HELMUT-SCHMIDT-UNIVERSITÄT
UNIVERSITÄT DER BUNDESWEHR HAMBURG
LEHRSTUHL FÜR BETRIEBSWIRTSCHAFTSLEHRE,
INSBES. LOGISTIK-MANAGEMENT
Prof. Dr. M. J. Geiger

Arbeitspapier / Research Report
RR-12-01-01 · January 2012 · ISSN 2192-0826

Test Instances for the Flexible Job Shop Scheduling Problem with Work Centers

Dennis Behnke[1,*] and Martin Josef Geiger[1]

[1]Helmut-Schmidt-University, Logistics Management Department, Hamburg, Germany.
*Corresponding author: dbehnke@hsu-hh.de

Contents

Abstract

The flexible job shop scheduling problem (FJSSP) is a generalization and extension of the classical job shop scheduling problem (JSSP) in which — prior to the sequencing of operations — an assignment of operations to machines is necessary. In this technical report, we are examining common instances for different specifications of the FJSSP. Furthermore, a new FJSSP specification is introduced, where similar machines are pooled to work centers, and the first and last work center are obligatory. For this practice-oriented problem specification new test instances are presented.

KEYWORDS: flexible job shop scheduling; work centers; instances; similar machines, flexibility, multi-purpose machines, flexible manufacturing systems, machine routing, scheduling practice

1 Introduction

A central assumption in classical job shop scheduling (JSSP) is that every operation has to be processed on one predetermined machine. The actual relevance of recent flexible job shop scheduling (FJSSP) approaches has been lying in the fact that in practice, there is often more than one machine that is able to process a particular manufacturing task. The flexible job shop scheduling problem, introduced by Brandimarte in 1993, accordingly extended and generalized the classical job shop scheduling problem such that for every operation there would be more than one possible machine assignment. Since then, several scientists have presented new FJSSP specifications and test instances that have caught broader attention: Initially, Brandimarte [7] introduced instances for the general FJSSP with varying degrees of production versatility of the machines. Later, Hurink et al., and Chambers and Barnes generated instances with the very special property that the processing times of the operations are independent of the assigned machines [10, 29]. Eventually, Kacem et al. presented a new problem specification with total flexibility where every operation can be processed on any one of the machines [35].
In our research, we have been striving to supplement the existing problem specifications and instances by additionally modeling "similarity" among the machines in a realistic way by grouping them to "work centers", which can frequently be observed in industrial practice [3, 45, 48, 51], but have not yet been considered in the existing FJSSP approaches [21,26,39]. A work center[1] can be described as a pool of production resources (workers, machines, and equipment) dedicated to fulfill specific processing tasks [48], e. g. pressing, welding, turning, or milling. Furthermore, a restricted (flexible) flow shop character is introduced by setting both the first and last work center obligatory, a situation often encountered in practice in form of preparing or closing procedures (cleaning, deburring, polishing). In this technical report, we provide a survey of common instances for the various FJSSP specifications and introduce new instances for the FJSSP with work centers (FJSSPWC).

[1]Sometimes also referred to as "load centers", "work stations", "work stages", "machine sets" or "machine pools" [45, 48, 51].

The rest of this article is organized as follows. In the second section, a problem description of the FJSSP is provided. Common test instances from the literature are examined in section 3. The design and generation of new instances for the FJSSPWC is presented in section 4. The article closes with a short conclusion.

2 Problem description

2.1 Problem definiton

In the classical JSSP (cf. table 1), a set $\mathcal{J} = \{J_1, ..., J_n\}$ of n jobs is given. Each job J_i consists of a fixed sequence of operations $\mathcal{O}_i = \{O_{i,1}, O_{i,2}, ...O_{i,h_i}\}$ and each operation $O_{i,k}$ has to be processed on a predetermined machine $M_{i,k} \in \mathcal{M}$, $\mathcal{M} = \{M_1, ..., M_m\}$ with $p_{i,k}$ being the necessary processing time and $\mathcal{O}$ being the set of all operations.
The FJSSP is an extension and generalization of the job shop scheduling problem such that each operation $O_{i,k}$ can be processed either on a subset $\mathcal{M}_{i,k} \subseteq \mathcal{M}$ ("partial flexibility", cf. table 2) or on all machines ($\mathcal{M}_{i,k} = \mathcal{M}$, "total flexibility", cf. tab 3). Therefore, prior to the *scheduling* of the operations, i. e. determining a starting time for each operation while taking into account precedence constraints and machine availability, an *assignment* of the operations to the machines is to be undertaken (sometimes referred to as 'routing', cf. [7, 11]).
As an extension and generalization of the job shop scheduling problem, the flexible job shop scheduling problem is equally known to be $\mathcal{NP}$-hard. Note that once the assignments are determined, the FJSSP turns into the classical JSSP. Common solution approaches for the *JSSP* include disjunctive programming and branch-and-bound algorithms as exact procedures, as well as shifting-bottleneck heuristics and local search meta-heuristics as heuristic procedures [45]. For the *FJSSP*, depending on whether the assignment and sequencing sub-problems are dealt with subsequently or jointly, *hierarchical* and *integrated* approaches can be distinguished. Among the heuristic approaches using neighborhood functions, particularly tabu search heuristics [7, 10, 11, 17, 29, 41], evolutionary algorithms [13, 21, 22, 25, 34, 35, 44, 49], as well as variable neighborhood search procedures [4, 22, 54] have proven to be effective.

