = i_1\}$$

with $y_{i_j,i_{j+1},k} > 0$ for $j = 1, \ldots, c-1$, and $c < n'$, where n' is the number of operations i that are assigned to machine k. Now, if

$$\sum_{i \in C, i' \notin C} y_{i,i',k} < 1,$$

the additional constraint

$$\sum_{i \in C, i' \notin C} y_{i,i',k} \geq 1$$

is introduced locally in the subtree of the branch and cut search. This dynamic cut-adding approach is formally summarized in Algorithm 2. The procedure stated above does not necessarily find all cycles, but for the separation of the solution of the linear relaxation it is sufficient. For the practical implementation of the developed mixed integer programming formulations, the CPLEX environment provided by IBM ILOG [cpl10] is chosen. For a more detailed description of the implementation, see Section 3.7. The utilized optimizer provides the functionality to add extra cuts during the branch and cut search in order to tighten the model formulation. Thus, the approach presented in this section is implemented as a supporting method for the branch and cut search of the optimizer. In Section 3.7, the performance of the model GPF is evaluated in the first place without the additional constraints, in the second place with all additional constraints, and in the third place with the dynamic cut-adding approach described above.

Algorithm 2 Dynamic Cut-Adding for Model IPF

for all $k = 1, \ldots, m$ **do**
  Create a graph $G = (V, E)$ with

$$V = \{i \in \tilde{O} \,|\, x_{i,k} = 1\},$$
$$E = \{(i,j) \,|\, i,j \in V \wedge y_{i,j,k} > 0\}.$$

  for all nodes i in V **do**
    Perform a depth first search starting at node i.
    if $\exists$ cycle C with $|C| < |V|$ in G **then**
      if $\sum_{i \in C, i' \notin C} y_{i,i',k} < 1$ **then**
        Add constraint

$$\sum_{i \in C, i' \notin C} y_{i,i',k} > 1.$$

      end if
    end if
  end for
end for

3.6.4 Additional Constraints for GPF

For the model GPF, additional valid inequalities are formulated, too. The general precedence formulation allows for an exploitation of the transitivity of the ordering. If operation i is scheduled before operation j on machine k, that is, $y_{i,j,k} = 1$, and operation j is scheduled before operation l on machine k, that is, $y_{j,l,k} = 1$, then operation i is scheduled before operation l as well, that is, $y_{i,l,k} = 1$. Consequently, the additional constraints

$$y_{i,j,k} + y_{j,l,k} \leq 1 + y_{i,l,k} \quad \text{for all } i \neq j \neq l \in I_k,\ k = 1, \ldots, m \tag{3.25}$$

are introduced. It remains to show that constraints (3.25) actually present valid inequalities tightening the formulation. Consider an example with $n = 6$, $n_i = 1$, $i = 1, \ldots, n$ and thus, each of the six jobs has exactly one operation and $N = 6$. Furthermore, $m = 2$ and $M_1 = \cdots = M_3 = \{\mu_1\}$ and $M_4 = \cdots = M_6 = \{\mu_2\}$. The processing times are given by $p_i = 1$, $i = 1, \ldots, 3$ and $p_i = 2$, $i = 4, \ldots, 6$. Consequently, the Big-M constant M is given by $M = 9$. The following assignment presents a feasible solution of

the linear relaxation of model GPF.

$$(y_{i,j,1}) = \begin{pmatrix} - & 1 & 0.5 \\ 0 & - & 1 \\ 0.5 & 0 & - \end{pmatrix} \qquad (y_{i,j,2}) = \begin{pmatrix} - & 1 & 0 \\ 0 & - & 1 \\ 1 & 0 & - \end{pmatrix}$$

$$(x_{i,k})^T = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

$$(S_i)^T = \begin{pmatrix} 0 & 1 & 2 & 0 & 2 & 4 \end{pmatrix}$$

The additional constraints (3.25) are violated, since

$$y_{1,2,1} + y_{2,3,1} = 2 > 1 + y_{1,3,1} = \frac{3}{2}.$$

Consequently, constraints (3.25) present valid inequalities tightening the formulation of model GPF. The number of additional constraints is given by

$$\sum_{k=1}^{m} I_k(I_k - 1)(I_k - 2) \in \mathcal{O}(N^3 m).$$

In order to avoid unnecessary additional constraints, a dynamical cut-adding method can also be applied in this case. If a feasible solution to the linear relaxation violates one of the constraints (3.25), this particular constraint is locally added in the subtree and removed by backtracking in the subtree. In Section 3.7, the performance of the model GPF is evaluated in the first place without the additional constraints, in the second place with all additional constraints added initially, and in the third place with the dynamical method, adding the constraints during the branch and cut search on demand.

3.6.5 Branching Strategy

For implementation the IBM ILOG CPLEX Optimizer is applied, which uses a branch and cut search. As described before, a branch and cut search is an algorithm searching the branch and bound tree of all possible solutions, branching on decision variables and cutting off those branches that do not lead toward a better solution than the one currently known. The most

important part about the branching strategy is to determine the order in which the binary variables should be selected for branching and subsequent fixation. The IBM ILOG CPLEX Optimizer provides the functionality to specify a branching priority for each variable. For the models IPF and GPF, a branching strategy for the machine assignment variables $x_{i,k}$ and the order variables $y_{i,j,k}$ is considered. It is proposed to branch first on the machine assignment variables $x_{i,k}$. The explanation for this decision lies in the dependence of the variables $y_{i,j,k}$ on the machine assignment variables $x_{i,k}$. Before the ordering of the operations on each machine is determined, it is beneficial to decide, which operations are actually assigned to each machine. Consequently, the branching priority for the machine assignment variables $x_{i,k}$ is increased in relation to the variables $y_{i,j,k}$. In Section 3.7 the performance of models IPF and GPF with respect to the branching strategy is compared.

3.7 Computational Results

The purpose of this section is to compare the performance of the models IPF, GPF, TEF, and MPF and to evaluate the impact of the improvements for the different models discussed in Sections 3.6.1, 3.6.2, 3.6.3, and 3.6.4. Furthermore, the influence of different parameters defining the flexible job shop scheduling problem is examined. For the practical implementation of the developed mixed integer programming formulations IBM ILOG CPLEX V12.1 is employed. The modeling was executed in IBM ILOG OPL V6.3 which includes the Optimization Programming Language (OPL) for developing optimization models. For the optimization, the IBM ILOG CPLEX Optimizer was used. The IBM ILOG CPLEX Optimizer is a mathematical programming solver for linear programming, mixed integer programming, and quadratic programming. Additionally, external functions, implemented in Java™ SE Runtime Environment (build 1.6.0_29-b11) are applied for the dynamic adding of cuts during the branch and cut search (cf. Section 3.6.4 and Section 3.6.3). All computations were performed on a PC with a Pentium Dual Core CPU E6500 2.93 Ghz, 1.96 GB RAM. The OPL source code of the implementation of the models IPF, GPF, TEF, and MPF, and the Java source code of the external functions can be found on the CD attached to this thesis.

The performance of the models is in the first place tested on randomly

generated instances $P_1, \ldots, P_{11}$ of the flexible job shop scheduling problem. Each problem is characterized by the number of jobs n, the number of operations N, the number of machines m, and the flexibility f. The flexibility is the probability that a machine is valid for an operation. The size of the instances varies from $n = 4$ jobs with $N = 15$ operations on $m = 4$ machines (P_1) to $n = 15$ jobs with $N = 48$ operations on $m = 7$ machines (P_{11}). For all instances $P_1, \ldots, P_{11}$, a flexibility f of 0.33 is chosen. Furthermore, for the generation of the test instances, the N operations are uniformly distributed among the n jobs and machine $k = 1, \ldots, m$ is valid for operation $i \in O$ with a probability of f. If no machine is valid for an operation after this process, one machine is chosen randomly. The processing times for the operations are chosen uniformly between 1 and 10. The choice of the size of these test instances is based on the experience obtained during extensive computational experiments and aims to cover instances ranging from easy to solve (P_1) to fairly hard to solve (P_{11}). The instances $P_1, \ldots, P_{11}$ and the generating script can be found on the CD attached to this thesis.

Obviously, the solvability of the instances is not only depending on the formulation of the model, but also on the utilized optimizer and hardware. Thus, it is difficult to compare the results presented in the following to results in the literature treating optimization models of the flexible job shop scheduling problem, for example [SMF06]. Still, it is mentioned that the size of the instances solved to optimality with the help of the described optimization models and optimization software exceeds, in all conscience, the size of instances solved with an optimization software presented in the literature. In the following the computational results for the models IPF, GPF, TEF, and MPF and the presented performance improvements are evaluated in detail. Table 3.2 on page 48 specifies the abbreviations and variables used in the tables listing the computational results and explains them in more detail.

3.7.1 Basic Models

First of all, the basic models IPF, GPF, TEF, and MPF are compared with the help of the instances $P_1, \ldots, P_{11}$. In Table 3.3 on page 49, the computational results are listed.

Model GPF outperforms the other models considerably with respect to the processing time for each instance and even for the largest instance P_{11},

Item	Description
Mod	model name
Per	applied performance improvements:
	B branching, see Section 3.6.5
	M improved Big-M constants, see Section 3.6.1
	T improved upper bound T, see Section 3.6.2
	AC additional constraints, see Sections 3.6.4 and 3.6.3
	DC dynamic cuts, see Sections 3.6.4 and 3.6.3
Pro	problem
n	number of jobs
N	number of operations
m	number of machines
f	flexibility
$\mathcal{V}$	number of variables
$\mathcal{C}$	number of constraints
(LB, UB)	lower and upper bound; if LB equals UB the optimality of the objective value is proven
$C_{\max}$	makespan, marked with * if it is equal to the optimal objective value
CPU	processing time in seconds; if no solution is found after 3600 seconds, the solution process is aborted manually and the best objective value is reported

Table 3.2: Structure of the computational results

Mod	Per	Pro	n	m	N	f	$\mathcal{V}$	$\mathcal{C}$	(LB, UB)	$C_{\max}$	CPU
IPF	-	P_1	4	4	15	0.36	199	216	(32, 32)	32*	0.09
GPF	-	P_1	4	4	15	0.36	147	246	(32, 32)	32*	0.03
TEF	-	P_1	4	4	15	0.36	2422	470	(32, 32)	32*	0.52
MPF	-	P_1	4	4	15	0.36	147	634	(32, 32)	32*	1.36
IPF	-	P_2	6	4	18	0.36	262	282	(35, 35)	35*	0.38
GPF	-	P_2	6	4	18	0.36	202	348	(35, 35)	35*	0.06
TEF	-	P_2	6	4	18	0.36	3122	516	(35, 35)	35*	1
MPF	-	P_2	6	4	18	0.36	202	1098	(35, 35)	35*	17.8
IPF	-	P_3	6	4	21	0.34	326	349	(38, 38)	38*	1.06
GPF	-	P_3	6	4	21	0.34	260	458	(38, 38)	38*	0.06
TEF	-	P_3	6	4	21	0.34	3946	586	(38, 38)	38*	1.91
MPF	-	P_3	6	4	21	0.34	260	1702	(33, 39)	39	3600
IPF	-	P_4	8	5	24	0.35	536	563	(24, 38)	38	3600
GPF	-	P_4	8	5	24	0.35	442	796	(34, 34)	34*	0.7
TEF	-	P_4	8	5	24	0.35	7100	893	(34, 34)	34*	23.5
MPF	-	P_4	8	5	24	0.35	442	3837	(23, 45)	45	3600
IPF	-	P_5	8	5	27	0.34	597	627	(39, 39)	39*	196
GPF	-	P_5	8	5	27	0.34	495	894	(39, 39)	39*	0.78
TEF	-	P_5	8	5	27	0.34	8972	1029	(39, 39)	39*	34.1
MPF	-	P_5	8	5	27	0.34	495	4401	(31, 54)	54	3600
IPF	-	P_6	10	5	30	0.34	717	750	(32, 40)	40	3600
GPF	-	P_6	10	5	30	0.34	605	1104	(35, 35)	35*	12.4
TEF	-	P_6	10	5	30	0.34	10763	1115	(35, 35)	35*	36.3
MPF	-	P_6	10	5	30	0.34	605	5953	(32, 52)	52	3600
IPF	-	P_7	10	6	33	0.31	859	896	(26, 37)	37	3600
GPF	-	P_7	10	6	33	0.31	723	1318	(34, 34)	34*	9.42
TEF	-	P_7	10	6	33	0.31	15006	1518	(34, 34)	34*	94.5
MPF	-	P_7	10	6	33	0.31	723	7098	(26, 40)	40	3600
IPF	-	P_8	12	6	36	0.31	975	1015	(31, 38)	38	3600
GPF	-	P_8	12	6	36	0.31	829	1520	(37, 37)	37*	715
TEF	-	P_8	12	6	36	0.31	17489	1638	(37, 37)	37*	1019
MPF	-	P_8	12	6	36	0.31	829	8442	(31, 50)	50	3600
IPF	-	P_9	12	6	39	0.32	1251	1294	(30, 42)	42*	3600
GPF	-	P_9	12	6	39	0.32	1087	2018	(42, 42)	42*	18.1
TEF	-	P_9	12	6	39	0.32	20446	1692	(42, 42)	42*	100
MPF	-	P_9	12	6	39	0.32	1087	13854	(30, 53)	53	3600
IPF	-	P_{10}	14	7	45	0.31	1682	1732	(30, 42)	42	3600
GPF	-	P_{10}	14	7	45	0.31	1470	2738	(36, 36)	36*	57.9
TEF	-	P_{10}	14	7	45	0.31	32771	2407	(35, 36)	36*	3600
MPF	-	P_{10}	14	7	45	0.31	1470	18315	(30, -)	-	3600
IPF	-	P_{11}	15	7	48	0.31	1871	1924	(32, 48)	48	3600
GPF	-	P_{11}	15	7	48	0.31	1647	3080	(37, 37)	37*	390
TEF	-	P_{11}	15	7	48	0.31	37487	2595	(37, 37)	37*	1968
MPF	-	P_{11}	15	7	48	0.31	1647	21771	(32, -)	-	3600

Table 3.3: Computational results for the basic models

the optimizer solves model GPF to optimality in 390 seconds. In Figure 3.3 on page 51, a machine-oriented Gantt chart visualizing the optimal solution for P_{11} is depicted. Operations belonging to the same job are colored equally.

In [SMF06], Saidi-Mehrabad et al. presented a mixed integer programming formulation for the flexible job shop scheduling problem and specified computational results for instances generated similarly as described above, obtained with the help of LINGO, a mathematical optimization software using a branch and bound method. Here, only instances up to a size of 3 jobs, 3 machines and 9 operations were solved to optimality within 3600 seconds. However, this comparison should be treated with caution, since the results do not only depend on the modeling, but also on the applied mathematical optimization software and hardware.

Model TEF is the only model besides model GPF for which the optimizer finds the optimal solution to each instance. The processing times for solving model TEF are consistently higher than for solving model GPF, but still below 3600 seconds except for instance P_{10}.

Furthermore, model TEF exhibits a comparatively high number of variables with respect to the other models. As discussed before, the number of variables for model TEF is dependent on the upper bound for the makespan T, see Table 3.1 on page 34, which is in the basic model defined by

$$T := \sum_{i \in O} \max_{k \in M_i} p_{i,k}.$$

Since $p_{i,k}$ is uniformly distributed in $[1, 10]$, T is a multiple of the number of operations N leading to a significantly higher number of variables. A more accurate reasoning for the difference in the number of variables can be established by evaluating the exact terms for the number of variables of the different models, which is left out here.

Models IPF and MPF cannot be solved to optimality by the optimizer in the given time limit for instances $P_4, \ldots, P_{11}$ and $P_3, \ldots, P_{11}$, respectively. Additionally, the number of constraints for model MPF is significantly high with respect to the other models, as expected by the analysis of the number of constraints given in Section 3.5 (see also Table 3.1 on page 34). Notably, the number of variables $\mathcal{V}(MPF)$ of model MPF equals the number of con-

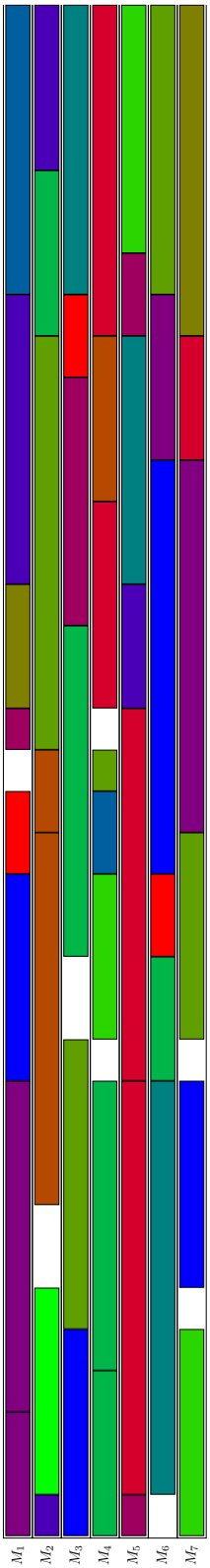


Figure 3.3: Machine-oriented Gantt chart for the optimal schedule for instance P_{11}

straints $\mathcal{V}(GPF)$ of model GPF, since

$$\sum_{k=1}^{m} |I_k| = \sum_{i \in O} |M_i| \quad \text{and}$$

$$\sum_{k=1}^{m} |I_k|^2 = \sum_{i \in O} \sum_{k \in M_i} |I_k|,$$

and therefore

$$\begin{aligned} \mathcal{V}(MPF) &= 1 + |O| + \sum_{i \in O} \sum_{k \in M_i} |I_k| \\ &= 1 + |O| + \sum_{k=1}^{m} |I_k| \left(|I_k| - 1\right) + \sum_{i \in O} |M_i| \\ &= \mathcal{V}(GPF). \end{aligned}$$

Finally, in order to give an overview of the performance of the different models, Figure 3.4 on page 53 depicts the processing times for the models IPF, GPF, TEF, and MPF for all instances in a semi-logarithmic plot. Here, it becomes evident that the processing time for solving the flexible job shop scheduling problem is approximatively exponential with respect to the model size, as expected by results from Section 2.3.

In the following sections, the performance improvements for the different models are added step-by-step in order to evaluate their impact.

3.7.2 Branching

In Table 3.4 on page 54, the computational results of the optimizer for the models IPF and GPF with improved branching strategy as described in Section 3.6.5 are listed. For the models TEF and MPF, a reasonable branching strategy is not available due to their structure.