2.2 Related scheduling problems

The challenge in scheduling theory and practice is to organize the processing of a set of (manufacturing) tasks by a limited number of production resources (machines). Depending on the number and properties of the tasks, resources, and objectives, there exists a large variety of deduced scheduling problems, ranging from rather easy-to-solve makespan minimizing single machine problems to particularly challenging multicriteria and flexible scheduling problems. Conway et al. introduced a notation in order to classify this variety of scheduling problems [15, 45] by its machine characteristics (denoted by α), its order properties (denoted by β), and the objectives to consider (denoted by γ). With respect to the machine properties (α) a natural extension of the classical single machine scheduling problems is to multiply the production resources such that

$p_{i,k}$		M_1	M_j	...	M_m
J_1	$O_{1,1}$	∞	∞	...	$p_{1,1}$
	$O_{1,2}$	$p_{1,2}$	∞	...	∞
	...	...	...	...	...
	O_{1,h_1}	∞	p_{1,h_1}	...	∞
J_i	$O_{i,1}$	∞	$p_{i,1}$	...	∞
	$O_{i,2}$	$p_{i,2}$	∞	...	∞
	...	...	...	...	...
	O_{i,h_i}	∞	∞	...	p_{i,h_i}
...		...	...	...	...
J_n	$O_{n,1}$	$p_{n,1}$	∞	...	∞
	$O_{n,2}$	∞	∞	...	$p_{n,2}$
	...	...	...	...	...
	O_{n,h_n}	∞	p_{n,h_n}	...	∞

Table 1: Assignment structure of the job shop scheduling problem

$p_{i,k,j}$		M_1	M_2	M_j	M_{j+1}	...	M_{m-1}	M_m
J_1	$O_{1,1}$	∞	$p_{1,1,2}$	$p_{1,1,j}$	$p_{1,1,j+1}$	...	$p_{1,1,m-1}$	$p_{1,1,m}$
	$O_{1,2}$	$p_{1,2,1}$	$p_{1,2,2}$	$p_{1,2,j}$	∞	...	$p_{1,2,m-1}$	$p_{1,2,m}$
	...	...	...	...	...	...	...	...
	O_{1,h_1}	$p_{1,h_1,1}$	$p_{1,h_1,2}$	$p_{1,h_1,j}$	$p_{1,h_1,j+1}$	...	∞	$p_{1,h_1,m}$
J_i	$O_{i,1}$	$p_{i,1,1}$	∞	$p_{i,1,j}$	$p_{i,1,j+1}$	...	$p_{i,1,m-1}$	$p_{i,1,m}$
	$O_{i,2}$	$p_{i,2,1}$	$p_{i,2,2}$	∞	$p_{i,2,j+1}$	...	$p_{i,2,m-1}$	$p_{i,2,m}$
	...	...	...	...	...	...	...	...
	O_{i,h_i}	∞	$p_{i,h_i,2}$	$p_{i,h_i,j}$	$p_{i,h_i,j+1}$	...	$p_{i,h_i,m-1}$	$p_{i,h_i,m}$
...		...	...	...	...	...	...	...
J_n	$O_{n,1}$	$p_{n,1,1}$	$p_{n,1,2}$	$p_{n,1,j}$	$p_{n,1,j+1}$	...	∞	$p_{n,1,m}$
	$O_{n,2}$	$p_{n,2,1}$	$p_{n,2,2}$	$p_{n,2,j}$	$p_{n,2,j+1}$	...	$p_{n,2,m-1}$	∞
	...	...	...	...	...	...	...	...
	O_{n,h_n}	$p_{n,h_n,1}$	$p_{n,h_n,2}$	∞	$p_{n,h_n,j+1}$	...	$p_{n,h_n,m-1}$	$p_{n,h_n,m}$

Table 2: Assignment structure of the flexible job shop scheduling problem with partial flexibility