In order to evaluate the impact of the branching strategy, Figure 3.5 on page 55 depicts the relative change in processing times for the models IPF and GPF due to the improved branching strategy. For a better perception the vertical scale is limited to 10. In fact, for problems P_5 and P_9 it takes 17.37 and 117.51 times longer to solve the model with improved branching strategy, respectively. Thus, for some instances the improved branching strategy accelerates the solving process, e.g. for instances P_2, P_4 (IPF) and P_6, P_8

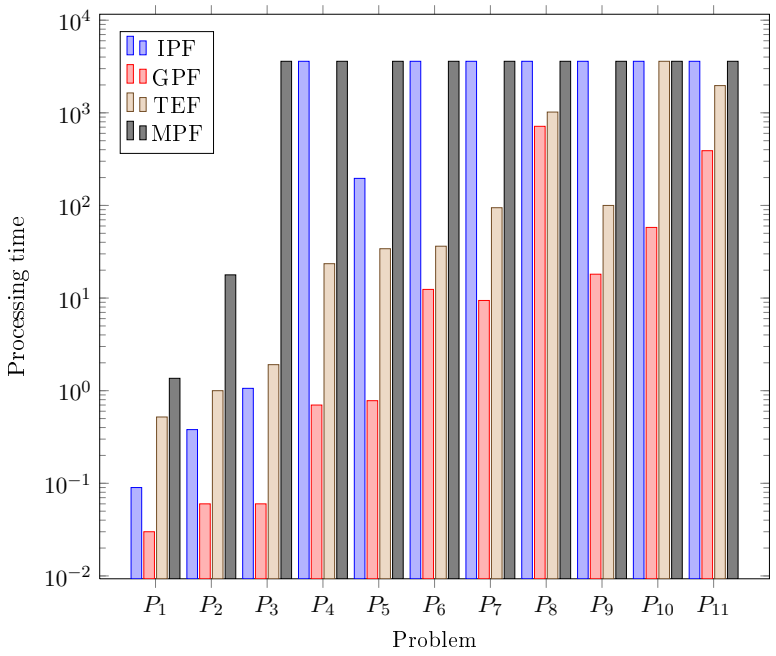


Figure 3.4: Processing times for the basic models

(GPF), whereas in other instances the model without the improved branching strategy are solved faster, e.g. for instances P_1, P_5 (IPF) and P_9, P_{11} (GPF). The solving technique of the utilized optimizer is not restricted to pure branch and cut search, but involves sophisticated cutting-plane strategies and feasibility heuristics [cpl10]. Thus, manually added problem-specific techniques can in certain circumstances either increase or decrease the overall performance, as seen in the results. Consequently, it is difficult to decide whether the branching strategy is actually an improvement for the models.

3.7.3 Upper Bounds and Individual Big-M Constants

In this section, the impact of the performance improvements presented in Sections 3.6.1 and 3.6.2 is evaluated. For the models IPF, GPF, and MPF, individual Big-M constants based on an upper bound for the makespan obtained by a list scheduling heuristics are implemented. The Big-M constant M chosen for the basic models and the average $\varnothing M_{i,j,k}$ of the individual Big-M constants obtained as described in Section 3.6.1, are listed in Table 3.5 on page 54.

Furthermore, the model TEF is now generated with a time period $[0, T_{\text{opt}}]$

Mod	Per	Pro	n	m	N	f	$\mathcal{V}$	$\mathcal{C}$	(LB, UB)	$C_{\max}$	CPU
IPF	B	P_1	4	4	15	0.36	199	216	(32, 32)	32*	0.13
GPF	B	P_1	4	4	15	0.36	147	246	(32, 32)	32*	0.09
IPF	B	P_2	6	4	18	0.36	262	282	(35, 35)	35*	0.27
GPF	B	P_2	6	4	18	0.36	202	348	(35, 35)	35*	0.11
IPF	B	P_3	6	4	21	0.34	326	349	(38, 38)	38*	1.34
GPF	B	P_3	6	4	21	0.34	260	458	(38, 38)	38*	0.13
IPF	B	P_4	8	5	24	0.35	536	563	(34, 34)	34*	388
GPF	B	P_4	8	5	24	0.35	442	796	(34, 34)	34*	2.36
IPF	B	P_5	8	5	27	0.34	597	627	(38, 39)	39	3600
GPF	B	P_5	8	5	27	0.34	495	894	(39, 39)	39*	0.66
IPF	B	P_6	10	5	30	0.34	717	750	(32, 38)	38	3600
GPF	B	P_6	10	5	30	0.34	605	1104	(35, 35)	35*	3.84
IPF	B	P_7	10	6	33	0.31	859	896	(26, 36)	36	3600
GPF	B	P_7	10	6	33	0.31	723	1318	(34, 34)	34*	14.4
IPF	B	P_8	12	6	36	0.31	975	1015	(31, 37)	37*	3529
GPF	B	P_8	12	6	36	0.31	829	1520	(37, 37)	37*	127
IPF	B	P_9	12	6	39	0.32	1251	1294	(30, 43)	43	3600
GPF	B	P_9	12	6	39	0.32	1087	2018	(42, 42)	42*	2145
IPF	B	P_{10}	14	7	45	0.31	1682	1732	(30, 41)	41	3600
GPF	B	P_{10}	14	7	45	0.31	1470	2738	(36, 36)	36*	98
IPF	B	P_{11}	15	7	48	0.31	1871	1924	(32, 43)	43	3600
GPF	B	P_{11}	15	7	48	0.31	1647	3080	(33, 38)	38	3600

Table 3.4: Computational results for the models IPF and GPF with improved branching strategy

Problem	M	$\varnothing M_{i,j,k}(IPF)$	$\varnothing M_{i,j,k}(GPF)$	$\varnothing M_{i,j,k}(MPF)$	T	T_{opt}
P_1	109	22.36	19.54	19.90	109	39
P_2	119	36.76	34.88	34.11	119	49
P_3	135	32.71	30.71	30.17	135	48
P_4	168	38.92	37.98	37.88	168	48
P_5	194	32.61	31.27	31.10	194	46
P_6	210	44.21	34.23	36.37	210	46
P_7	241	36.63	34.62	39.73	241	47
P_8	260	32.06	34.11	43.95	260	46
P_9	268	41.46	40.64	44.86	268	53
P_{10}	330	40.94	40.11	34.20	330	53
P_{11}	356	42.10	41.38	34.37	356	53

Table 3.5: Comparison of Big-M constants and upper bounds T

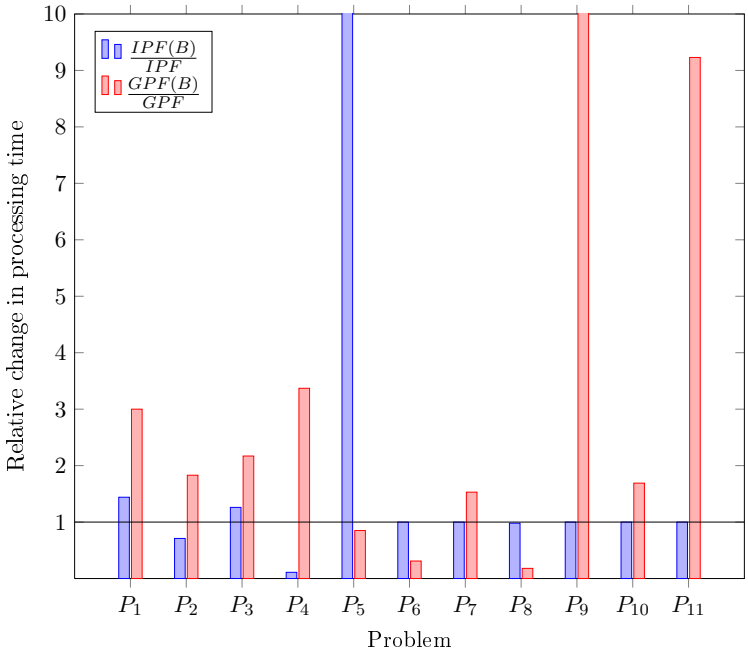


Figure 3.5: Relative change in processing time for the models IPF and GPF due to improved branching strategy

based on the improved upper bound. The values for T_{opt} are specified in Table 3.5 on page 54, too. For each instance $P_1, \ldots, P_{11}$ the average of the individual Big-M constants and the values for T_{opt} improve on the values chosen for the basic models, as depicted in Figure 3.6 on page 57. The computational results for the models IPF, GPF, TEF, and MPF with improved Big-M constants and improved upper bound T are specified in Table 3.6 on page 56.

Just the same as without the improved Big-M constants, Model MPF is not solved to optimality for the problems $P_4, \ldots, P_{11}$. In comparison to the basic version of the model MPF problem P_3 is solved to optimality in addition to P_1 and P_2 with the help of the improved Big-M constants, but for larger instances a feasible solution is not found. Since the decrease of the Big-M constants leads to a reduction of the solution space for the problem, it becomes harder to determine a feasible solution at all. In the following evaluations, model MPF is skipped, since no further improvements for the model are at hand and it is strictly dominated by the other models for all test instances.

Mod	Per	Pro	n	m	N	f	$\mathcal{V}$	$\mathcal{C}$	(LB, UB)	$C_{\max}$	CPU
IPF	M	P_1	4	4	15	0.36	199	216	(32, 32)	32*	0.08
GPF	M	P_1	4	4	15	0.36	147	246	(32, 32)	32*	0.02
TEF	T	P_1	4	4	15	0.36	732	190	(32, 32)	32*	0.19
MPF	M	P_1	4	4	15	0.36	147	961	(32, 32)	32*	0.38
IPF	M	P_2	6	4	18	0.36	262	282	(35, 35)	35*	1.02
GPF	M	P_2	6	4	18	0.36	202	348	(35, 35)	35*	0.14
TEF	T	P_2	6	4	18	0.36	1135	236	(35, 35)	35*	0.5
MPF	M	P_2	6	4	18	0.36	202	1206	(35, 35)	35*	29.7
IPF	M	P_3	6	4	21	0.34	326	349	(38, 38)	38*	1.41
GPF	M	P_3	6	4	21	0.34	260	458	(38, 38)	38*	0.09
TEF	T	P_3	6	4	21	0.34	1239	238	(38, 38)	38*	2.52
MPF	M	P_3	6	4	21	0.34	260	1837	(38, 38)	38*	620
IPF	M	P_4	8	5	24	0.35	536	563	(34, 34)	34*	606
GPF	M	P_4	8	5	24	0.35	442	796	(34, 34)	34*	0.89
TEF	T	P_4	8	5	24	0.35	1806	293	(34, 34)	34*	4.52
MPF	M	P_4	8	5	24	0.35	442	3837	(23, -)	-	3600
IPF	M	P_5	8	5	27	0.34	597	627	(39, 39)	39*	317
GPF	M	P_5	8	5	27	0.34	495	894	(39, 39)	39*	0.7
TEF	T	P_5	8	5	27	0.34	1873	289	(39, 39)	39*	8.77
MPF	M	P_5	8	5	27	0.34	495	4511	(31, -)	-	3600
IPF	M	P_6	10	5	30	0.34	717	750	(32, 38)	38	3600
GPF	M	P_6	10	5	30	0.34	605	1104	(35, 35)	35*	5.69
TEF	T	P_6	10	5	30	0.34	2083	295	(35, 35)	35*	6.72
MPF	M	P_6	10	5	30	0.34	605	6058	(32, -)	-	3600
IPF	M	P_7	10	6	33	0.31	859	896	(29, 36)	36	3600
GPF	M	P_7	10	6	33	0.31	723	1318	(34, 34)	34*	3.8
TEF	T	P_7	10	6	33	0.31	2594	354	(34, 34)	34*	34.7
MPF	M	P_7	10	6	33	0.31	723	7188	(26, -)	-	3600
IPF	M	P_8	12	6	36	0.31	975	1015	(31, 38)	38	3600
GPF	M	P_8	12	6	36	0.31	829	1520	(37, 37)	37*	141
TEF	T	P_8	12	6	36	0.31	2326	318	(37, 37)	37*	1787
MPF	M	P_8	12	6	36	0.31	829	9572	(31, -)	-	3600
IPF	M	P_9	12	6	39	0.32	1251	1294	(30, -)	-	3600
GPF	M	P_9	12	6	39	0.32	1087	2018	(42, 42)	42*	28.64
TEF	T	P_9	12	6	39	0.32	3267	372	(42, 42)	42*	35.1
MPF	M	P_9	12	6	39	0.32	1087	14641	(30, -)	-	3600
IPF	M	P_{10}	14	7	45	0.31	1682	1732	(30, -)	-	3600
GPF	M	P_{10}	14	7	45	0.31	1470	2738	(36, 36)	36*	132
TEF	T	P_{10}	14	7	45	0.31	3950	412	(35, 36)	36*	3600
MPF	M	P_{10}	14	7	45	0.31	1470	18381	(30, -)	-	3600
IPF	M	P_{11}	15	7	48	0.31	1871	1924	(32, -)	-	3600
GPF	M	P_{11}	15	7	48	0.31	1647	3080	(37, 37)	37*	129
TEF	T	P_{11}	15	7	48	0.31	4400	432	(37, 38)	38	3600
MPF	M	P_{11}	15	7	48	0.31	1647	22606	(32, -)	-	3600

Table 3.6: Computational results for the models IPF, GPF, TEF, and MPF with improved Big-M constants and improved upper bound T

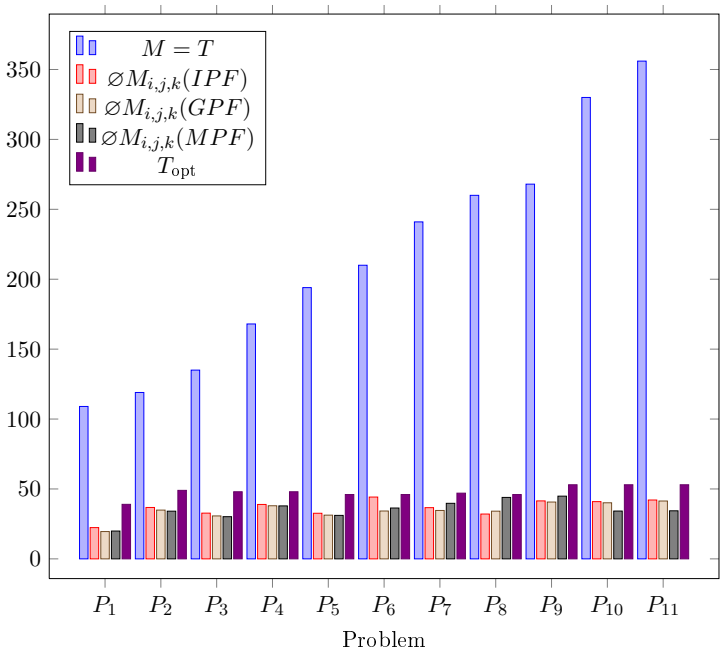


Figure 3.6: Improvements for Big-M constants and upper bound T

The relative change in processing time for the models IPF, GPF, and TEF due to the improved Big-M constants and the improved upper bound T is visualized in Figure 3.7 on page 59.

For model TEF, the impact of the improved upper bound G is evidenced by a significant decrease in the number of variables, see Table 3.6 on page 56. This reduction in the model size is reflected in a faster processing time for problems P_4, P_5, P_6, P_7, and P_9. Still, for some problems the processing time increases for model TEF with the improved upper bound T. For the models GPF and IPF a similar behavior is observed, which can be explained by the same reasoning as for model MPF above. Especially for the larger problems $P_6, \ldots, P_{11}$ there is no impact of the improved Big-M constants detectable for model IPF.

3.7.4 Additional Constraints and Dynamic Cuts for IPF

In this section the additional constraints and dynamic cuts developed in Section 3.6.3 are applied to model IPF and the impact of the improvements is evaluated. In Table 3.7 on page 58, the computational results for problems $P_1, \ldots, P_{11}$ are listed.

Mod	Per	Pro	n	m	N	f	$\mathcal{V}$	$\mathcal{C}$	(LB, UB)	$C_{\max}$	CPU
IPF	AC	P_1	4	4	15	0.37	719	8096	(32, 32)	32*	0.59
IPF	DC	P_1	4	4	15	0.37	199	216	(32, 32)	32*	1.17
IPF	AC	P_2	6	4	18	0.36	1694	32706	(35, 35)	35*	12.1
IPF	DC	P_2	6	4	18	0.36	262	282	(35, 35)	35*	32.6
IPF	AC	P_3	6	4	21	0.35	3406	84389	(38, 38)	38*	116
IPF	DC	P_3	6	4	21	0.35	326	349	(38, 38)	38*	21.8
IPF	AC	P_4	8	5	24	0.35	22174	970549	(23, 48)	48	3600
IPF	DC	P_4	8	5	24	0.35	536	563	(25, 35)	35	3600
IPF	AC	P_5	8	5	27	0.34					
IPF	DC	P_5	8	5	27	0.34	597	627	(34, 39)	39*	3600
IPF	AC	P_6	10	5	30	0.34					
IPF	DC	P_6	10	5	30	0.34	717	750	(32, 52)	52	3600
IPF	AC	P_7	10	6	33	0.31					
IPF	DC	P_7	10	6	33	0.31	859	896	(26, 52)	52	3600
IPF	AC	P_8	12	6	36	0.31					
IPF	DC	P_8	12	6	36	0.31	975	1015	(31, 50)	50	3600
IPF	AC	P_9	12	6	39	0.32					
IPF	DC	P_9	12	6	39	0.32	1251	1294	(30, 55)	55	3600
IPF	AC	P_{10}	14	7	45	0.31					
IPF	DC	P_{10}	14	7	45	0.31	1682	1732	(30, 47)	47	3600
IPF	AC	P_{11}	15	7	48	0.31					
IPF	DC	P_{11}	15	7	48	0.31	1871	1924	(32, 58)	58	3600

Table 3.7: Computational results for the model IPF with additional constraints (AC) and dynamic cuts (DC)

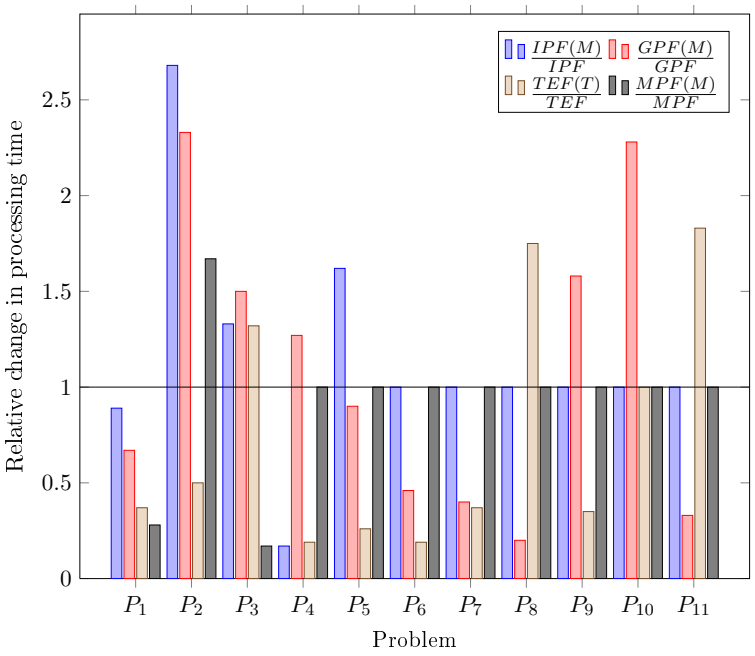


Figure 3.7: Relative change in processing time for the models IPF, GPF, MPF and TEF due to the improved Big-M constants and the improved upper bound T, respectively

As already mentioned in Section 3.6.3, the number of additional constraints is growing exponentially with respect to the number of operations, see $P_1, \ldots, P_4$ in the corresponding lines in Table 3.7 on page 58. Furthermore, model IPF cannot be generated with additional constraints for problems $P_5, \ldots, P_{11}$ since the optimizer runs out of memory. Therefore, the respective lines are empty in Table 3.7 on page 58. But also the enhanced approach to introduce the constraints dynamically does neither lead to an improvement of processing times, nor to better upper and lower bounds for the objective value, compare Table 3.3 on page 49 and Table 3.7 on page 58. The effort to identify the additional constraints and add them to the model exceeds the benefits and consequently, this approach is not adaptable in practice.