$p_{i,k,j}$		M_1	M_2	M_j	M_{j+1}	...	M_{m-1}	M_m
J_1	$O_{1,1}$	$p_{1,1,1}$	$p_{1,1,2}$	$p_{1,1,j}$	$p_{1,1,j+1}$	...	$p_{1,1,m-1}$	$p_{1,1,m}$
	$O_{1,2}$	$p_{1,2,1}$	$p_{1,2,2}$	$p_{1,2,j}$	$p_{1,2,j+1}$	...	$p_{1,2,m-1}$	$p_{1,2,m}$
	...	...	...	...	...	...	...	...
	O_{1,h_1}	$p_{1,h_1,1}$	$p_{1,h_1,2}$	$p_{1,h_1,j}$	$p_{1,h_1,j+1}$	...	$p_{1,h_1,m-1}$	$p_{1,h_1,m}$
J_i	$O_{i,1}$	$p_{i,1,1}$	$p_{i,1,2}$	$p_{i,1,j}$	$p_{i,1,j+1}$	...	$p_{i,1,m-1}$	$p_{i,1,m}$
	$O_{i,2}$	$p_{i,2,1}$	$p_{i,2,2}$	$p_{i,2,j}$	$p_{i,2,j+1}$	...	$p_{i,2,m-1}$	$p_{i,2,m}$
	...	...	...	...	...	...	...	...
	O_{i,h_i}	$p_{i,h_i,1}$	$p_{i,h_i,2}$	$p_{i,h_i,j}$	$p_{i,h_i,j+1}$	...	$p_{i,h_i,m-1}$	$p_{i,h_i,m}$
...		...	...	...	...	...	...	...
J_n	$O_{n,1}$	$p_{n,1,1}$	$p_{n,1,2}$	$p_{n,1,j}$	$p_{n,1,j+1}$	...	$p_{n,1,m-1}$	$p_{n,1,m}$
	$O_{n,2}$	$p_{n,2,1}$	$p_{n,2,2}$	$p_{n,2,j}$	$p_{n,2,j+1}$	...	$p_{n,2,m-1}$	$p_{n,2,m}$
	...	...	...	...	...	...	...	...
	O_{n,h_n}	$p_{n,h_n,1}$	$p_{n,h_n,2}$	$p_{n,h_n,j}$	$p_{n,h_n,j+1}$	...	$p_{n,h_n,m-1}$	$p_{n,h_n,m}$

Table 3: Assignment structure of the flexible job shop scheduling problem with total flexibility

several identical ($\alpha = Pm$) or similar machines are considered ($\alpha = Qm, Rm$). Furthermore, the orders to fulfill can be modeled in a more detailed way such that instead of "tasks", "jobs" are considered, each of which consists of "operations" in a particular precedence order (JSSP, $\beta = Jm$). In a flow shop scheduling problem (FSSP), on the one hand, these operations additionally have to obey the same machine order for every job ($\beta = Fm$). In an open shop scheduling problem (OSSP), on the other hand, there are no precedence constraints at all ($\beta = Om$), whereas in a Process Plan Selection Job Shop Scheduling Problems (PPSS), the processing order can be selected from a limited set of possible options [8, 37]. In the flexible job shop problem we are dealing with in this article ($\beta = FJc$), the precedence constraints among the operations are fixed. However, an assignment of the operations to the production resources has to be executed before an actual scheduling of the operations can be achieved. This is also the case in flexible flow shop problems (FFSSP) where the "processing orders" are fixed ($\beta = FFc$): In contrast to classical flow shops, however, the operations have to pass machine clusters ("stages") instead of single machines. Therefore, the FJSSP and the FFSSP are extensions and generalizations of the JSSP, and the FSSP respectively. On the other hand, JSSP and FJSSP are generalizations and extensions of the FSSP, and FFSSP respectively, because the latter are special cases of the first. Although the FJSSP thus allows for a modeling of many real-life industrial scheduling problems, two central assumptions are still being made in many solution approaches: Firstly, the operations are inseparable such that every processing of a particular operation by more than one machine is not possible. Secondly, the machines are working in a non-preemptive way, i. e. no interruption in the manufacturing process of an operation in order to process another operation in-between is allowed.
For some special applications of the FJSSP, problem specific instances for the FJSSP have led to different optimization challenges in terms of neighborhoods to define, or metaheuristics to deploy. Accordingly, these application-oriented FJSSP specifications have been given different names. In the job shop scheduling problem with multi-purpose machines (MPM-JSSP), each operation can be processed on a set of different machines, each of which is therefore considered to be of a "multi-purpose" nature [29]. Note that in MPM-JSSP, the processing times for the operations do not depend upon which machine has been chosen. In the scheduling of a flexible manufacturing system (FMS), several technological constraints – e. g. an overall cap of jobs in process – are often taken additionally into account [43]. Due to the fact that – apart from additional constraints or objectives to consider – these problem specifications primarily differ in the way the instances have been generated, they may all be categorized as flexible job shop scheduling problems.