3.7.5 Additional Constraints and Dynamic Cuts for GPF

In this section, the additional constraints and dynamic cuts developed in Section 3.6.4 are applied to model GPF and the impact of the improvements

Mod	Per	Pro	n	m	N	f	$\mathcal{V}$	$\mathcal{C}$	(LB, UB)	$C_{\max}$	CPU
GPF	AC	P_1	4	4	15	0.36	147	702	(32, 32)	32*	0.11
GPF	DC	P_1	4	4	15	0.36	147	246	(32, 32)	32*	0.16
GPF	AC	P_2	6	4	18	0.36	202	1206	(35, 35)	35*	0.19
GPF	DC	P_2	6	4	18	0.36	202	348	(35, 35)	35*	4.61
GPF	AC	P_3	6	4	21	0.34	260	1856	(38, 38)	38*	0.22
GPF	DC	P_3	6	4	21	0.34	260	458	(38, 38)	38*	32.6
GPF	AC	P_4	8	5	24	0.35	442	4132	(34, 34)	34*	4.06
GPF	DC	P_4	8	5	24	0.35	442	796	(34, 34)	34*	544
GPF	AC	P_5	8	5	27	0.34	495	4734	(39, 39)	39*	5.92
GPF	DC	P_5	8	5	27	0.34	495	894	(39, 39)	39*	294
GPF	AC	P_6	10	5	30	0.34	605	6378	(35, 35)	35*	29.5
GPF	DC	P_6	10	5	30	0.34	605	1104	(34, 38)	38	3602
GPF	AC	P_7	10	6	33	0.31	723	7606	(34, 34)	34*	74.8
GPF	DC	P_7	10	6	33	0.31	723	1318	(30, 36)	36	3601
GPF	AC	P_8	12	6	36	0.31	829	9038	(37, 37)	37*	2241
GPF	DC	P_8	12	6	36	0.31	829	1520	(34, 38)	38	3541
GPF	AC	P_9	12	6	39	0.32	1087	14678	(42, 42)	42*	660
GPF	DC	P_9	12	6	39	0.32	1087	2018	(30, -)	-	3601
GPF	AC	P_{10}	14	7	45	0.31	1470	19448	(36, 36)	36*	1547
GPF	DC	P_{10}	14	7	45	0.31	1470	2738	(30, -)	-	3600
GPF	AC	P_{11}	15	7	48	0.31	1647	23060	(34, 38)	38	3600
GPF	DC	P_{11}	15	7	48	0.31	1647	3080	(32, -)	-	3605

Table 3.8: Computational results for the model GPF with additional constraints (AC) and dynamic cuts (DC)

is evaluated. In Table 3.8 on page 60, the computational results for problems $P_1, \ldots, P_{11}$ are listed.

Once again, the additional constraints tighten the formulation on the one hand, but on the other hand the model size increases due to the additional constraints, compare Table 3.3 on page 49 and Table 3.8 on page 60. The empirical results lead to the conclusion that the performance loss due to an increased model size outweighs the benefits of a tightened formulation, as depicted in Figure 3.8 on page 61. Here, the relative change in processing time is plotted up to a multiple of ten. For the relative change in processing time exceeding a multiple of ten it is referred to Table 3.8 on page 60. Furthermore, the enhanced approach to introduce the constraints dynamically leads to an increase in processing times, see Figure 3.8 on page 61. Analogously to Section 3.7.4, the effort to identify the additional constraints and

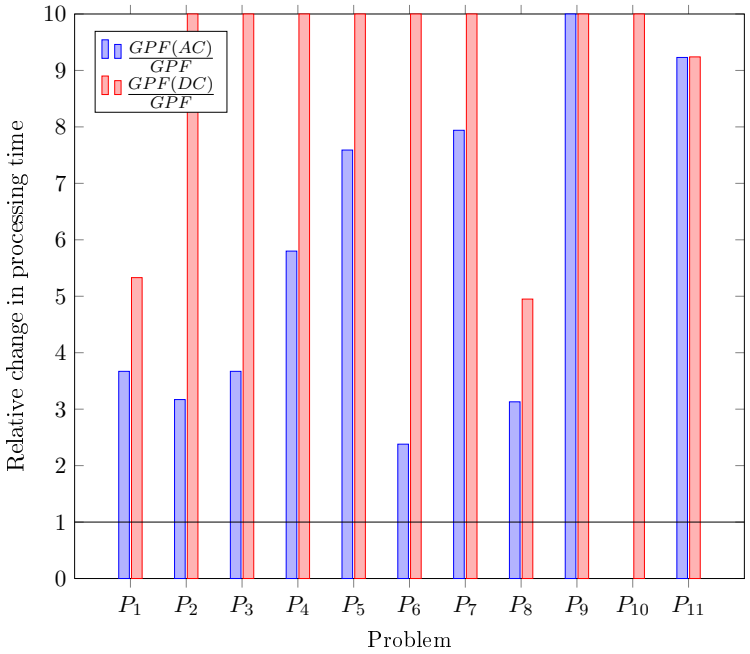


Figure 3.8: Relative change in processing time for the model GPF due to additional constraints (AC) and dynamic cuts (DC)

add them to the model exceeds the benefits and consequently, this approach is not adaptable in practice.

3.7.6 Influence of Problem Parameters

In order to achieve a better understanding of the difficulties imposed by the flexible job shop problem, the influence of different problem parameters on the performance of the models GPF and TEF is empirically evaluated in this section. Models GPF and TEF are chosen due to their superior performance with respect to the other models and their basically varying model structure. Model GPF is applied with improved Big-M constants, see Section 3.6.1 and model TEF is applied with improved upper bound T, see Section 3.6.2.

First of all, the influence of the number of operations per job on the performance of the models is discussed. For each $n = 1, \ldots, 20$, ten random instances with n jobs, $N = 20$ operations, $m = 5$ machines, and flexibility $f = 0.3$ are generated as described above. Consequently, the range of operations per job varies from one job with 20 operations to 20 jobs with one operation each. Due to the small size of the instances, both models can be

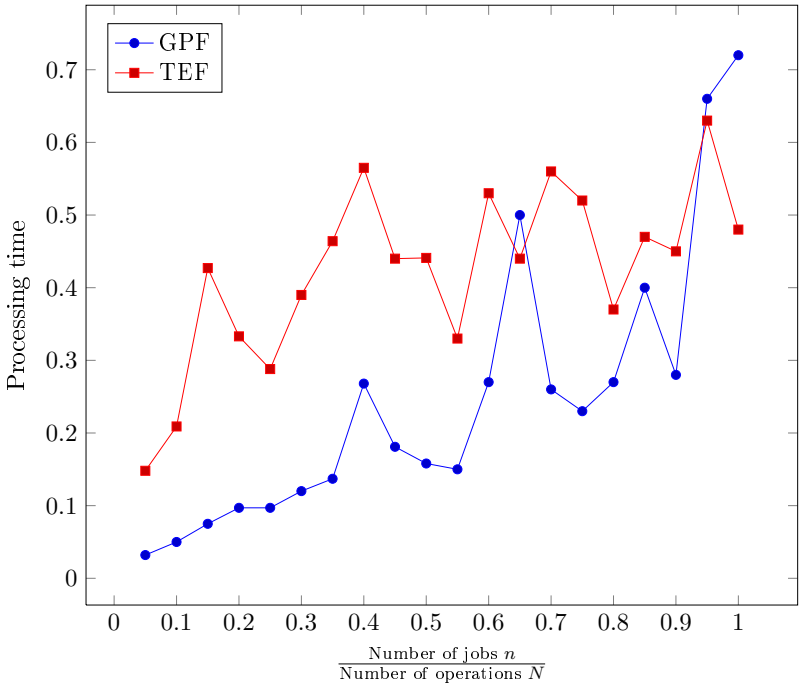


Figure 3.9: Influence of the number of operations per job on the processing time

solved to optimality in a reasonable amount of time. For both models, the average processing time for the ten instances for each value of n is plotted in Figure 3.9 on page 62. The behavior with respect to an increasing number of operations per job is the same for both models. The less operations per job, the higher is the average processing time. It is easier to solve an instance with few jobs consisting of many operations, since most of the ordering of operations is already determined by the sequence of operations of each job and basically the machine with the lowest processing time for the operation is chosen.

Up next, the influence of the number of machines on the performance of the models is discussed. For each $m = 1, \ldots, 10$, ten random instances with $n = 5$ jobs, $N = 10$ operations, m machines, and flexibility $f = 0.3$ are generated as described in Section 3.7. For both models, the average processing time for the ten instances for each value of m is plotted in Figure 3.10 on page 63. For $m = 2, \ldots, 10$ both models perform similar and the influence of the number of machines on the processing time is negligible for such small instances. However, for $m = 1$ the processing time of model GPF

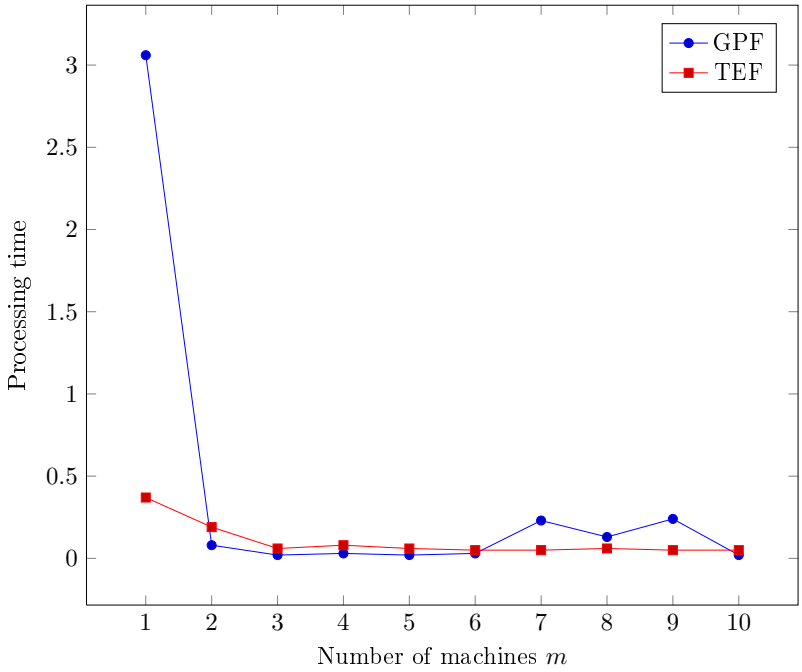


Figure 3.10: Influence of the number of machines on the processing time

is significantly higher than all other processing times. By intuition, the case of one machine is the most simple one and is solved by scheduling all operations consecutively on the single machine. For model GPF, the task is significantly more complex. In the case of one machine, an ordering of all operations has to be established. Due to the modeling of the disjunctive constraints, this results basically in a traveling salesman problem complicated by Big-M constants. The time expanded formulation of model TEF avoids this complication. Still, it must be mentioned that this only serves as an interpretation of the observed behavior.

Last but not least, the influence of the flexibility on the processing time of both models is investigated. For each $f = 0.1, 0.2, \ldots, 1$, ten random instances with $n = 5$ jobs, $N = 25$ operations, $m = 10$ machines, and flexibility f are generated as described in Section 3.7. Thus, for instances generated with $f = 0.1$, each operation is executable on few machines, whereas for instances generated with $f = 1$, each operation is executable on all machines. For both models, the average processing time for the ten instances for each value of f is plotted in Figure 3.11 on page 64. The performances of the models GPF and TEF are contrary to each other. For model GPF, the

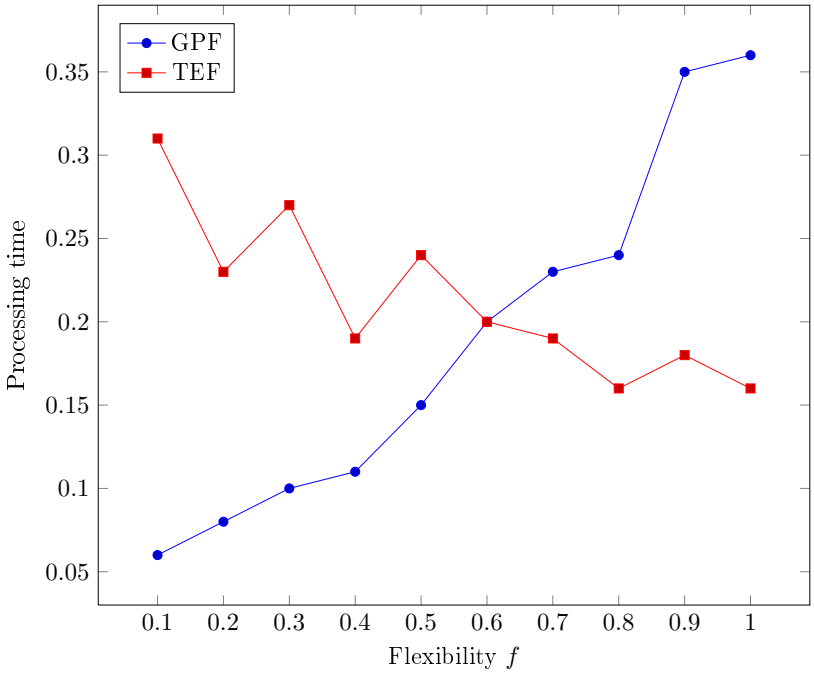


Figure 3.11: Influence of the flexibility on the processing time

processing time is increasing for an increasing flexibility and for model TEF the processing time is decreasing for an increasing flexibility. The behavior of model GPF is again attributed to the modeling of the disjunctive constraints. For a higher flexibility, each machine is valid for a higher number of operations. Consequently, the complexity of ordering operations on each machine, implicitly modeled as a traveling salesman problem, is increasing and dominates the complexity of the model.

Due to the modeling structure, model TEF is not subject to this complication, presenting even a decrease in processing time for a higher flexibility. As described in Section 3.6.2, the size of model TEF is dependent on the upper bound T. Since model TEF is applied with an improved upper bound T for the computational studies in this section, the upper bound T is determined by the list scheduling algorithm presented in Section 3.6.2. For the randomly generated instances, the makespan decreases with an increasing flexibility with a high probability. For $f = 0.1, 0.2, \ldots, 1$, the average optimal makespan of the randomly generated instances decreases constantly from 48.4 for $f = 0.1$ to 11.5 for $f = 1$. Consequently, the makespan of the list scheduling algorithm producing the upper bound T is likely to be smaller

for an increasing flexibility, leading to a decreasing model size and a faster processing time.

In conclusion, the computational studies evaluated in this section provided a better understanding of the models GPF and TEF and pointed out the main differences between them.

3.7.7 Summary of the Computational Results

In this section, the most important findings with respect to the modeling of the flexible job shop scheduling problem obtained by means of the computational studies presented in the preceding sections are briefly summarized.

1. Obviously, the way of modeling the flexible job shop scheduling problem is actually crucial for the performance of the model. In particular, the complexity of the flexible job shop scheduling problem is for the most part attributed to the conjunctive / non-overlapping constraints. Thus, the modeling of these constraints is one of the decisive factors for the performance of the model.

2. Overall, model GPF outperforms the other models. Model TEF performs almost as well as model GPF, in some cases model TEF outperforms model GPF. In general, the ratio of the number of operations N and the upper bound for the makespan T is decisive for the ratio of the performances of the models GPF and TEF. For a high number of operations and a low upper bound for the makespan model TEF performs relatively better than model GPF and vice versa.

3. The models IPF and GPF are strictly dominated by the models GPF and TEF.

4. Instances up to a size of 15 jobs with 48 operations on 7 machines are solved to optimality in a reasonable amount of time. In all conscious, this exceeds the size of instances solved with an optimization software presented in the literature considerably.

5. An improved branching strategy, individual (smaller) Big-M constants, and a lower upper bound T potentially increase the performance of the models. However, in some cases these advancements have no effect or corrupt the performance of the models. In order to give more explicit

statements concerning these advancements more extensive computational studies are required, which goes beyond the scope of this thesis.

6. The additional valid inequalities tightening the formulation of the models IPF and GPF are theoretically an improvement for the models. However, the empirical results lead to the conclusion that the performance loss due to an increased model size (additional valid inequalities added initially to the model) and due to the effort to identify the additional constraints and add them to the model (dynamic cut-adding), respectively, outweighs the benefits of a tightened formulation.