3 Common instances for the FJSSP

3.1 Overview

Among the instances used in the various approaches to the FJSSP, the ones by Brandimarte (1993), Hurink et al. (1994), Dauzère-Pérès and Paulli (1994), Chambers and Barnes (1996), Kacem et al. (2002), and Fattahi et al. (2007) are the ones most often used for computational evaluations. This is probably due to the fact that most of them are available via the OR library [6, 40], the ones by Kacem et al. and Fattahi et al. being exceptions. In the following subsections, we present these six approaches briefly and provide the data for the smallest instance of each approach. Furthermore, lower bounds (LB) and best known upper bounds (UB) from the literature for the minimization of the makespan are provided. In some cases, where no lower bounds from the literature are available, they have been calculated according to [27]. Eventually, all instances have been run by the ILOG Constraint Programming (CP) engine [30–32] for 10 minutes. Note that in 16 cases, the best known upper bounds from the literature have been improved by these new CP solutions. An asterisk behind the UB values indicates that these solutions have been proven to be optimal. Accordingly, the quoted authors are not those who obtained the given solution first, but those who additionally confirmed its optimality.
Other test instances are designed according to similar designing principles and neither widely used nor publicly available, e. g. [13, 42, 49, 56] or generated with respect to particular objectives or restrictions, e. g. [4, 7, 12, 47, 50, 52]. For our purposes, the most important fact is, however, that although the pooling of machines to work centers is recognized as being of practical importance, this has not been implemented in the common test instances: Often, the similarity of machines is merely modeled by a varying degree of flexibility [21, 45, 50].

*

3.2 Instances for the general FJSSP by Brandimarte

Brandimarte introduced the general FJSSP and provided a set of 15 problem instances with medium flexibility [7]. The parameters of the instances were set as follows (cf. table 4):

Instance	n	m	h_i	$\lvert\mathcal{M}_{i,k}\rvert$	$p_{i,k,j}$	LB	UB	CP
mk1	10	6	5...7	3	1...7	36 [41]	39 [53]	40* (!)
mk2	10	6	5...7	6	1...7	24	26 [41]	27
mk3	15	8	10	5	1...20	204 [41]	204* [41]	204*
mk4	15	8	3...10	3	1...10	48 [41]	60* [41]	60
mk5	15	4	5...10	2	5...10	168	172 [22]	174
mk6	10	15	15	5	1...10	33	58 [41]	59
mk7	20	5	5	5	1...20	133 [41]	139 [44]	143
mk8	20	10	5...15†	2	5...20	523 [41]	523* [41]	523*
mk9	20	10	10...15	5	5...20	299 [41]	307 [41]	307*
mk10	20	15	10...15	5	5...20	165 [41]	197 [22]	214
mk11‡	30	5	5...8	2	10...30	594	649 [7]	615
mk12‡	30	10	5...10	2	10...30	320	518 [7]	508*
mk13‡	30	10	5...10	5	10...30	353	478 [7]	430
mk14‡	30	15	8...12	2	10...30	334	694 [7]	694*
mk15‡	30	15	8...12	5	10...30	283	383 [7]	341

Table 4: Instances by Brandimarte [7] († There is a mistake in the original article here (saying "10...5" [7]), ‡ not available via [40])

$\lvert\mathcal{M}_{i,k}\rvert$ indicates the maximum number of assignable machines per operation (flexibility level) with 1 being the minimum number of assignable machines. The processing times were generating by a uniform distribution within the given limits. Note that test instances "mk10" to "mk15" are not available via the FJSSP instance collection [40]. Table 5 provides the data of the first instance "mk1".