Chapter 4

Approximation Algorithms

The flexible job shop scheduling problem is an $\mathcal{NP}$-hard problem and thus, unless $\mathcal{P} = \mathcal{NP}$, there are no efficient algorithms to find optimal solutions to this problem. Consequently, this chapter deals with approximation algorithms solving the flexible job shop scheduling problem approximately in efficient time. Formally, an approximation algorithm is defined as follows.

Definition 4.1. *Let $OPT(I)$ be the value of an optimal solution to instance I of the optimization problem $\mathcal{P}$. A k-factor approximation algorithm for $\mathcal{P}$ is a polynomial-time algorithm A for $\mathcal{P}$ such that*

$$\frac{1}{k} OPT(I) \leq A(I) \leq k OPT(I)$$

for all instances I of $\mathcal{P}$ [KV08]. It is said that A has performance-ratio k.

First of all, a performance guarantee for the flexible job shop scheduling problem based on the best known approximation algorithms for the job shop scheduling problem and the scheduling problem with restricted assignments is given in Section 4.1. Subsequently, LP-based heuristics are covered in Section 4.2.

LP-based heuristics are algorithms relying on solving the linear relaxation and exploiting the available information enclosed in the solution. On the one hand, LP-based heuristics utilizing the convex hull of feasible solution vectors are considered in Section 4.2.1 and Section 4.2.2. This approach allows for the derivation of further performance-ratios. On the other hand, LP-based heuristics applying a modified version of the time-expanded formulation of the flexible job shop scheduling problem are treated in Section

4.2.3. The latter approach leads to an efficient algorithm for the flexible job shop scheduling problem with a practicable empirical performance-ratio.

4.1 A Performance Guarantee for $FJ|p_{i,k} = p_i|C_{\max}$

In this section, a performance guarantee for the flexible job shop scheduling with identical processing times is presented. The reasoning is based on approximation algorithms for the job shop scheduling problem and the scheduling problem with restricted assignments.

Theorem 4.1. *For $FJ\,|\,p_{i,k} = p_i\,|\,C_{\max}$ let $\mathrm{val}(OPT)$ be the value of an optimal schedule. Then there exists a polynomial-time algorithm that delivers a schedule for $FJ\,|\,p_{i,k} = p_i\,|\,C_{\max}$ of makespan*

$$\mathcal{O}\left(\rho\left(3 - \frac{1}{p_{\max}}\right)\right)\mathrm{val}(OPT),$$

where

$$\rho = \frac{\log(m\mu)}{\log\log(m\mu)}\left\lceil\frac{\log(\min\{m\mu, p_{\max}\})}{\log\log(m\mu)}\right\rceil.$$

Proof. Let $\mathcal{X}$ be the set of all possible machine assignments and let $\Pi_{\max}(x)$ denote the maximum machine load corresponding to the machine assignment $x \in \mathcal{X}$. $\Pi_{\max}(x)$ is given by

$$\Pi_{\max}(x) = \max_{k=1,\ldots,m}\sum_{i=1}^{N} x_{i,k}p_i.$$

Furthermore, define

$$\Pi_{\max}^{\mathrm{OPT}} := \min_{x\in\mathcal{X}}\Pi_{\max}(x)$$

as the minimal maximum machine load. In addition, let $P_{\max}$ be the maximum job length given by

$$P_{\max} = \max_{j=1,\ldots,n}\sum_{\substack{i=1,\ldots,N:\\ J(i)=j}} p_i.$$

Both $\Pi_{\max}^{\mathrm{OPT}}$ and $P_{\max}$ present lower bounds for the makespan of an optimal schedule. Thus,

$$\max\left\{\Pi_{\max}^{\mathrm{OPT}},\, P_{\max}\right\} \leq \mathrm{val}(\mathrm{OPT}). \tag{4.1}$$

The determination of a machine assignment $x^* \in \mathcal{X}$ such that

$$\Pi_{max}(x^*) = \Pi_{\max}^{\mathrm{OPT}}$$

is equivalent to minimizing the makespan of N jobs on m parallel machines, where each job may be assigned to a subset of the m machines. It is noted, that in this case a job does not consist of a sequence of operations and consequently, no precedence constraints exist. This problem is also denoted as scheduling problem with restricted assignments. Glass et al. proved in [GK06] that this problem is $\mathcal{NP}$-hard and Gairing et al. presented in [GLMM04] a polynomial-time approximation algorithm yielding a performance-ratio ρ with

$$\rho = 2 - \frac{1}{p_{\max}},$$

where $p_{\max}$ is the maximum processing time given by

$$p_{\max} = \max_{i=1,\ldots,N} p_i.$$

In all conscience, this is currently the best approximation algorithm for the scheduling problem with restricted assignments. Let $\tilde{x}$ be the assignment produced by this algorithm. Consequently,

$$\Pi_{\max}(\tilde{x}) \leq \left(2 - \frac{1}{p_{\max}}\right) \Pi_{\max}^{\mathrm{OPT}}. \tag{4.2}$$

By means of a given machine assignment $\tilde{x}$ the flexible job shop scheduling problem is reduced to a job shop scheduling problem with maximum machine load $\Pi_{\max}(\tilde{x})$. Let

$$\mu = \max_{j=1,\ldots,n} n_j$$

denote the maximum number of operations per job. In [GPSS01] Goldberg et al. presented a polynomial-time algorithm for the job shop scheduling problem delivering schedules of makespan $\mathcal{O}\left(\left(P_{\max} + \Pi_{\max}\right)\rho\right)$, with

$$\rho = \frac{\log(m\mu)}{\log\log(m\mu)} \left\lceil \frac{\log(\min\{m\mu, p_{\max}\})}{\log\log(m\mu)} \right\rceil$$

and $\Pi_{\max}$ denotes the maximum machine load of the job shop scheduling problem. Let val(ALG) be the value of the schedule produced by this algo-

rithm applied to the job shop scheduling problem resulting from the flexible job shop scheduling problem and the machine assignment $\tilde{x}$. Thus,

$$\begin{aligned}
\mathrm{val}(\mathrm{ALG}) &\leq \mathcal{O}\left(\left(P_{\max} + \Pi_{\max}(\tilde{x})\right)\rho\right) \\
&\overset{(4.2)}{\leq} \mathcal{O}\left(\left(P_{\max} + \left(2 - \frac{1}{p_{\max}}\right)\Pi_{\max}^{\mathrm{OPT}}\right)\rho\right) \\
&\overset{(4.1)}{\leq} \mathcal{O}\left(\left(\mathrm{val}(\mathrm{OPT}) + \left(2 - \frac{1}{p_{\max}}\right)\mathrm{val}(\mathrm{OPT})\right)\rho\right) \\
&= \mathcal{O}\left(\rho\left(3 - \frac{1}{p_{\max}}\right)\right)\mathrm{val}(\mathrm{OPT}).
\end{aligned}$$

□

Consequently, there exists an approximation algorithm with an identical performance guarantee as for the job shop scheduling problem up to a constant factor of $\left(3 - \frac{1}{p_{\max}}\right)$. A similar result is presented by Shmoys et al. in [SSW94] for scheduling on unrelated parallel machines with chain precedence constraints, but with a higher constant factor of 6.

4.2 LP-Based Heuristics

In this section, LP-based heuristics, i.e., algorithms using the optimal solution to a linear programming relaxation of the mixed integer programming formulation of the flexible job shop scheduling problem, are presented. To start with, LP-based heuristics for the problems $FJ \,|\, n_i = 1, p_{i,k} = p_i \,|\, C_{\max}$ and $FJ \,|\, p_{i,k} = p_i \,|\, C_{\max}$ are developed in Sections 4.2.1 and 4.2.2 respectively, and performance ratios are derived. In Section 4.2.3, another implementable and efficient LP-based heuristic based on a modified version of the model TEF for the flexible job shop scheduling problem is developed.

4.2.1 $FJ \,|\, n_i = 1, p_{i,k} = p_i \,|\, C_{\max}$

In the first instance, the problem $FJ \,|\, n_i = 1, p_{i,k} = p_i \,|\, C_{\max}$ is considered. Each job consists of exactly one operation, i.e. there are no precedence constraints and $p_{i,k} = p_{i,l}$ for all machines $k, l \in M_i$. In [Sch96], Schulz presented LP-based approximation algorithms for several scheduling problems including the problems of minimizing the total weighted completion time on

a single-machine and on identical parallel machines. The approximation algorithm for the problem $FJ \,|\, n_i = 1, p_{i,k} = p_i \,|\, C_{\max}$ developed in this section is based on these investigations.

First of all, the problem of minimizing the makespan on one machine is regarded. We are given a set $J = \{1, \ldots, n\}$ of n jobs to be processed on a single machine which can execute at most one job at a time. Each job j must be processed for $p_j > 0$ time units without preemption. The objective function is the makespan $C_{\max}$. According to the classification scheme introduced in Section 2.2 this problem is denoted by $1 \,||\, C_{\max}$.

In [Sch96], it is proven that the convex hull of feasible completion time vectors for the problem of minimizing the weighted completion time on one machine is completely described by the following linear inequalities:

$$\sum_{i \in S} p_i C_i \geq \frac{1}{2} \left(\left(\sum_{i \in S} p_i \right)^2 + \sum_{i \in S} p_i^2 \right) \quad \text{for all } S \subseteq J. \tag{4.3}$$

Obviously, the convex hull of feasible completion time vectors for the problem of minimizing the makespan $C_{\max}$ on one machine is also completely described by the inequalities (4.3). In [Sch96], this approach is extended to the model of identical parallel machines. Here, m identical parallel machines are given instead of a single machine, each job must be processed by one of these machines, and may be assigned to any of these machines.

Lemma 4.2. *The completion time vector C of every feasible schedule on m identical parallel machines satisfies*

$$\sum_{i \in S} p_i C_i \geq \frac{1}{2m} \left(\sum_{i \in S} p_i \right)^2 + \frac{1}{2} \sum_{i \in S} p_i^2 \quad \textit{for all } S \subseteq J. \tag{4.4}$$

In order to extend the approach described above even further to the problem $FJ \,|\, n_i = 1, p_{i,k} = p_i \,|\, C_{\max}$, it has to be shown that the identical parallel machine problem $P \,||\, C_{\max}$ is a specialization of the problem $FJ \,|\, n_i = 1, p_{i,k} = p_i \,|\, C_{\max}$. According to the notation of the flexible job shop scheduling problem in Section 2.2, the identical parallel machine problem can be written as follows. We are given a shop environment with a set $M = \{\mu_1, \ldots, \mu_m\}$ of m machines and have to process n jobs $J_1, \ldots, J_n$. Job J_i consists of $n_i = 1$ operation $O_{i,1}$, which can be processed by a set of ma-

chines $M_{i,1} = \{\mu_1, \ldots, \mu_m\} = M$. As before, it is convenient to identify the operations $O_{i,j}$ by numbers $1, \ldots, N$, where $N = \sum_{i=1}^{n} n_i = n$, and consider the set of operations $O = \{1, \ldots, N\}$.

Since the identical parallel machine problem is a specialization of the problem $FJ \,|\, n_i = 1, p_{i,k} = p_i \,|\, C_{\max}$, Lemma 4.2 holds for every feasible schedule of the problem $FJ \,|\, n_i = 1, p_{i,k} = p_i \,|\, C_{\max}$ and a linear relaxation of the problem $FJ \,|\, n_i = 1, p_{i,k} = p_i \,|\, C_{\max}$ is given by

$$\begin{aligned} \min \quad & C_{\max} \\ \text{s.t.} \quad & \sum_{i \in S} p_i C_i \geq \frac{1}{2m} \left(\sum_{i \in S} p_i \right)^2 + \frac{1}{2} \sum_{i \in S} p_i^2 \qquad \text{for all } S \subseteq O \\ & C_i \geq p_i \qquad\qquad\qquad\qquad\qquad\qquad\quad \text{for all } i \in O. \end{aligned} \tag{4.5}$$

In [Sch96], Schulz proves that the separation problem associated with inequalities (4.4) can be solved in polynomial time. Thus, it follows from the equivalence of optimization and separation with respect to polynomial time solvability (cf. [GLS88]), that the linear relaxation (4.5) can be solved in polynomial time.

The linear relaxation (4.5) of the problem $FJ \,|\, n_i = 1, p_{i,k} = p_i \,|\, C_{\max}$ is the starting point for our approximation algorithm, formally summarized in Algorithm 3.

Algorithm 3 $FJ \,|\, n_i = 1, p_{i,k} = p_i \,|\, C_{\max}$

1: Find an optimal solution C^{LP} to the linear relaxation (4.5).
2: Index the operations such that $C_1^{LP} \leq C_2^{LP} \leq \cdots \leq C_N^{LP}$.
3: **for all** $i = 1, \ldots, N$ **do**
4: Schedule operation i on the machine $k \in M_i$ on which it can start as early as possible. If more than one machine meets this condition, operation i is by convention assigned to the machine with the smallest index.
5: **end for**
6: The resulting completion time vector is denoted by C^H.

The completion time vector C^H resulting from Algorithm 3 satisfies by construction

$$C_j^H \leq \sum_{i \in S_{j-1,k}} p_i + p_j \quad \text{for all } k \in M_j, \tag{4.6}$$

where $S_{j-1,k}$ is the set of operations in $\{1, 2, \ldots, j-1\}$ processed on machine k in the schedule C^H. Before the performance ratio of Algorithm 3 is proven, an auxiliary property of feasible solutions to the linear relaxation (4.5) is established.

Lemma 4.3. *Every point $C \in \mathbb{R}^N$ satisfying inequalities (4.4) and, w.l.o.g. $C_1 \leq C_2 \leq \cdots \leq C_N$, also satisfies, for each $j = 1, \ldots, N$*

$$\frac{m}{|M_j|} 2C_j^{LP} - \frac{m}{|M_j|} \frac{\sum_{i=1}^{j} p_i^2}{\sum_{i=1}^{j} p_i} \geq \frac{1}{|M_j|} \sum_{k \in M_j} \sum_{i \in S_{j-1,k}} p_i. \tag{4.7}$$

Proof. From inequalities (4.4) we have for $S = \{1, 2, \ldots, j\}$

$$\begin{aligned} p_j C_j \quad &\geq \quad \frac{1}{2m} \left(\sum_{i=1}^{j} p_i \right)^2 + \frac{1}{2} \sum_{i=1}^{j} p_i^2 - \sum_{i=1}^{j-1} p_i C_i \\ &\overset{(C_i \leq C_j)}{\geq} \frac{1}{2m} \left(\sum_{i=1}^{j} p_i \right)^2 + \frac{1}{2} \sum_{i=1}^{j} p_i^2 - \sum_{i=1}^{j-1} p_i C_j, \end{aligned}$$

and therefore,

$$C_j \geq \frac{1}{2m} \sum_{i=1}^{j} p_i + \frac{1}{2} \frac{\sum_{i=1}^{j} p_i^2}{\sum_{i=1}^{j} p_i}.$$

Consequently,

$$\begin{aligned} 2C_j - \frac{\sum_{i=1}^{j} p_i^2}{\sum_{i=1}^{j} p_i} &\geq \frac{1}{m} \sum_{i=1}^{j} p_i \\ &\geq \frac{1}{m} \sum_{k \in M_j} \sum_{i \in S_{j-1,k}} p_i + \frac{1}{m} \sum_{k \in M \setminus M_j} \sum_{i \in S_{j-1,k}} p_i, \end{aligned}$$

and thus,

$$\begin{aligned} \frac{m}{|M_j|} 2C_j^{LP} - \frac{m}{|M_j|} \frac{\sum_{i=1}^{j} p_i^2}{\sum_{i=1}^{j} p_i} &\geq \frac{1}{|M_j|} \sum_{k \in M_j} \sum_{i \in S_{j-1,k}} p_i \\ &\quad + \frac{1}{|M_j|} \sum_{k \in M \setminus M_j} \sum_{i \in S_{j-1,k}} p_i, \end{aligned}$$

implying inequalities (4.7). □

Theorem 4.4. *For* $FJ\,|\,n_i = 1, p_{i,k} = p_i\,|\,C_{\max}$ *let* val(*OPT*) *be the value of an optimal schedule and let* val(*Algorithm 3*) *be the value of the schedule produced by Algorithm 3. Then*

$$\text{val}(\textit{Algorithm 3}) \leq \left(\frac{2m}{\min\limits_j |M_j|} + 1\right)\left(1 - \frac{m}{\max\limits_j |M_j|\, n + m}\right)\text{val}(\textit{OPT})$$

Proof. Summing up inequalities (4.6) for all $k \in M_j$ and dividing the result by $|M_j|$, we obtain for each $j = 1, \ldots, N$

$$\begin{aligned}
C_j^H &\leq \frac{1}{|M_j|}\sum_{k\in M_j}\sum_{i\in S_{j-1,k}} p_i + p_j \\
&\leq \frac{1}{|M_j|}\sum_{k\in M_j}\sum_{i\in S_{j-1,k}} p_i + C_j^{LP} && (\text{since } C_j^{LP} \geq p_j) \\
&\leq \frac{m}{|M_j|}2C_j^{LP} - \frac{m}{|M_j|}\frac{\sum_{i=1}^j p_i^2}{\sum_{i=1}^j p_i} + C_j^{LP} && (\text{Lemma 4.3}) \\
&\leq \frac{m}{|M_j|}2C_j^{LP} - \frac{m}{|M_j|}\frac{1}{n}\sum_{i=1}^j p_i + C_j^{LP} && (\text{Cauchy-Schwarz inequality}) \\
&\leq \frac{m}{|M_j|}2C_j^{LP} - \frac{m}{|M_j|}\frac{1}{n}C_j^H + C_j^{LP} && (C_j^H \leq \sum_{i=1}^j p_i) \\
&= \left(\frac{2m}{|M_j|} + 1\right)C_j^{LP} - \frac{m}{|M_j|\,n}C_j^H,
\end{aligned}$$

implying

$$\begin{aligned}
& C_j^H\left(1 + \frac{m}{|M_j|\,n}\right) \leq \left(\frac{2m}{|M_j|} + 1\right)C_j^{LP} \\
\Leftrightarrow \quad & C_j^H \leq \left(\frac{2m}{|M_j|} + 1\right)\left(\frac{|M_j|\,n}{|M_j|\,n + m}\right)C_j^{LP} \\
\Leftrightarrow \quad & C_j^H \leq \left(\frac{2m}{|M_j|} + 1\right)\left(1 - \frac{m}{|M_j|\,n + m}\right)C_j^{LP}.
\end{aligned}$$

Consequently,

$$C_{\max}^H \leq \left(\frac{2m}{\min\limits_{j=1,\ldots,N}|M_j|} + 1\right)\left(1 - \frac{m}{\max\limits_{j=1,\ldots,N}|M_j|\,n + m}\right)C_{\max}^{LP}.$$

Let val(LP) be the optimal solution value of the linear relaxation (4.5). Since val(LP) $\leq$ val(OPT), the theorem follows. □

This performance-ratio also holds for the objective function of the total flow time $\sum C_i$, since, by the proof of Theorem 4.4,

$$C_j^H \leq \left(\frac{2m}{|M_j|}+1\right)\left(1-\frac{m}{|M_j|\,n+m}\right)C_j^{LP} \quad \text{for each } j=1,\ldots,N.$$

Corollary 4.5. *For* $FJ\,|\,n_i=1, p_{i,k}=p_i\,|\,\sum C_i$ *let* val(OPT) *be the value of an optimal schedule and let* val(*Algorithm 3*) *be the value of the schedule produced by Algorithm 3. Then*

$$\text{val}(\textit{Algorithm 3}) \leq \left(\frac{2m}{\min\limits_j |M_j|}+1\right)\left(1-\frac{m}{\max\limits_j |M_j|\,n+m}\right)\text{val}(OPT)$$

Example 4.6 shows, that for an arbitrary number of operations per job (and thus preceding constraints) the completion times produced by the list scheduling rule used in Algorithm 3 do not satisfy inequalities (4.6) in general.

Example 4.6. *Consider a single job consisting of two consecutive operations* $O_1 \to O_2$ *with processing times* p_1 *and* p_2*. The set of machines is given by* $M=\{\mu_1,\mu_2\}$ *and the valid machines for operation* O_1 *and operation* O_2 *are constituted by* $M_1=\{\mu_1\}$ *and* $M_2=\{\mu_2\}$*, respectively. Due to the precedence constraints and according to Algorithm 3, operation* O_1 *is scheduled on machine* μ_1 *with completion time* p_1 *and operation* O_2 *is scheduled on machine* μ_2 *with completion time* p_1+p_2*. Thus, the completion time of operation* O_2 *violates inequality (4.6), since*

$$p_1+p_2=C_2^H > \sum_{i\in S_{j-1,2}} p_i + p_2 = p_2.$$

4.2.2 $FJ\,|\,p_{i,k}=p_i\,|\,C_{\max}$

In this section, the approach from Section 4.2.1 is extended to the problem $FJ\,|\,p_{i,k}=p_i\,|\,C_{\max}$, that is each job consists of an arbitrary number of operations and, consequently, precedence constraints exist. As seen in Example 4.6, the approach from Section 4.2.1 is no longer applicable. The complexity of this problem arises from the simultaneous presence of precedence constraints and the circumstance that each operation i can only be executed by a subset $M_i \subseteq \{M_1,\ldots,M_m\}$ of all machines. In order to deal

with this difficulty, it is made use of the special structure of the flexible job shop scheduling problem. There are only precedence constraints between the operations of each job, so the jobs themselves are independent from each other. Let μ be the maximum number of operations per job. The original problem is subdivided in μ subproblems. Each subproblem $s = 1, \ldots, \mu$ only schedules the s^{th} operation of each job consisting of at least s operations. Since the jobs are independent from each other, a slightly modified version of Algorithm 3 can be used to find schedules for the subproblems. Subsequently, the schedules for the subproblems are plainly stringed together, resulting in an overall schedule for the primary problem with an estimable objective value. A linear relaxation of subproblem $s = 1, \ldots, \mu$ is given by

$$\begin{aligned} \min \quad & C_{\max} \\ \text{s.t.} \quad & \sum_{i \in S} p_i C_i \geq \frac{1}{2m} \left(\sum_{i \in S} p_i \right)^2 + \frac{1}{2} \sum_{i \in S} p_i^2 \qquad \text{for all } S \subseteq O_s \\ & C_i \geq p_i \qquad\qquad\qquad\qquad\qquad\qquad\quad \text{for all } i \in O_s, \end{aligned} \tag{4.8}$$

where $O_s = \{O_{i,s} \,|\, i = 1, \ldots, n \wedge n_i \geq s\}$. For $s = 1, \ldots, \mu$ let n^s be the number of jobs with at least s operations. The approximation algorithm for the problem $FJ \,|\, p_{i,k} = p_i \,|\, C_{\max}$ is now formally summarized in Algorithm 4.

Algorithm 4 Approximation Algorithm for the problem $FJ \,|\, p_{i,k} = p_i \,|\, C_{\max}$

1: **for all** $s = 1, \ldots, \mu$ **do**
2: Find an optimal solution $C^{LP_s} \in \mathbb{R}^{n^s}$ to the linear relaxation (4.8).
3: Index the operations such that $C_1^{LP_s} \leq C_2^{LP_s} \leq \cdots \leq C_{n^s}^{LP_s}$.
4: **for all** $i = 1, \ldots, n^s$ **do**
5: Schedule operation i on that machine $k \in M_i$ on which it can start as early as possible. If more than one machine meets this condition, operation i is assigned to the machine with the smallest index.
6: **end for**
7: The resulting completion time vector for the subproblem s is denoted by $C^s \in \mathbb{R}^{n^s}$.
8: **end for**
9: The overall completion time vector $C^H \in \mathbb{R}^N$ is set to

$$C^H = \left(C^1, C^2 + C^1_{\max}, \ldots, C^\mu + C^{\mu-1}_{\max} \right).$$

Theorem 4.7. *For* $FJ \,|\, p_{i,k} = p_i \,|\, C_{\max}$ *let* val(OPT) *be the value of an optimal schedule and let* val(*Algorithm 4*) *be the value of the schedule produced by Algorithm 4. Then,*

$$\text{val}(\textit{Algorithm 4}) \leq \mu \left(\frac{2m}{\min\limits_j |M_j|} + 1 \right) \left(1 - \frac{m}{\max\limits_j |M_j|\, n + m} \right) \text{val}(OPT)$$

Proof. For the completion time vector C^H resulting from Algorithm 4, we have

$$\begin{aligned}
C^H_{\max} &= \sum_{s=1}^{\mu} C^s_{\max} && \text{(Algorithm 4)} \\
&\leq \sum_{s=1}^{\mu} \left(\frac{2m}{\min\limits_j |M_j|} + 1 \right) \left(1 - \frac{m}{\max\limits_j |M_j|\, n + m} \right) C^{LP_s}_{\max} && \text{(Theorem 4.4)} \\
&\leq \sum_{s=1}^{\mu} \left(\frac{2m}{\min\limits_j |M_j|} + 1 \right) \left(1 - \frac{m}{\max\limits_j |M_j|\, n + m} \right) C^{LP}_{\max} \\
&= \mu \left(\frac{2m}{\min\limits_j |M_j|} + 1 \right) \left(1 - \frac{m}{\max\limits_j |M_j|\, n + m} \right) C^{LP}_{\max},
\end{aligned}$$

where C^{LP} is the optimal solution to the linear relaxation (4.5). Let val(LP) be the value of the optimal solution to the LP relaxation (4.5). Since val(LP) $\leq$ val(OPT), the theorem follows. □

This theorem also provides a guarantee for the quality of the lower bound obtained by solving the linear relaxation (4.5), since, by the proof of Theorem 4.7 given above,

$$\begin{aligned}
\text{val(OPT)} &\leq \text{val(Algorithm 4)} \\
&\leq \mu \left(\frac{2m}{\min\limits_j |M_j|} + 1 \right) \left(1 - \frac{m}{\max\limits_j |M_j|\, n + m} \right) \text{val(LP)}.
\end{aligned}$$

Taking the reciprocal of the performance-ratio yields the following corollary.

Corollary 4.8. *For* $FJ \,|\, p_{i,k} = p_i \,|\, C_{\max}$ *let* val(OPT) *be the value of an optimal schedule and let* val(LP) *be the value obtained from solving (4.5).*

Then

$$\text{val}(LP) \geq \frac{1}{\mu}\left(\frac{\min\limits_j |M_j|}{2m + \min\limits_j |M_j|}\right)\left(1 + \frac{m}{\max\limits_j |M_j|\, n}\right)\text{val}(OPT).$$

However, the performance-ratio of Algorithm 4 is considerably too large in order to justify a practical application. Therefore, an efficient LP-based heuristic making use of the models developed in Chapter 3 is presented in the next section.

4.2.3 Structured Time-Expanded Formulation

In this section, an LP-based heuristic using a modified version of the model TEF introduced in Section 3.4 is presented. In contrast to the models IPF, GPF and MPF, model TEF does not contain any constraints using Big-M constants. As described in Sections 3.4 and 3.6.3, constraints using Big-M constants significantly corrupt the sharpness of the linear relaxation. Therefore, model TEF is chosen as the basis for an underlying model of the LP-based heuristic.

In the following, the flexible job shop scheduling problem with identical processing times, i.e., $FJ \,|\, p_{i,k} = p_i \,|\, C_{\max}$, is considered and additionally, Assumption 1 is made.

Assumption 1. *There are $\tilde{m}$ machine groups $\mathcal{M}_1, \ldots, \mathcal{M}_{\tilde{m}}$ such that*

$$\bigcup_{k=1}^{\tilde{m}} \mathcal{M}_k = M \text{ and } \mathcal{M}_k \cap \mathcal{M}_l = \emptyset \quad \text{for all } k \neq l \in 1, \ldots, \tilde{m}.$$

Furthermore, the set of valid machines M_i for operation i is given by

$$M_i = \mathcal{M}_k \quad \text{for some } k \in 1, \ldots, \tilde{m}.$$

Consequently, each operation is dedicated to exactly one machine group.

From a practical point of view, this assumption is reasonable, since manufacturing facilities are actually often subdivided into machine groups, e.g. several (different) machines for welding, lathing, or milling. Under Assumption 1 , the problem $FJ \,|\, p_{i,k} = p_i \,|\, C_{\max}$ is now modeled as follows. Let T be an upper bound for the makespan $C_{\max}$. First of all, binary vari-

ables are introduced marking the beginning of the processing of an operation. The earliest possible starting time α_i for operation $i \in O$ is now given by

$$\alpha_i = \sum_{j \in B_i} p_j,$$

and the latest possible starting time β_i of operation $i \in O$ is now defined by

$$\beta_i = T - \sum_{j \in A_i} p_j + p_i.$$

Consequently, for all $\alpha_i \leq t \leq \beta_i$, $i \in O$ the binary variables $x_{i,t}$ are introduced with

$$x_{i,t} = \begin{cases} 1, & \text{if operation } i \text{ starts at time } t \\ 0, & \text{otherwise.} \end{cases}$$

Furthermore, the makespan is defined by

$$\sum_{t=\alpha_i}^{\beta_i} x_{i,t}(t + p_i) \leq C_{\max} \quad \text{for all } i \in O : P(i) = n_{J(i)}. \tag{4.9}$$

The precedence constraints are given by

$$\sum_{t=\alpha_i}^{\beta_i} x_{i,t}(t + p_i) - \sum_{t=\alpha_j}^{\beta_j} x_{j,t} t \leq 0 \quad \text{for all } i, j \in C, \tag{4.10}$$

and furthermore, each operation has to be scheduled at exactly one point in time, which is assured by

$$\sum_{t=\alpha_i}^{\beta_i} x_{i,t} = 1 \quad \text{for all } i \in O. \tag{4.11}$$

For each machine group $\mathcal{M}_k$, $k = 1, \ldots, \tilde{m}$, it has to be assured that there are not more than $\mathcal{M}_k$ operations scheduled at the same time. Let $\tilde{T}_{i,t}$ be defined by

$$\tilde{T}_{i,t} = \{\tau \,|\, \alpha_i \leq \tau \leq \beta_i,\ t - p_i + 1 \leq \tau \leq t\} \quad \text{for all } i \in O,\ t = 1, \ldots, T.$$

If operation $i \in O$ starts at time $\tau \in \tilde{T}_{i,t}$, then operation i occupies machine group M_i at time t. Therefore, the maximal capacity of the machine group

is preserved by

$$\sum_{i \in O} \sum_{\tau \in \tilde{T}_{i,t}} x_{i,\tau} \leq |\mathcal{M}_j| \quad \text{for all } k = 1, \ldots, \tilde{m},\ t = 1, \ldots, T. \tag{4.12}$$

Under Assumption 1, each feasible solution $x = (x_{i,t})$ for the problem $FJ \,|\, p_{i,k} = p_i \,|\, C_{\max}$ satisfies constraints (4.10), (4.11), and (4.12). By Lemma 4.9, constraints (4.10), (4.11), and (4.12) are sufficient, too.

Lemma 4.9. *Let $x = (x_{i,t})$ satisfy constraints (4.10), (4.11) and (4.12). Then, x is a feasible solution to the problem $FJ \,|\, p_{i,k} = p_i \,|\, C_{\max}$ under Assumption 1.*

First of all, the following definitions from graph theory are needed for the proof.

Definition 4.2. *An undirected graph G is called an* interval graph *if its vertices can be put into one-to-one correspondence with a set of intervals of a linearly ordered set such that two vertices are connected by an edge of G if and only if their corresponding intervals have nonempty intersection [Gol80, p. 13].*

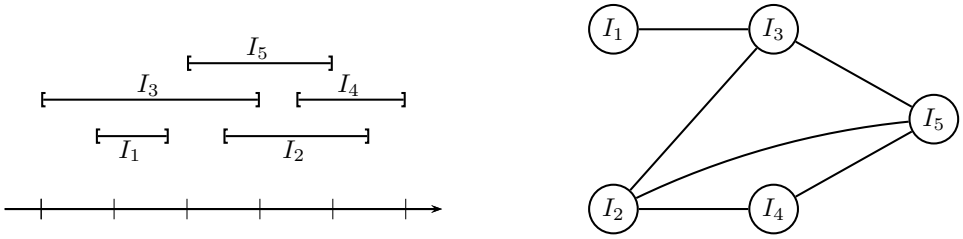


Figure 4.1: A set of intervals and the corresponding interval graph

Definition 4.3. *A* c-coloring *of an undirected graph $G = (V, E)$ is a partition of the vertices $V = X_1 + X_2 + \cdots + X_c$ such that each X_i is a stable set, i.e. a subset of vertices no two of which are adjacent. In such a case, the members of X_i are painted with color i and adjacent vertices will receive different colors [Gol80, p. 7].*

Proof. Let $x = (x_{i,t})$ satisfy constraints (4.10), (4.11), and (4.12). Obviously, each operation starts at exactly one point in time by constraints (4.11) and the precedence constraints are satisfied by constraints (4.10). Furthermore,

at each point in time, at most $|\mathcal{M}_j|$ operations are assigned to machine group $\mathcal{M}_j$, $j = 1, \ldots, \tilde{m}$ by constraints (4.12). Each operation i corresponds to an interval

$$I_i = \left[\sum_{t=\alpha_i}^{\beta_j} x_{i,t} \cdot t,\ \sum_{t=\alpha_i}^{\beta_j} x_{i,t}(t + p_i) \right],$$

and for $j = 1, \ldots, \tilde{m}$ machine group $\mathcal{M}_j$ is associated with a set of intervals $\mathcal{I}_j$ given by

$$\mathcal{I}_j = \{ I_i \,|\, i \in O \wedge M_i = \mathcal{M}_j \}.$$

For $j = 1, \ldots, \tilde{m}$, the set of intervals $\mathcal{I}_j$ is now represented as an interval graph denoted by G^j.

Now it is shown, that the interval graph G^j can be colored by an algorithm in $\mathcal{O}(N \log(N))$ time using at most $|\mathcal{M}_j|$ colors. To that end, the left edge algorithm stated in Algorithm 5 is used.

Algorithm 5 Left Edge Algorithm [HS71]

Require: A set of intervals

$$\{ [l_i, r_i] \,|\, l_i \leq r_i,\ i = 1, \ldots, N \}$$

and a set of colors $\{c_1, \ldots, c_N\}$

1: Sort the intervals in order of nondecreasing left end point.
2: Color the intervals in this order by assigning to each interval $[l_i, r_i]$ the color with the smallest index that has not yet been assigned to an interval overlapping $[l_i, r_i]$.

Suppose that the left edge algorithm uses $|\mathcal{M}_j| + 1$ colors. Thus, at some point in time an interval I_i is assigned to the $(|\mathcal{M}_j| + 1)$ th color. Step 2 of the left edge algorithm implies that $|\mathcal{M}_j|$ other intervals overlap with interval I_i. This is a contradiction to constraint (4.12). Consequently, the left edge algorithm uses at most $|\mathcal{M}_j|$ colors. Furthermore, it follows that the operations can be assigned to the machines of machine group $\mathcal{M}_j$ without overlapping and x is a feasible solution to the problem $FJ \,|\, p_{i,k} = p_i \,|\, C_{\max}$ under Assumption 1. □

By Lemma 4.9, the mixed integer program formulation STEF (Structured Time-Expanded Formulation) for the problem $FJ \,|\, p_{i,k} = p_i \,|\, C_{\max}$ under Assumption 1 is now given by

$$\min \quad C_{\max} \qquad \text{(STEF)}$$

s.t.

$$\sum_{t=\alpha_i}^{\beta_i} x_{i,t}(t+p_i) \leq C_{\max} \quad \text{for all } i \in O : P(i) = n_{J(i)}$$

$$\sum_{t=\alpha_i}^{\beta_i} x_{i,t}(t+p_i) - \sum_{t=\alpha_j}^{\beta_j} x_{j,t} \cdot t \leq 0 \quad \text{for all } (i,j) \in C$$

$$\sum_{t=\alpha_i}^{\beta_i} x_{i,t} = 1 \quad \text{for all } i \in O$$

$$\sum_{i \in O} \sum_{\tau \in \tilde{T}_{i,t}} x_{i,\tau} \leq |\mathcal{M}_j| \quad \text{for all } k = 1, \ldots, \tilde{m}, \; t = 1, \ldots, T$$

$$x_{i,t} \in \{0,1\} \quad \text{for all } i \in O, \; t = \alpha_i, \ldots, \beta_i.$$

An upper bound T can be achieved by a simple scheduling rule. First of all, consider the first operation of each job and schedule these operations successively according to a priority rule as early as possible. Then, consider the second operation of each job with at least two operations and schedule these operations successively according to a priority rule as early as possible, and so forth, until all operations are scheduled. This procedure is formally summarized in Algorithm 6.