$p_{i,k,j}$		M_1	M_2	M_3	M_4	M_5	M_6
J_1	$O_{1,1}$	5	∞	4	∞	∞	∞
	$O_{1,2}$	∞	1	5	∞	3	∞
	$O_{1,3}$	∞	∞	4	∞	∞	2
	$O_{1,4}$	1	6	∞	∞	∞	5
	$O_{1,5}$	∞	∞	1	∞	∞	∞
	$O_{1,6}$	∞	∞	6	3	∞	6
J_2	$O_{2,1}$	∞	6	∞	∞	∞	∞
	$O_{2,2}$	∞	∞	1	∞	∞	∞
	$O_{2,3}$	2	∞	∞	∞	∞	∞
	$O_{2,4}$	∞	6	∞	6	∞	∞
	$O_{2,5}$	1	∞	∞	∞	∞	5
J_3	$O_{3,1}$	∞	6	∞	∞	∞	∞
	$O_{3,2}$	∞	∞	4	∞	∞	2
	$O_{3,3}$	1	6	∞	∞	∞	5
	$O_{3,4}$	∞	6	4	∞	∞	6
	$O_{3,5}$	1	∞	∞	∞	5	∞
J_4	$O_{4,1}$	1	6	∞	∞	∞	5
	$O_{4,2}$	∞	6	∞	∞	∞	∞
	$O_{4,3}$	∞	∞	1	∞	∞	∞
	$O_{4,4}$	∞	1	5	∞	3	∞
	$O_{4,5}$	∞	∞	4	∞	∞	2
J_5	$O_{5,1}$	∞	1	5	∞	3	∞
	$O_{5,2}$	1	6	∞	∞	∞	5
	$O_{5,3}$	∞	6	∞	∞	∞	∞
	$O_{5,4}$	5	∞	4	∞	∞	∞
	$O_{5,5}$	∞	6	∞	6	∞	∞
	$O_{5,6}$	∞	6	4	∞	∞	6
J_6	$O_{6,1}$	∞	∞	4	∞	∞	2
	$O_{6,2}$	2	∞	∞	∞	∞	∞
	$O_{6,3}$	∞	6	4	∞	∞	6
	$O_{6,4}$	∞	6	∞	∞	∞	∞
	$O_{6,5}$	1	6	∞	∞	∞	5
	$O_{6,6}$	3	∞	∞	2	∞	∞
J_7	$O_{7,1}$	∞	∞	∞	∞	∞	1
	$O_{7,2}$	3	∞	∞	2	∞	∞
	$O_{7,3}$	∞	6	4	∞	∞	6
	$O_{7,4}$	6	6	∞	∞	1	∞
	$O_{7,5}$	∞	∞	1	∞	∞	∞
J_8	$O_{8,1}$	∞	∞	4	∞	∞	2
	$O_{8,2}$	∞	6	4	∞	∞	6
	$O_{8,3}$	1	6	∞	∞	∞	5
	$O_{8,4}$	∞	6	∞	∞	∞	∞
	$O_{8,5}$	∞	6	∞	6	∞	∞
J_9	$O_{9,1}$	∞	∞	∞	∞	∞	1
	$O_{9,2}$	1	∞	∞	∞	5	∞
	$O_{9,3}$	∞	∞	6	3	∞	6
	$O_{9,4}$	2	∞	∞	∞	∞	∞
	$O_{9,5}$	∞	6	4	∞	∞	6
	$O_{9,6}$	∞	6	∞	6	∞	∞
J_{10}	$O_{10,1}$	∞	∞	4	∞	∞	2
	$O_{10,2}$	∞	6	4	∞	∞	6
	$O_{10,3}$	∞	1	5	∞	3	∞
	$O_{10,4}$	∞	∞	∞	∞	∞	1
	$O_{10,5}$	∞	6	∞	6	∞	∞
	$O_{10,6}$	3	∞	∞	2	∞	∞

Table 5: Instance "mk1" by Brandimarte [40]

3.3 MPM-JSSP-instances by Hurink et al.

Hurink et al. adapted some instances from classical JSSPs [29]: The instances "mt06", "mt10", and "mt20" in table 6 are originally instances of Fisher and Thompson [20]. The 40 instances "la01" to "la40" are instances generated by Lawrence [38]. It holds for every instance that the number of operations per job $h_i = h$ exactly equal the total number of machines m of the respective instance setting.

Instance	n	m
mt06†	6	6
mt10†	10	10
mt20†	20	5
la01...la05‡	10	5
la06...la10‡	15	5
la11...la15‡	20	5
la16...la20‡	10	10
la21...la25‡	15	10
la26...la30‡	20	10
la31...la35‡	30	10
la36...la40‡	15	15

Table 6: Classical JSSP instances as a basis of the "sdata"-instances by Hurink et al. [29] († from [20], ‡ from [38])

Hurink et al. have adopted the original versions of these instances where every operation is assignable to exactly one particular machine as "sdata". In order to modify these instances to suit the MPM problem specification, the data are transformed to three different instance sets "edata", "rdata", and "vdata" by enlarging the respective set of assignable machines with a particular probability distribution [40]:

1. edata: Few operations may be assigned to more than one machine.
2. rdata: Most of the operations may be assigned to some machines.
3. vdata: All operations may be assigned to several machines.

The resulting properties are provided in table 7 with avrg$|\mathcal{M}_{i,k}|$ being the average number of assignable machines per operation, and max$|\mathcal{M}_{i,k}|$ being the maximum number of assignable machines per operation. Note that due to the multi-purpose-machine specification, the processing times for each operation are constant among the assignable machines. The data of the first instance "mt06" is provided in table 9. Table 8 contains the best known results concerning the minimization of C_{max}.

sdata	edata		rdata		vdata	
avrg$\|\mathcal{M}_{i,k}\|$ = max$\|\mathcal{M}_{i,k}\|$	avrg$\|\mathcal{M}_{i,k}\|$	max$\|\mathcal{M}_{i,k}\|$	avrg$\|\mathcal{M}_{i,k}\|$	max$\|\mathcal{M}_{i,k}\|$	avrg$\|\mathcal{M}_{i,k}\|$	max$\|\mathcal{M}_{i,k}\|$
1	1.15	2(if m≤6) 3(if m≥10)	2	3	$\frac{1}{2}m$	$\frac{4}{5}m$

Table 7: Modification of the sdata instances by Hurink et al. [29]