Algorithm 6 Scheduling Algorithm

1: **for all** $p = 1, \ldots, \mu$ **do**
2: **for all** operations $i \in O$ with $P(i) = p$ **do**
3: Schedule operation i as early as possible after the preceding operation of job $J(i)$ is completed and a machine is available.
4: **end for**
5: **end for**

Lemma 4.10. *Under Assumption 1, algorithm 6 produces a feasible schedule for* $FJ \,|\, p_{i,k} = p_i \,|\, C_{\max}$. *Furthermore, let* $\text{val}(OPT)$ *be the value of an optimal schedule of* $FJ \,|\, p_{i,k} = p_i \,|\, C_{\max}$ *under Assumption 1 and let* $\text{val}(\textit{Algorithm 6})$ *be the value of the schedule produced by Algorithm 6. Then*

$$\text{val}(\textit{Algorithm 6}) \leq \mu \max_{k=1,\ldots,\tilde{m}} |\mathcal{M}_k| \, \text{val}(OPT)$$

Proof. Before an operation with position p is considered, all operations with position $p-1$ are scheduled. Additionally, an operation is not scheduled before the preceding operation is completed. Therefore, the precedence constraints are satisfied. By step 3, an operation is only scheduled when a machine is actually available. Consequently, each machine processes at most one operation at a time and as a result, Algorithm 6 produces a feasible schedule.

Furthermore, let $\mathcal{J}_k$ be the set of operations that are assigned to machine group $\mathcal{M}_k$, $k = 1, \ldots, \tilde{m}$, i.e. $\mathcal{J}_k = \{i \in O \,|\, \mathcal{M}_i = \mathcal{M}_k\}$. Recall that μ denotes the maximum number of operation per job. Then,

$$\begin{aligned}
\text{val(Algorithm 6)} &\leq \mu \max_{k=1,\ldots,\tilde{m}} \sum_{i \in \mathcal{J}_k} p_i \\
&= \mu \max_{k=1,\ldots,\tilde{m}} |\mathcal{M}_k| \sum_{i \in \mathcal{J}_k} \frac{p_i}{|\mathcal{M}_k|} \\
&\leq \mu \max_{k=1,\ldots,\tilde{m}} |\mathcal{M}_k| \,\text{val(OPT)},
\end{aligned}$$

since $\text{val(OPT)} \geq \max_{k=1,\ldots,\tilde{m}} \sum_{i \in \mathcal{J}_k} \frac{p_i}{|\mathcal{M}_k|}$. □

Due to the model structure, a reasonable ordering of starting times of the operations can be derived from the solution of the linear relaxation of model STEF. The necessity of an assignment of operations to machines in model TEF is replaced by a predefined assignment of operations to machine groups in model STEF and by constraints (4.11),

$$\sum_{t=\alpha_i}^{\beta_i} x_{i,t} = 1 \quad \text{for all } i \in O.$$

Consequently, a feasible solution of the linear relaxation corresponds to a probability distribution of starting times for each operation.

In order to achieve a feasible solution to the flexible job shop scheduling problem, first a specific starting time for each operation is derived from the solution of the linear relaxation. In the following, two different approaches to transforming the solution of the linear relaxation into a specific starting time for each operation are presented.

Approach 1 For each operation $i \in O$ the starting time S_i is defined as

$$S_i := \sum_{t=\alpha_i}^{\beta_i} x_{i,t} \cdot t.$$

By this choice, the precedence constraints are satisfied, since

$$S_i + p_i - S_j = \sum_{t=\alpha_i}^{\beta_i} x_{i,t} \cdot t + p_i - \sum_{t=\alpha_j}^{\beta_j} x_{j,t} \cdot t \overset{(4.10)}{\leq} 0 \quad \text{for all } (i,j) \in C.$$

Furthermore, the makespan corresponding to the starting times S_i equals the makespan of the solution of the linear relaxation, since

$$S_i + p_i = \sum_{t=\alpha_i}^{\beta_i} x_{i,t} t + p_i \overset{(4.9)}{\leq} C_{\max} \quad \text{for all } i \in O : P(i) = n_{J(i)}.$$

Still, due to the machine group capacities, the schedule corresponding to the starting times S_i is in general not feasible, see Example 4.11.

Example 4.11. *Consider a machine group with one machine and two operations i and j with $x_{i,t} = x_{j,t} = \frac{1}{2}$ for $t = 1$ and $t = 3$. This is a feasible solution to the linear relaxation of model STEF. Now, the defined starting times $S_i := \sum x_{i,t} \cdot t = \frac{1}{2}1 + \frac{1}{2}3 = 2$ and $S_j := \sum x_{j,t} \cdot t = \frac{1}{2}1 + \frac{1}{2}3 = 2$ violate the machine group capacity constraints 4.12, since both operation i and j are scheduled at the same time on the only machine of the machine group.*

Approach 2 For each operation $i \in O$, the starting time S_i is defined as

$$S_i := \operatorname*{argmax}_{t=\alpha_i,\ldots,\beta_i} x_{i,t}.$$

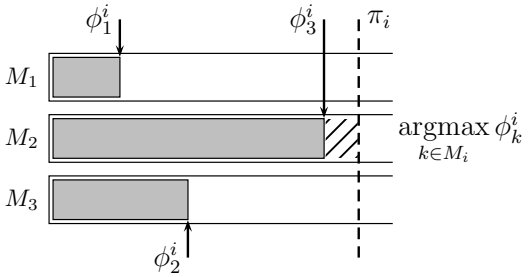
In this case, no statements can be made about the feasibility of the defined starting times. Still, the choice is rational, since for each operation the most likely starting time according to the solution of the linear relaxation of the model STEF is chosen.

The operations $i \in O$ are now indexed such that $S_1 \leq S_2 \leq \ldots \leq S_N$. In order to use the starting times as an input for a list scheduling heuristic, at least the precedence constraints have to be satisfied. For the first approach

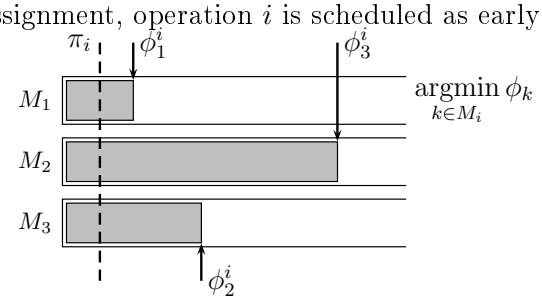
to defining the starting times, these constraints are satisfied by definition. For the second approach to defining the starting times, the operations have to be rearranged with respect to a correct ordering of the positions $P(i)$ of the operations.

The operations are now scheduled according to the given ordering as early as possible and the resulting schedule is denoted by S^H. In the course of scheduling the operations as early as possible, three different cases have to be distinguished. Consider scheduling operation i and let π_i be the completion time of the preceding operation, i.e., $\pi_i = S_j^H + p_j$, where $j \in O$ with $J(j) = J(i)$ and $P(j) = P(i) - 1$. If $P(i) = 1$, then $\pi_i = 0$. Furthermore, let ϕ_k^i be the time when machine k becomes idle with respect to the scheduled operations $\{1, \ldots, i-1\}$.

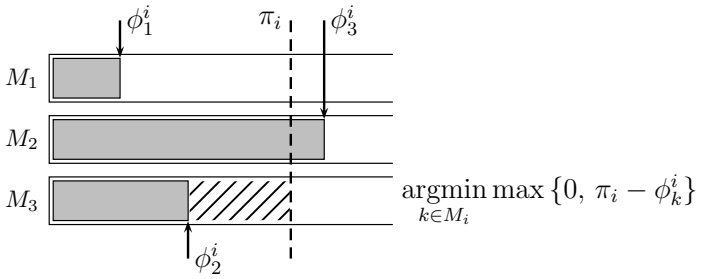
Case 1: $\left(\pi_i \geq \max\limits_{k \in M_i} \phi_k^i\right)$ In case the preceding operation to operation i is completed later than any valid machine is idle, operation i is assigned to machine $m = \operatorname{argmax}\limits_{k \in M_i} \phi_k^i$ and the starting time is given by $S_i^H = \pi_i$. By this assignment, operation i is scheduled as early as possible and the idle time $[\phi_m^i, \pi_i]$ is minimized.



Case 2: $\left(\pi_i \leq \min\limits_{k \in M_i} \phi_k^i\right)$ In case the preceding operation to operation i is completed before any valid machine is idle, operation i is assigned to machine $m = \operatorname{argmin}\limits_{k \in M_i} \phi_k$ and the starting time is given by $S_i^H = \min\limits_{k \in M_i} \phi_k^i$. By this assignment, operation i is scheduled as early as possible.



Case 3: $\left(\min\limits_{k\in M_i}\phi_k^i < \pi_i < \max\limits_{k\in M_i}\phi_k^i\right)$ In case the preceding operation to operation i is completed later than the first valid machine becomes idle and earlier than the last valid machine becomes idle, operation i is assigned to machine $m = \operatorname{argmin}\limits_{k\in M_i} \max\left\{0,\, \pi_i - \phi_k^i\right\}$ and the starting time is given by $S_i^H = \pi_i$. By this assignment, operation i is scheduled as early as possible, though it leads to an idle time of $\min\limits_{k\in M_i}\{\pi_i - \phi_m^i\}$ time units on machine k.



Eventually, operations can be scheduled in the idle time intervals emerging in Case 1 and Case 3. Assuming operation i is scheduled according to Case 1, machine k is idle during the interval $[\phi_k^i, \pi_i]$. Now, if there exists an operation $j \in \{i+1, \ldots, N\}$, such that the preceding operation to operation j is completed early enough to schedule operation j in the idle time interval $[\phi_k^i, \pi_i]$ and the idle interval is long enough to schedule operation j, i.e.,

$$\pi_j \leq \pi_i - p_j \quad \text{and} \quad \pi_j \leq \pi_i - \max\{\pi_j, \phi_k^i\},$$

then operation j is scheduled on machine k at $S_j^H = \max\{\pi_j, \phi_k^i\}$. Now, set $\phi_k^i = \phi_k^i + S_j^H + p_j$ and repeat this process until there is either no idle time left in the interval $[\phi_k^i, \pi_i]$ or there exists no operation satisfying the criteria described above. If operation i is scheduled according to Case 3, the procedure is analogous. Obviously, the preferred scheduling of operations during emerging idle time intervals may not deteriorate the resulting schedule with respect to the makespan.

Algorithm 7 summarizes the LP-based heuristic for the flexible job shop scheduling problem described above formally. Obviously, the quality of the solutions produced by Algorithm 7 is better if there is a reasonable subdivision of the set of machines M in disjoint machine groups $\mathcal{M}_j$, $j = 1, \ldots, \tilde{m}$, each operation is preassigned to a machine group and a subset of machines of the assigned machine group is valid for the operation. This setting is

approximately given for the instances in the practical situation described in Section 5. Still, Algorithm 7 also produces feasible solutions for arbitrary instances of the problem $FJ \,|\, p_{i,k} = p_i \,|\, C_{\max}$.

Lemma 4.12. *Algorithm 7 produces a feasible schedule for the problem* $FJ \,|\, p_{i,k} = p_i \,|\, C_{\max}$.

Proof. In order to achieve a feasible setting for model STEF, each operation is assigned to exactly one of the machine groups determined in Step 1 according to the maximum cardinality of the intersection of the set of valid machines for the operation and machines of the machine group.

In steps 8 - 31, the operations are actually scheduled. Thereby, the precedence constraints are always satisfied, since the operations are ordered such that $i < j$ for all $i, j : J(i) = J(j) \wedge P(i) < P(j)$ and in any case an operation is not scheduled earlier than the preceding operation is completed. This also holds good for the operations scheduled in the emerging idle time intervals, see Steps 23 - 26. Furthermore, each operation is scheduled at exactly one point in time on exactly one machine and each machine processes at most one operation at a time, since an operation is in any case not scheduled before the machine is idle. Consequently, the resulting schedule is a feasible solution to the problem $FJ \,|\, p_{i,k} = p_i \,|\, C_{\max}$. □

The LP-based heuristic presented in Algorithm 7 is implemented using the CPLEX environment provided by IBM ILOG (cf. Section 3.7) and the source code of the implementation can be found on the CD attached to this thesis. In the following, an empirical evaluation of the performance-ratio of the LP-based heuristic is presented. First of all, 100 random instances with $N = 20$ operations, $n = 6$ jobs, $m = 4$ machines, $\tilde{m} = 2$ machine groups, and flexibility $f = 0.8$ are generated as described in Section 3.7. Additionally, the machines are distributed uniformly among the machine groups and each machine group consists of at least one machine. Furthermore, each operation is randomly assigned to exactly one machine group. The set of valid machines for each operation is chosen as a subset of the assigned machine group, where each machine of the subset is valid for an operation with probability f. These instances are in the following referred to as small instances and are solvable to optimality by the optimizer in a reasonable amount of time. Consequently, the solution of the LP-based heuristic described in Algorithm 7

Algorithm 7 LP-based heuristic for the problem $FJ \,|\, p_{i,k} = p_i \,|\, C_{\max}$

1: Subdivide the set of machines M in disjoint machine groups $\mathcal{M}_j$, $j = 1, \ldots, \tilde{m}$ and assign each operation $i \in O$ to the machine group $\underset{j=1,\ldots,\tilde{m}}{\operatorname{argmax}} |M_i \cap \mathcal{M}_j|$.
2: Solve the linear relaxation of the model STEF with machine groups $\mathcal{M}_j$, $j = 1, \ldots, \tilde{m}$. Let $x^* = \left(x^*_{i,t}\right)$ be the solution of the linear relaxation.
3: Define the starting times S_i for $i \in O$ by

Approach 1: $S_i = \sum_{t=\alpha_i}^{\beta_i} x^*_{i,t} \cdot t$ **or Approach 2:** $S_i = \underset{t=\alpha_i,\ldots,\beta_i}{\operatorname{argmax}} x^*_{i,t}$

4: Index the operations such that $S_1 \leq S_2 \leq \cdots \leq S_N$.
5: **if** Approach 2 is chosen **then**
6: rearrange the operations such that $i < j$ for all $i, j : J(i) = J(j) \wedge P(i) < P(j)$.
7: **end if**
8: **for all** $i = 1, \ldots, N$ **do**
9: **if** operation i is not already scheduled by Step 25 **then**
10: **if** $\pi_i \geq \max_{k \in M_i} \phi^i_k$ **then**
11: $m = \underset{k \in M_i}{\operatorname{argmax}} \phi^i_k$
12: $S^H_i = \pi_i$
13: **else if** $\pi_i \leq \min_{k \in M_i} \phi^i_k$ **then**
14: $m = \underset{k \in M_i}{\operatorname{argmin}} \phi^i_k$
15: $S^H_i = \min_{k \in M_i} \phi^i_k$
16: **else if** $\min_{k \in M_i} \phi^i_k < \pi_i < \max_{k \in M_i} \phi^i_k$ **then**
17: $m = \underset{k \in M_i}{\operatorname{argmin}} \max\left\{0,\, \pi_i - \phi^i_k\right\}$
18: $S^H_i = \pi_i$
19: **end if**
20: Schedule operation i on machine m at starting time S^H_i.
21: **if** idle time interval $[\phi^i_k, \pi_i]$ exists **then**
22: **for all** $j = i+1, \ldots, N$ **do**
23: **if** $\pi_j \leq \pi_i - p_j \wedge p_j \leq \pi_i - \max\{\pi_j, \phi^i_k\}$ **then**
24: $S^H_j = \max\{\pi_j, \phi^i_k\}$
25: Schedule operation j on machine m at starting time S^H_j.
26: Set $\phi^i_k = \phi^i_k + S^H_j + p_j$.
27: **end if**
28: **end for**
29: **end if**
30: **end if**
31: **end for**

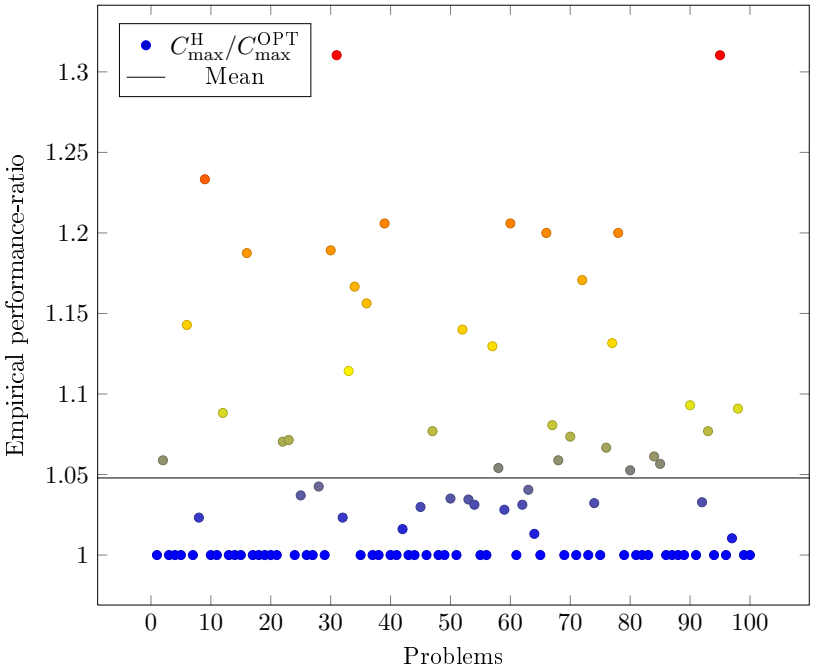


Figure 4.2: Empirical performance-ratio of the LP-based heuristic using Approach 1 for small instances

can be compared to the optimal solution. In Figures 4.2 and 4.3 the quotient of the solution of the LP-based heuristic C^{H}_{max} using Appproach 1 and Approach 2, respectively, and the optimal solution C^{OPT}_{max} is plotted for the small instances. The results indicate an average (empirical) performance-ratio of 1.047 and 1.082 for Approach 1 and Approach 2 respectively, and substantiate the reasoning for the LP-based heuristic. Additionally, the average computational time for the small instances of the LP-based heuristic amounts to 0.06 seconds for both Approach 1 and Approach 2.

In order to evaluate the LP-based heuristic in comparison to the basic scheduling algorithm presented in Algorithm 6, Figure 4.4 depicts the quotient of the solution of the LP-based heuristic C^{H}_{max} using Approach 1 and the solution of Algorithm 6 denoted by C^{SA}_{max} for the small instances. In very few instances the basic scheduling algorithm outperforms the LP-based heuristic with respect to the makespan, but on average the LP-based heuristic performs 7.15 % better than the basic scheduling algorithm.

As discussed in Section 3, for larger instances an optimal solution cannot be obtained in reasonable time. Therefore, the solution of the LP-based

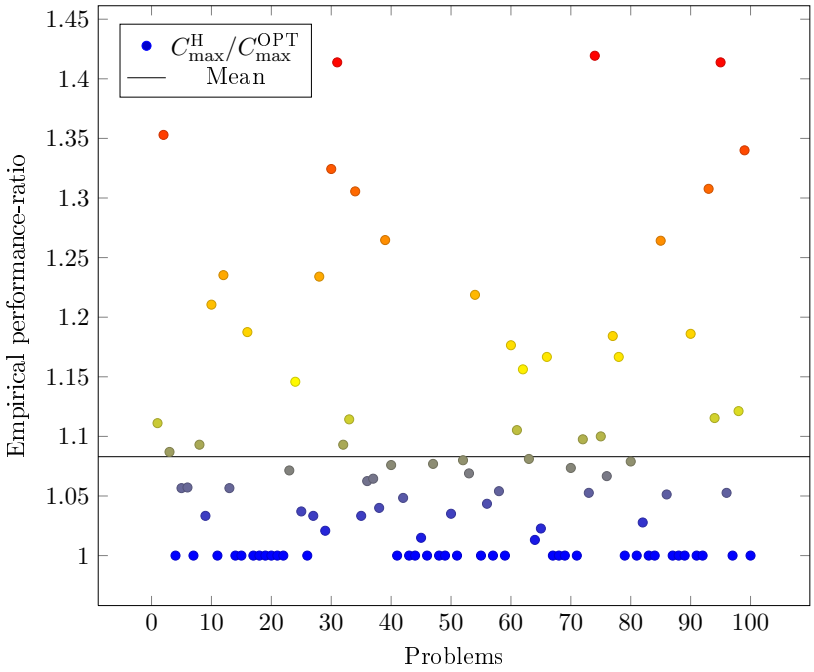


Figure 4.3: Empirical performance-ratio of the LP-based heuristic using Approach 2 for small instances

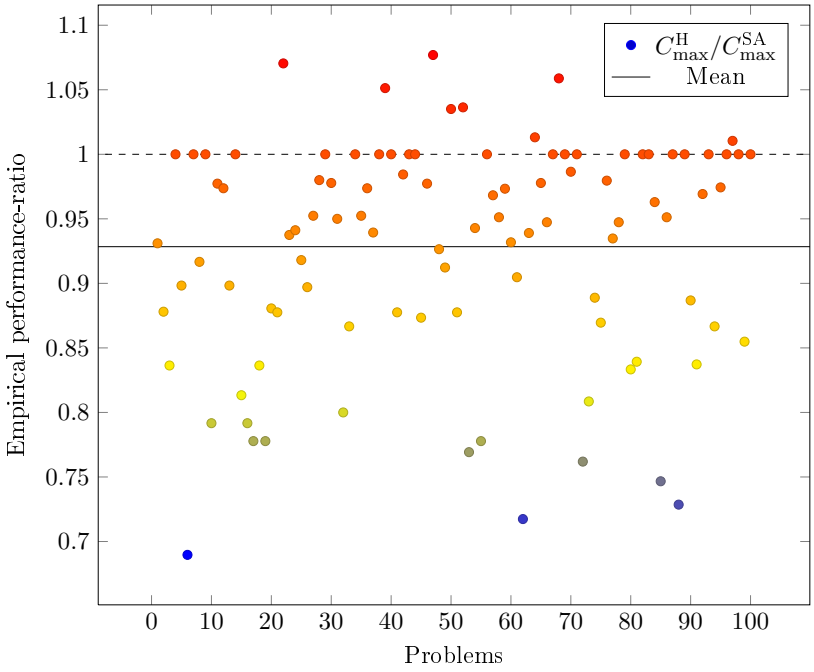


Figure 4.4: Performance of the LP-based heuristic using Approach 1 compared to the basic scheduling algorithm for the small instances

heuristic is now compared to the solution of the linear relaxation of model TEF, which constitutes a lower bound for the optimal solution. The minimal maximum machine load $\Pi_{\max}^{\mathrm{OPT}}$ and the maximum job length $P_{\max}$ constitute lower bounds for the optimal solution, too. Still, for the larger instances generated in the course of these computational studies, the maximum job length $P_{\max}$ is always smaller than the solution of the linear relaxation of model TEF. On average, the solution of the linear relaxation of model TEF is 5.60 % larger then the maximum job length $P_{\max}$. Furthermore, the determination of the minimal maximum machine load $\Pi_{\max}^{\mathrm{OPT}}$ is $\mathcal{NP}$-hard, as mentioned in Section 4.1. Thus, the linear relaxation of model TEF is a reasonable choice as a lower bound for the purposes of these computational studies.

In Figures 4.5 and 4.6, the quotient of the solution of the LP-based heuristic using Approach 1 and Approach 2, respectively, and the solution of the linear relaxation of model TEF is plotted for 100 instances with $N = 200$ operations, $n = 60$ jobs, $m = 30$ machines, $\tilde{m} = 5$ machine groups, and flexibility $f = 0.7$ generated as described above. These instances are in the following referred to as large instances. The results indicate an average (empirical) performance-ratio of 1.329 and 1.377 for Approach 1 and Approach 2, respectively. The reasoning for the LP-based heuristic is substantiated further by these results, since the performance-ratio of the LP-based heuristic is in this case measured against a probably weak lower bound and is still in an acceptable range. For both the small and the large instances, Approach 1 performs slightly better than Approach 2. The average computational time for the large instances of the LP-based heuristic amounts to 0.82 seconds for both Approach 1 and Approach 2.

The comparison of the LP-based heuristic and the basic scheduling algorithm presented in Algorithm 6 is also carried out for the large instances. In Figure 4.7, the quotient of the solution of the LP-based heuristic $C_{\max}^{\mathrm{H}}$ using Approach 1 and the solution of Algorithm 6 $C_{\max}^{\mathrm{SA}}$ is plotted for the large instances. Here, the LP-based heuristic always performs better than the basic scheduling algorithm and only for a few instances, the basic scheduling algorithm achieves the same objective value than the LP-based heuristic. On average, the LP-based heuristic performs 12.3 % better than the basic scheduling algorithm. For the large instances, the relative performance of the LP-based heuristic with respect to the basic scheduling algorithm is bet-

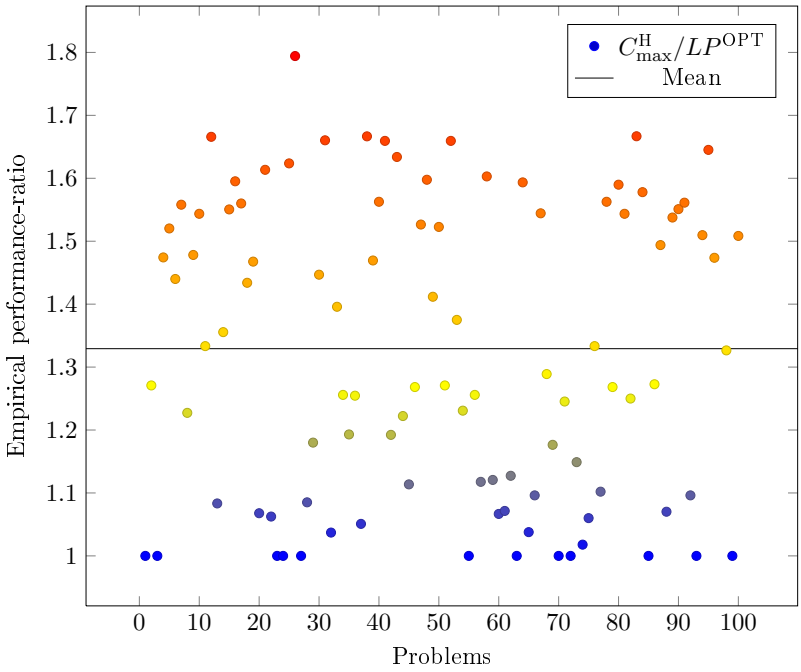


Figure 4.5: Empirical performance-ratio of the LP-based heuristic using Approach 1 for large instances

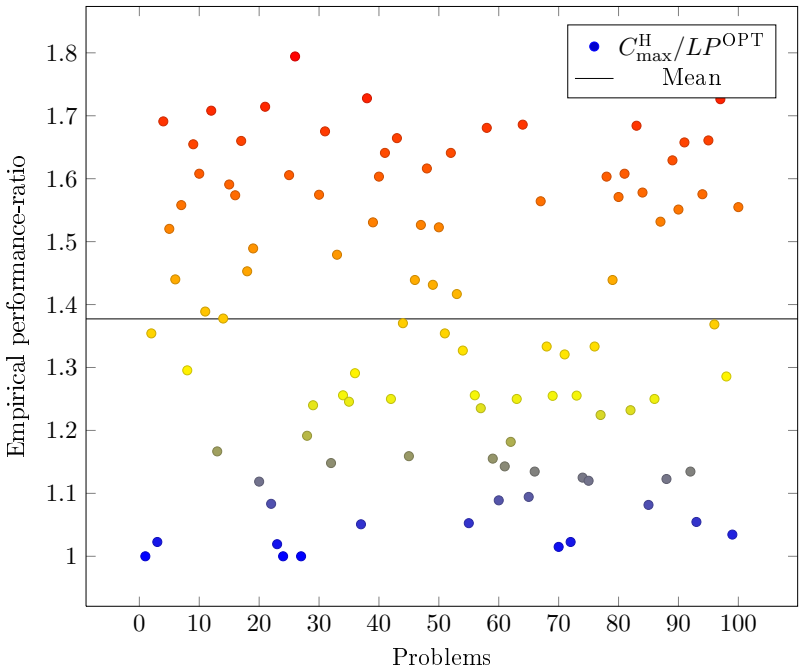


Figure 4.6: Empirical performance-ratio of the LP-based heuristic using Approach 2 for large instances

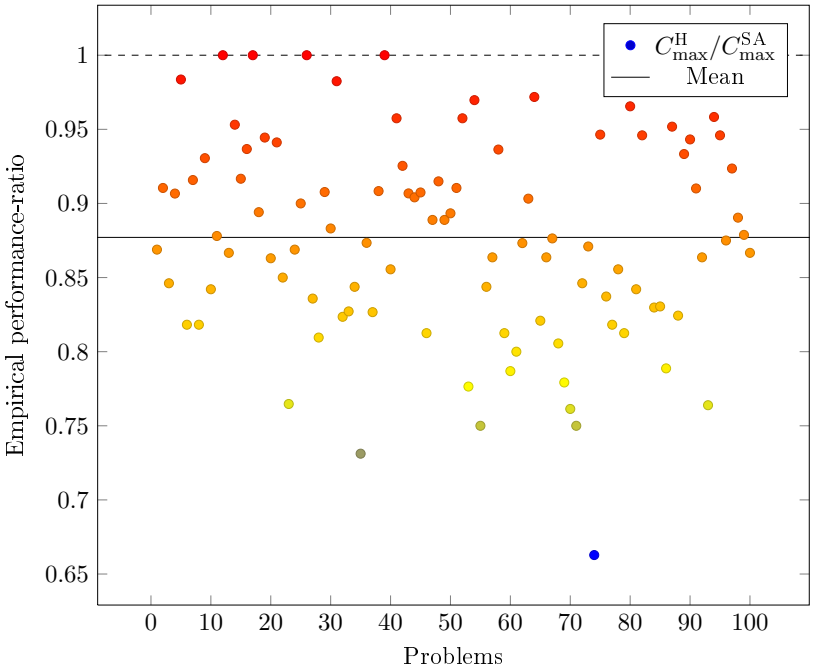


Figure 4.7: Performance of the LP-based heuristic using Approach 1 compared to the basic scheduling algorithm for the large instances

ter than for the small instances, since the number of possibilities to arrange the operations increases significantly and the basic scheduling algorithm only employs the ordering of operations for each job.

The computational results for the LP-based heuristic are summarized in Table 4.1, whereat $\varnothing\rho$ denotes the average empirical performance-ratio. It is noted, that for the small instances the empirical performance-ratio is measured against the optimal solution, whereas for the large instances it is measured against the solution of the linear relaxation, which constitutes only a lower bound for the optimal solution. Furthermore, $\varnothing$ CPU denotes the average processing time in seconds.

The bottom line is that even for large instances of the flexible job shop

Approach	n	m	$\tilde{m}$	N	f	$\varnothing\rho$	$\varnothing$ CPU
1	6	4	2	20	0.8	1.047	0.06
2	6	4	2	20	0.8	1.082	0.06
1	60	30	5	200	0.8	1.329	0.82
2	60	30	5	200	0.8	1.377	0.82

Table 4.1: Computational Results for LP-based Heuristic

scheduling problem approximately featuring a machine group structure, the LP-based heuristic presented in this section produces schedules with a make-span close to the optimum in efficient time. In Figure 4.8, a machine-oriented Gantt chart of a schedule produce with the presented LP-based heuristic for a randomly generated large instance (60 jobs, 30 machines, 5 machine groups, flexibility 0.7) is depicted. Operations belonging to the same job are colored equally.

In the following chapter, the heuristic's operational capability is examined by means of actual instances of the flexible job shop scheduling problem provided by a manufacturer of vacuum chambers.

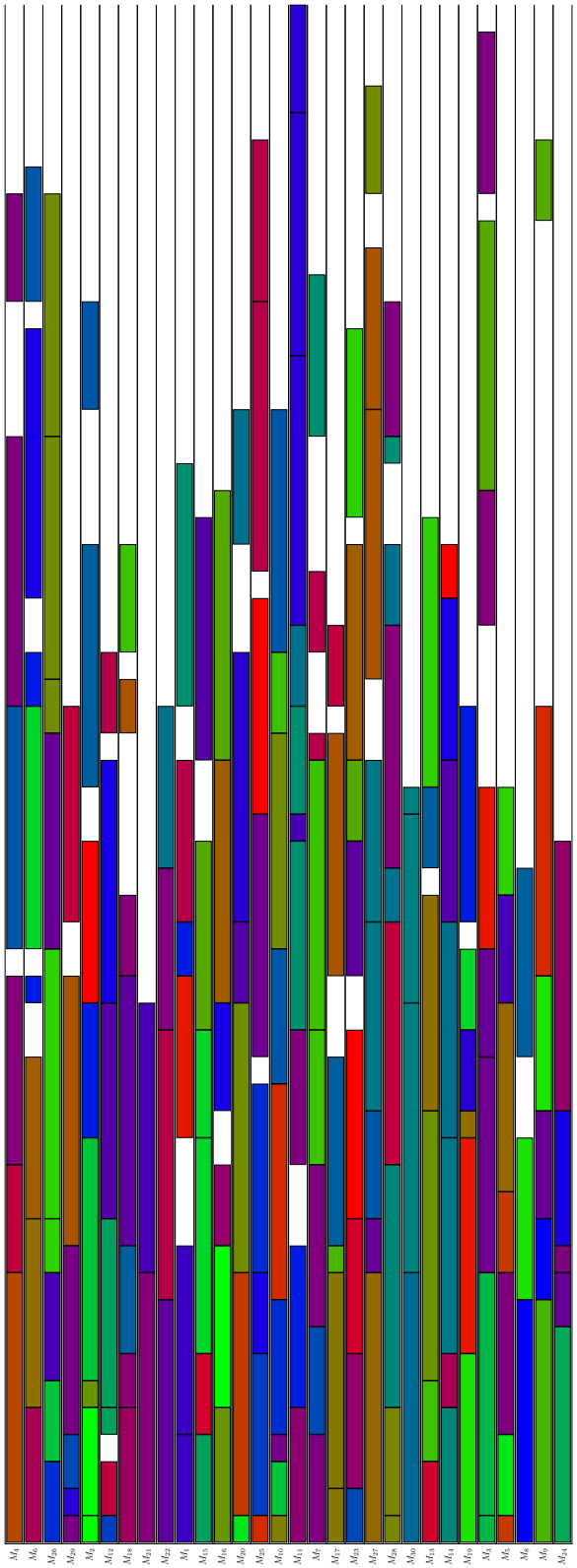


Figure 4.8: Machine-oriented Gantt chart for a large instance produced with the LP-based heuristic

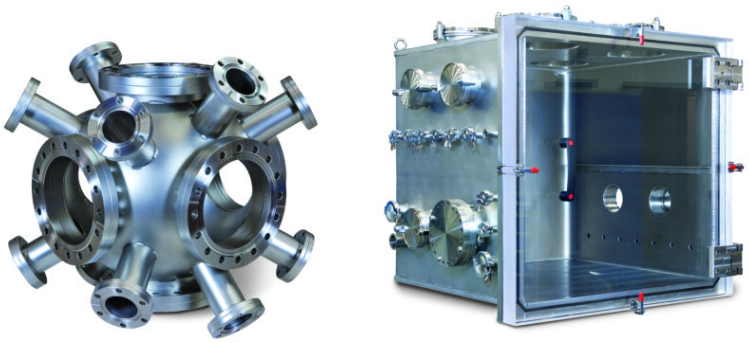
Chapter 5

Practical Application

This chapter focuses on the practical application of the LP-based heuristic developed and evaluated in Section 4.2.3. The subject of this master thesis, the flexible job shop scheduling problem, is among other things motivated by an actual scheduling problem of Trinos Vakuum-System GmbH, a manufacturer of vacuum chambers. A brief outline of the implementation of the LP-based heuristic with respect to practical environment is given in this chapter. In Section 5.1, a brief characterization of the company is given. Furthermore, in Section 5.2, the data provided by Trinos Vakuum-Systeme GmbH is specified. Finally, the model of the flexible job shop scheduling problem is adjusted in order meet the requirements of the given task and the results of the LP-based heuristic for actual instances provided by Trinos Vakuum-System GmbH are presented in Section 5.3.