Inst.	edata			rdata			vdata		
	LB	UB	CP	LB	UB	CP	LB	UB	CP
m06	55	55* [33]	55*	47	47* [33]	47*	47	47* [33]	47*
m10	871	871* [33]	877	679	686 [17]	686*	655	655* [33]	655*
m20	1088	1088* [33]	1088*	1022	1022* [17]	1024	1022	1022* [33]	1023
la1	609	609* [33]	609*	570	571 [33]	573	570	570* [33]	570
la2	655	655* [33]	655*	529	530 [33]	534	529	529* [33]	529
la3	550	550* [33]	567	477	478 [33]	478	477	477* [41]	478
la4	568	568* [33]	568	502	502* [33]	504	502	502* [33]	502
la5	503	503* [33]	503*	457	457* [33]	458	457	457 [41]	458
la6	833	833* [33]	833*	799	799* [17]	799	799	799* [33]	799
la7	762	762* [33]	765	749	750 [33]	750	749	749* [17]	750
la8	845	845* [33]	845*	765	765* [41]	766	765	765* [17]	766
la9	878	878* [33]	878	853	853* [17]	854	853	853* [17]	854
la10	866	866* [33]	866	804	804* [17]	805	804	804* [33]	804
la11	1087	1103 [33]	1106	1071	1071* [33]	1072	1071	1071* [33]	1071
la12	960	960* [33]	960*	936	936* [17]	936	936	936* [33]	936
la13	1053	1053* [33]	1053*	1038	1038* [33]	1038	1038	1038* [33]	1038
la14	1123	1123* [33]	1123	1070	1070* [33]	1071	1070	1070* [17]	1070
la15	1111	1111* [33]	1111*	1089	1090 [17]	1091	1089	1089* [17]	1090
la16	892	892* [33]	904	717	717* [33]	717*	717	717* [33]	717*
la17	707	707* [33]	707	646	646* [33]	646*	646	646* [33]	646*
la18	842	842* [33]	843	666	666* [33]	666*	663	663* [33]	663*
la19	796	796* [33]	799	647	700 [41]	703	617	617* [33]	617
la20	857	857* [33]	857	756	756* [33]	757* (!)	756	756* [33]	756*
la21	895	1017 [41]	1044	808	835 [41]	845	800	806 [41]	804
la22	832	882 [33]	887	737	760 [41]	775	733	739 [41]	736
la23	950	950* [41]	950*	816	842 [41]	857	809	815 [41]	815
la24	881	909 [41]	913	775	808 [41]	818	773	777 [41]	775
la25	894	941 [41]	955	752	791 [41]	805	751	756 [41]	756
la26	1089	1125 [41]	1143	1056	1061 [41]	1074	1052	1054 [41]	1054
la27	1181	1186 [41]	1188	1085	1091 [41]	1101	1084	1085 [41]	1084
la28	1116	1149 [41]	1153	1075	1080 [41]	1084	1069	1070 [41]	1070
la29	1058	1118 [41]	1128	993	998 [41]	1006	993	994 [41]	995
la30	1147	1204 [41]	1241	1068	1078 [41]	1087	1068	1069 [41]	1072
la31	1523	1539 [41]	1552	1520	1521 [41]	1525	1520	1520* [41]	1522
la32	1698	1698* [16]	1698*	1657	1659 [41]	1664	1657	1658 [33]	1661
la33	1547	1547* [33]	1560	1497	1499 [41]	1502	1497	1497* [41]	1500
la34	1592	1604 [33]	1609	1535	1536 [41]	1542	1535	1535* [41]	1537
la35	1736	1736* [33]	1736*	1549	1550 [41]	1556	1549	1549* [41]	1551
la36	1006	1162 [41]	1160	1016	1030 [17]	1034	948	948* [33]	948*
la37	1355	1397 [41]	1397*	989	1077 [41]	1084	986	986* [33]	986*
la38	1019	1144 [41]	1146	943	962 [41]	973	943	943* [33]	943*
la39	1151	1184 [41]	1184	966	1024 [17]	1018	922	922* [33]	922*
la40	1034	1150 [17]	1174	955	970 [41]	984	955	955* [33]	955*

Table 8: Best known LB and UB, as well as CP solutions for the instances of Hurink et al. [29] (all LB from [33])

$p_{i,k,j}$		M_1	M_2	M_3	M_4	M_5	M_6
J_1	$O_{1,1}$	∞	∞	1	∞	∞	∞
	$O_{1,2}$	3	∞	∞	∞	∞	∞
	$O_{1,3}$	∞	6	∞	∞	∞	∞
	$O_{1,4}$	∞	∞	∞	7	∞	∞
	$O_{1,5}$	∞	∞	∞	3	∞	3
	$O_{1,6}$	∞	∞	∞	∞	6	∞
J_2	$O_{2,1}$	∞	8	∞	∞	∞	∞
	$O_{2,2}$	∞	∞	5	∞	∞	∞
	$O_{2,3}$	∞	∞	∞	∞	10	∞
	$O_{2,4}$	∞	∞	∞	∞	∞	10
	$O_{2,5}$	10	∞	∞	∞	∞	∞
	$O_{2,6}$	∞	∞	∞	4	∞	∞
J_3	$O_{3,1}$	∞	∞	5	∞	∞	5
	$O_{3,2}$	∞	∞	∞	4	∞	∞
	$O_{3,3}$	∞	∞	∞	∞	∞	8
	$O_{3,4}$	9	∞	∞	∞	∞	∞
	$O_{3,5}$	∞	1	∞	∞	∞	1
	$O_{3,6}$	∞	∞	∞	∞	7	∞
J_4	$O_{4,1}$	∞	5	∞	∞	∞	∞
	$O_{4,2}$	5	∞	∞	∞	∞	∞
	$O_{4,3}$	∞	∞	5	∞	∞	∞
	$O_{4,4}$	3	∞	∞	3	∞	∞
	$O_{4,5}$	∞	∞	∞	∞	8	∞
	$O_{4,6}$	∞	∞	∞	∞	∞	9
J_5	$O_{5,1}$	∞	∞	9	∞	∞	∞
	$O_{5,2}$	∞	3	∞	∞	∞	∞
	$O_{5,3}$	∞	∞	∞	∞	5	∞
	$O_{5,4}$	∞	4	∞	∞	∞	4
	$O_{5,5}$	3	∞	∞	∞	∞	∞
	$O_{5,6}$	∞	∞	∞	1	∞	∞
J_6	$O_{6,1}$	∞	3	∞	3	∞	∞
	$O_{6,2}$	∞	∞	∞	3	∞	∞
	$O_{6,3}$	∞	∞	∞	∞	∞	9
	$O_{6,4}$	10	∞	∞	∞	∞	∞
	$O_{6,5}$	∞	∞	∞	∞	4	∞
	$O_{6,6}$	∞	∞	1	∞	∞	∞

Table 9: Instance "mt06-edata" by Hurink et al. [40]

Besides these adaptations of the JSSP instances by Fisher and Thompson, and Lawrence, Hurink et al. provided further adaptations of the JSSP instances by Adams et al. [1], Carlier and Pinson [9], and Applegate and Cook [2] by the same procedure as abovementioned: For each of the original "sdata" instances, similarly "edata"-, "rdata"-, and "vdata"-instances have been generated. The resulting instances in table 10 are available via the ORlibrary, but not widely considered in contrast to the instances in table 7.

Instance	n	m	LB	CP (Edata)	CP (Rdata)	CP (Vdata)
abz5†	10	10	859	1176	962	860*
abz6†	10	10	742	925	807	742*
abz7†	20	15	492	638	544	495
abz8†	20	15	506	654	555	509
abz9†	20	15	497	668	562	500
car1‡	11	5	5005	6176	5057	5013
car2‡	13	4	5929	6455	5987	5930
car3‡	12	5	5597	6856	5626	5600
car4‡	14	4	6514	7789*	6518	6517
car5‡	10	6	4909	7229*	5764	4932
car6‡	8	9	5486	8478	6147	5486*
car7‡	7	7	4216	6123	4432	4281*
car8‡	8	8	4613	7689	5692	4613*
orb1††	10	10	695	988	763	695*
orb2††	10	10	620	870	703	620*
orb3††	10	10	648	960	720	648*
orb4††	10	10	753	1016	753*	753*
orb5††	10	10	584	865	643	584*
orb6††	10	10	715	1004	766	715*
orb7††	10	10	275	387	302	275*
orb8††	10	10	573	894*	651	573*
orb9††	10	10	659	933	694*	659*
orb10††	10	10	681	937	750	681*

Table 10: Further FJSSP instances by Hurink et al. [29] (on the basis of † [1], ‡ [9], †† [2])

3.4 Multiprocessor-JSSP/FMS-instances by Dauzère-Pérès and Paulli

Dauzère-Pérès and Paulli introduced 18 instances originally generated for the multi-processor-JSSP [16,17] but later also used for a FMS environment [43]. An overview of the 18 instances is given in table 11 where max$|p_{i,k,j}|$-min$|p_{i,k,j}|$ $\forall i, k$ is the maximum allowed difference between the processing times of a particular operation on the fastest and slowest assignable machine. Furthermore, P is the probability used to decide whether a machine is to set assignable to a particular operation. If low probabilities lead to a situation in which no machine is assignable for a particular operation, one randomly selected machine is chosen.