5.1 Trinos Vakuum-Systeme GmbH

Trinos Vakuum-Systeme GmbH is a leading manufacturer of vacuum chambers, vacuum systems and special components for high-end applications in industry and research. The company's current product lines include chambers, components, fittings, feedthroughs, valves, manipulators, and complete systems for high and ultra high vacuum applications, see Figure 5.1. The in-house manufacturing facility is fitted with a variety of machines and production equipment, for example 5-axis machining centers, multiple welding stations, cleaning facilities, a clean room, assembly areas, and a leak testing laboratory. Each job consists of a multitude of individual operations



(a) (b)

Figure 5.1: A spherical chamber 5.1(a) and a standard cubical chamber with custom ports 5.1(b)

that have to be executed at the different stations. In order to ensure the completion of each job within its stated period and approach full capacity, a detailed planning of machine assignments and scheduling is essential. Furthermore, the machine environment of Trinos Vakuum-Systeme GmbH corresponds to a flexible job shop, since each operation may be processed by any machine from a given set. For instance, several milling machines are available in the manufacturing facility and due to the type and complexity of an operation, all milling machines are valid for the operation or only a subset is valid for the operation. In the following section the data provided by Trinos Vakuum-Systeme GmbH is specified.

5.2 Data

The job data provided by Trinos Vakuum-Systeme GmbH contains all important information that is necessary to solve the flexible job shop scheduling problem. In Table 5.1, the structure of the job data is specified. Each line corresponds to an operation i, which is dedicated to a certain job $J(i)$, a position in the job sequence $P(i)$ and a set of valid machines M_i. The processing times p_i assigned to the operations are based on empirical values. Additionally, the set of machines is subdivided in disjunctive machine groups $\mathcal{M}_j$ and each operation is associated with a certain machine group. This structure supports the approach chosen for the LP-based heuristic in Section 4.2.3. Finally, a deadline d_i for each job is given.

$J(i)$	$P(i)$	p_i	M_i	Machine Group $\mathcal{M}_j$	Deadline d_i
J_1	2	64	6,8,9,54,58	Manuell Drehen	30.09.2011
J_2	3	27	8,9	Manuell Drehen	15.01.2011
J_3	1	50	1,32,52,59	Zuschnitt	30.03.2012
J_4	4	280	22,34,55,62,63	Montage	29.02.2012
J_4	5	80	43,34,55,62,63	QS (Lecksuche, Messen)	29.02.2012
J_4	6	90	56,34,55,62,63	Werkstück Beschriften	29.02.2012
J_4	7	90	26,34,55,62,63	Versand	29.02.2012
J_5	2	138	19,20,64	Schweißen	31.12.2012
J_5	5	192	30,29,9,54,58	Strahlen	31.12.2012
J_5	6	190	28,29,9,54,58	Dichtflächen herstellen	31.12.2012

Table 5.1: Job data

The manufacturing facility of Trinos Vakuum-Systeme GmbH comprises 146 machines and stations and several hundreds of operations are included in the planning process. Consequently, due to the size of the actual scheduling problem of Trinos Vakuum-Systeme GmbH, an optimal solution by means of a mixed integer programming formulation of the flexible job shop scheduling problem is not achievable, see Section 3.7. Thus, the LP-based heuristic developed in Section 4.2.3 is applied to the scheduling problem.

5.3 Solution of the Scheduling Problem

In order to solve the scheduling problem with respect to the goals of Trinos Vakuum-System GmbH, first of all the objective function is adjusted. According to the deadlines, the objective is to minimize the sum of tardinesses given by

$$\min \sum_{j=1}^{n} \max \{0, C_j - d_j\}, \tag{5.1}$$

where C_j is the completion time of job j and d_j is the deadline for job j. The objective function (5.1) is then applied to the model STEF by changing constraints (4.9).

Furthermore, the production at Trinos Vakuum-Systeme GmbH is organized as a shift operation and each machine is either operated during all shifts of a day or only during a subset of the shifts of a day. In order to incorporate this constraint in the solving processes, the shifts are optimized consecutively. For each shift, the available machines are known and after a shift is optimized the data set is updated with regard to the operations

executed during the shift. If an operation is assigned to a machine and is not finished during the shift, the machine is set unavailable in the following shifts until the operation is finished.

By means of these adjustments, the LP-based heuristic presented in Section 4.2.3 is applicable for the scheduling problem of Trinos Vakuum-Systeme GmbH. In Figure 5.2 a schedule produced by the LP-based heuristic for a single shift from 07:00 a.m. to 03:00 p.m. is depicted in a machine-oriented Gantt chart. Only machines that are actually occupied are plotted and operations belonging to the same job are colored equally. Due to the information provided by the linear relaxation, the operations are scheduled with respect of a minimization of tardiness and simultaneously, a high occupancy rate is achieved.

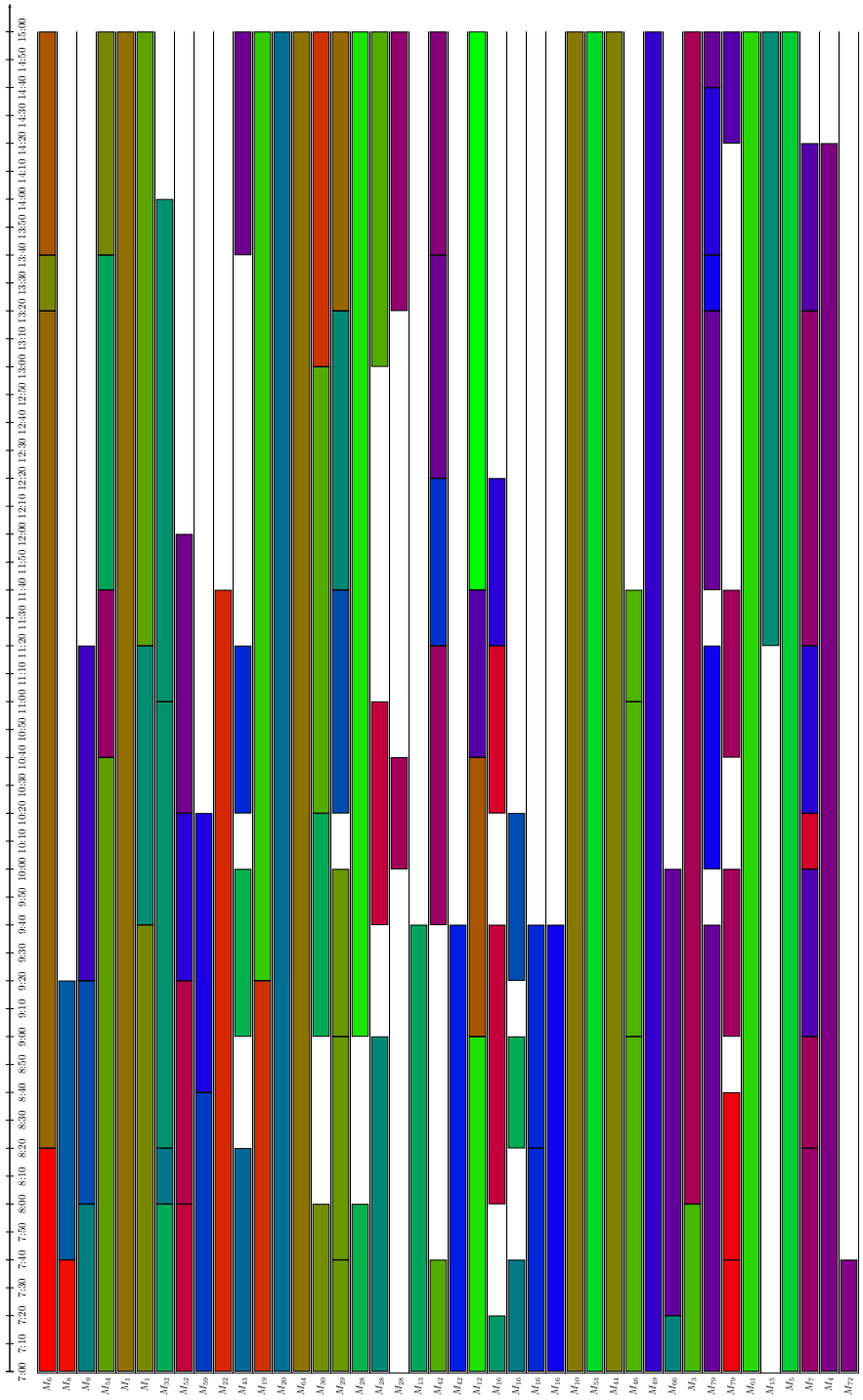


Figure 5.2: Machine-oriented Gantt chart for a schedule for an instance of Trinos Vakuum-Systeme GmbH produced by the LP-based heuristic

Chapter 6

Conclusion

In the course of this thesis, the flexible job shop scheduling problem is approached from three different angles. Thereby, each approach is chosen as a direct consequence of the results from the preceding approach. Initially, the problem is modeled and solved to optimality for instances up to a certain size. Due to the complexity, there is no efficient technique to solve arbitrary instances of the flexible job shop scheduling problem to optimality, unless $\mathcal{P} = \mathcal{NP}$. Consequently, approximation algorithms with provable performance-ratios are developed. However, the derived performance-ratios are not adequate for a practical application. Therefore, an efficient LP-based heuristic based on the initial models is designed in a final step. Each approach contributes its own share to the field of flexible job shop scheduling.

1. The modeling of the flexible job shop scheduling problem is intensively studied and four different models are developed. The models and several model extensions are evaluated and compared to each other with respect to size, structure and performance. In this way, the difficulties with respect to the modeling of the flexible job scheduling problems are exposed and preferable models are determined.

2. For the flexible job shop scheduling problem with identical processing times $FJ \,|\, p_{i,k} = p_i \,|\, C_{\max}$, two approximation algorithms with provable performance-ratio are developed. The first one is based on approximation algorithms for the job shop scheduling problem and the scheduling problem with restricted assignments and achieves a performance-ratio comparable to the best known performance-ratio for the job shop scheduling problem up to a constant factor. The second

approximation algorithm extends an existing approach employing a characterization of the convex hull of feasible solutions with linear inequalities.

3. Finally, an LP-based heuristic for the flexible job shop scheduling problem is presented and its efficiency and effectiveness for the solution of instances with practical relevance is established in computational studies.

Appendix A

Frequently Used Notation

	Jobs
J_i	Job i
n	Number of jobs
n_i	Number of operations of job i
d_i	Due date of job i
$P_{\max}$	Maximum job length
	Operations
O	Set of operations
$O_{i,j}$	Operation j of job i
O_i	Operation i
N	Number of operations
μ	Maximal number of operations per job
$J(i)$	Job of operation i
$P(i)$	Position of operation i in its job $J(i)$
p_i	Processing time of operation i
$p_{\max}$	Maximum processing time
I_k	Operations processable on machine k
	Machines
M	Set of machines
μ_i	Machine i

m	Number of machines
M_i	Set of valid machines for operation i
$\mathcal{M}_i$	Machine group i
$\Pi_{\max}$	Maximum machine load
	Schedules
S	Schedule
S_i	Starting time of operation i
C_i	Completion time of job i
$C_{\max}$	Makespan
T	Time period
α_i	Earliest possible starting time for operation i
β_i	Latest possible starting time for operation i
C	Set of conjunctions
D	Set of disjunctions
x	IPF, GPF: Machine assignment
	TEF: Machine and starting time assignment
	MPF: Machine and position assignment
y	IPF, GPF: Sequencing variables

Bibliography

[Ake56] Sheldon B. Akers Jr. A Graphical Approach to Production Scheduling Problems. *Operations Research*, 4(2):244–245, December 1956.

[BJK97] Peter Brucker, Bernd Jurisch, and Andreas Krämer. Complexity of scheduling problems with multi-purpose machines. *Annals of Operations Research*, 70:57 – 73, 1997.

[BK06] Peter Brucker and Sigrid Knust. *Complex Scheduling*. GOR Publications. Springer, 2006.

[Bra93] Paolo Brandimarte. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, 41(3):157–183, 1993.

[Bru04] Peter Brucker. *Scheduling Algorithms*. Springer, fourth edition, 2004.

[BS90] Peter Brucker and R. Schlie. Job-Shop Scheduling with Multi-Purpose Machines. *Computing*, 45:369–375, January 1990.

[CC02] In-Chan Choi and Dae-Sik Choi. A local search algorithm for jobshop scheduling problems with alternative operations and sequence-dependent setups. *Computers & Industrial Engineering*, 42:43–58, April 2002.

[CIL99] Haoxun Chen, Jürgen Ihlow, and Carsten Lehmann. A Genetic Algorithm for Flexible Job-Shop Scheduling. *Proceedings of the 1999 IEEE International Conference on Robotics & Automation*, pages 1120–1125, 1999.

[cpl10] IBM Corporation. IBM ILOG CPLEX Optimizer Data Sheet. `http://public.dhe.ibm.com/common/ssi/ecm/en/wsd14044usen/WSD14044USEN.PDF`, 2010. Date accessed: 12 February 2012.

[Dav91] L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.

[DFJ54] G. Dantzig, R. Fulkerson, and David S. Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the Operational Research Society*, 2:393–410, 1954.

[EAS+11] Abd Elazeem, Mohamed Abd, Mohamed Sayed, Ali Osman, and Mohamed Bayoumi. Optimality of the flexible job shop scheduling problem. *Science*, 4(10):321–328, 2011.

[FSMJ07] Parviz Fattahi, Mohammad Saidi-Mehrabad, and Fariborz Jolai. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of Intelligent Manufacturing*, 18(3):331–342, 2007.

[FYL+08] Mingyue Feng, Xianqing Yi, Guohui Li, Shaoxun Tang, and He Jun. A Grouping Particle Swarm Optimization Algorithm for Flexible Job Shop Scheduling Problem. In *Computational Intelligence and Industrial Application 2008 PACIIA 08 PacificAsia Workshop on*, volume 1, pages 332–336. Ieee, 2008.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., 1979.

[GJ09] B. S. Girish and N. Jawahar. A particle swarm optimization algorithm for flexible job shop scheduling problem. *Automation Science and Engineering 2009 CASE 2009 IEEE International Conference on*, 2(4):298–303, 2009.

[GJS76] Michael R. Garey, David S. Johnson, and Ravi Sethi. The Complexity of Flowshop and Jobshop Scheduling. *Mathematics of Operations Research*, 1(2):117–129, 1976.

[GK06] Celia A. Glass and Hans Kellerer. Parallel Machine Scheduling with Job Assignment Restrictions. *Naval Research Logistics*, 54(3):250–257, 2006.

[GLLR79] R. L. Graham, E. L. Lawler, Jan Karel Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5(2):287–326, 1979.

[GLMM04] Martin Gairing, Thomas Lücking, Marios Mavronicolas, and Burkhard Monien. Computing Nash equilibria for scheduling on restricted parallel links. *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing - STOC '04*, page 613, 2004.

[Glo89] Fred Glover. Tabu Search - Part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[GLS88] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization.* Springer, August 1988.

[Gol80] Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs*, volume 57 of *Annals of Discrete Mathematics.* Academic Press, Inc. (London) Ltd., 1980.

[GPSS01] Leslie Ann Goldberg, Mike Paterson, Aravind Srinivasan, and Elizabeth Sweedyk. Better Approximation Guarantees for Job-Shop Scheduling. *SIAM Journal on Discrete Mathematics*, 14(1):67–92, 2001.

[GSG08] Jie Gao, Linyan Sun, and Mitsuo Gen. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. *Computers & Operations Research*, 35(9):2892–2907, 2008.

[Hol75] J. H. Holland. *Adaptation in Natural and Artificial Systems*, volume Ann Arbor. University of Michigan Press, 1975.

[HS71] Akihiro Hashimoto and James Stevens. Wire Routing by Optimizing Channel Assignment within Large Apertures. *DAC '71*

Proceedings of the 8th Design Automation Workshop, pages 155–169, 1971.

[Jur92] Bernd Jurisch. *Scheduling Jobs in Shops with Multi-Purpose Machines*. PhD thesis, 1992.

[Kal02] Josef Kallrath. *Gemischt-ganzzahlige Optimierung: Modellierung in der Praxis : mit Fallstudien aus Chemie, Energiewirtschaft, Metallgewerbe, Produktion und Logistik*. Vieweg, 2002.

[Kar84] N. Karmarkar. A New Polynomial-Time Algorithm for Linear Programming. *Combinatorica*, 4(4):373–395, 1984.

[KE95] J. Kennedy and Russell Eberhart. Particle swarm optimization. *PLoS ONE*, 4(6):1942–1948, 1995.

[KV08] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Algorithms and Combinatorics. Springer, 2008.

[MG00] Monaldo Mastrolilli and Luca Maria Gambardella. Effective neighbourhood functions for the flexible job shop problem. *Journal of Scheduling*, 3(1):3–20, January 2000.

[NW88] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*, volume 41 of *Series in Discrete Mathematics and Optimization*. Wiley-Interscience, 1988.

[PMC08] F. Pezzella, G. Morganti, and G. Ciaschetti. A genetic algorithm for the Flexible Job-shop Scheduling Problem. *Computers & Operations Research*, 35(10):3202–3212, 2008.

[Sch96] Andreas S. Schulz. *Polytopes and Scheduling*. PhD thesis, 1996.

[SE98] Yuhui Shi and Russell Eberhart. A Modified Particle Swarm Optimizer. *Applied Mathematics and Computation*, 189(5):69–73, 1998.

[SMF06] Mohammad Saidi-Mehrabad and Parviz Fattahi. Flexible job shop scheduling with tabu search algorithms. *The Interna-*

tional Journal of Advanced Manufacturing Technology, 32:563–570, May 2006.

[SS95] Yuri N. Sotskov and N. V. Shakhlevich. NP-hardness of shop-scheduling problems with three jobs. *Discrete Applied Mathematics*, 59:237–266, 1995.

[SSW94] David B. Shmoys, Clifford Stein, and Joel Wein. Improved Approximation Algorithms for Shop Scheduling Problems. *SIAM Journal on Computing*, 23:617–632, 1994.

[ZGS11] Guohui Zhang, Liang Gao, and Yang Shi. An effective genetic algorithm for the flexible job-shop scheduling problem. *Expert Systems with Applications*, 38(4):3563–3573, 2011.

[ZSLG09] Guohui Zhang, Xinyu Shao, Peigen Li, and Liang Gao. An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers & Industrial Engineering*, 56(4):1309–1318, May 2009.

Declaration of Academic Honesty

I hereby declare that this master thesis has been written only by the undersigned and without any assistance from third parties. Furthermore, I confirm that no sources have been used in the preparation of this thesis other than those indicated in the thesis itself.

Göttingen, 14 March 2012 _____

(Morten Tiedemann)