Inst.	n	m	h_i	$p_{i,k,j}$	max$\|p_{i,k,j}\|$-min$\|p_{i,k,j}\|$ $\forall i, k$	P	LB	UB	CP
1	10	5	15...25	10...100	0	0.1	2505	2518 [41]	2568
2	10	5	15...25	10...100	0	0.3	2228	2231 [41]	2237
3	10	5	15...25	10...100	0	0.5	2228	2229 [41]	2231
4	10	5	15...25	10...100	5	0.1	2503	2503* [41]	2565
5	10	5	15...25	10...100	5	0.3	2189	2216 [41]	2243
6	10	5	15...25	10...100	5	0.5	2162	2196 [22]	2215
7	15	8	15...25	10...100	0	0.1	2187	2283 [41]	2329
8	15	8	15...25	10...100	0	0.3	2061	2069 [41]	2080
9	15	8	15...25	10...100	0	0.5	2061	2066 [41]	2064
10	15	8	15...25	10...100	5	0.1	2178	2291 [41]	2342
11	15	8	15...25	10...100	5	0.3	2017	2063 [41]	2078
12	15	8	15...25	10...100	5	0.5	1969	2030 [22]	2048
13	20	10	20...25	10...100	0	0.1	2161	2257 [22]	2293
14	20	10	20...25	10...100	0	0.3	2161	2167 [41]	2173
15	20	10	20...25	10...100	0	0.5	2161	2165 [22]	2164
16	20	10	20...25	10...100	5	0.1	2148	2255 [41]	2327
17	20	10	20...25	10...100	5	0.3	2088	2140 [22]	2162
18	20	10	20...25	10...100	5	0.5	2057	2127 [22]	2155

Table 11: Instances by Dauzère-Pérès and Paulli [16] (LB from [17])

The most striking property of the instances by Dauzère-Pérès and Paulli is the fact that the design is such that the number of operations per job is in any case higher than the number of machines. However, in contrast to Hurink et al. in some instances they allow for small deviations among the processing times of one particular operation with respect to the assignable machines.

$p_{i,k,j}$		M_1	M_2	M_3	M_4	M_5
J_1	$O_{1,1}$	42	∞	∞	∞	∞
	$O_{1,2}$	∞	∞	62	∞	∞
	$O_{1,3}$	∞	∞	∞	∞	22
	$O_{1,4}$	∞	∞	∞	16	∞
	$O_{1,5}$	96	∞	96	∞	96
	$O_{1,6}$	∞	∞	∞	∞	24
	$O_{1,7}$	63	∞	∞	∞	∞
	$O_{1,8}$	∞	29	∞	∞	∞
	$O_{1,9}$	∞	∞	83	∞	∞
	$O_{1,10}$	∞	∞	∞	∞	23
	$O_{1,11}$	∞	∞	∞	∞	66
	$O_{1,12}$	∞	∞	29	∞	∞
	$O_{1,13}$	∞	∞	∞	11	11
	$O_{1,14}$	∞	∞	62	∞	∞
	$O_{1,15}$	94	∞	∞	∞	∞
J_2	$O_{2,1}$	41	∞	41	∞	∞
	$O_{2,2}$	∞	90	∞	∞	∞
	$O_{2,3}$	63	∞	∞	∞	∞
	$O_{2,4}$	45	∞	∞	∞	∞
	$O_{2,5}$	∞	91	∞	∞	∞
	$O_{2,6}$	∞	∞	73	∞	∞
	$O_{2,7}$	∞	∞	∞	∞	31
	$O_{2,8}$	∞	∞	∞	97	∞
	$O_{2,9}$	∞	∞	∞	∞	11
	$O_{2,10}$	∞	∞	20	∞	∞
	$O_{2,11}$	∞	∞	∞	90	∞
	$O_{2,12}$	∞	21	∞	21	∞
	$O_{2,13}$	∞	78	∞	∞	∞
	$O_{2,14}$	∞	∞	∞	∞	37
	$O_{2,15}$	∞	∞	∞	∞	68
	$O_{2,16}$	48	48	∞	∞	∞
	$O_{2,17}$	∞	∞	∞	55	∞
	$O_{2,18}$	38	∞	∞	∞	∞
	$O_{2,19}$	∞	49	∞	∞	49
	$O_{2,20}$	∞	69	∞	∞	∞
J_3	$O_{3,1}$	∞	∞	37	∞	∞
	$O_{3,2}$	80	∞	∞	∞	∞
	$O_{3,3}$	∞	∞	83	83	∞
	$O_{3,4}$	∞	∞	∞	∞	39
	$O_{3,5}$	∞	∞	∞	∞	98
	$O_{3,6}$	∞	∞	∞	∞	71
	$O_{3,7}$	∞	24	∞	∞	∞
	$O_{3,8}$	∞	89	∞	∞	∞
	$O_{3,9}$	∞	∞	73	∞	∞
	$O_{3,10}$	∞	∞	∞	44	∞
	$O_{3,11}$	∞	∞	61	∞	∞
	$O_{3,12}$	53	∞	∞	∞	53
	$O_{3,13}$	35	∞	∞	∞	∞
	$O_{3,14}$	95	∞	∞	∞	∞
	$O_{3,15}$	33	∞	∞	∞	∞
	$O_{3,16}$	∞	∞	∞	∞	59
	$O_{3,17}$	∞	68	∞	∞	∞
	$O_{3,18}$	∞	∞	∞	∞	61
	$O_{3,19}$	56	∞	∞	∞	∞
	$O_{3,20}$	∞	28	∞	∞	∞
	$O_{3,21}$	∞	∞	∞	18	∞
	$O_{3,22}$	∞	∞	∞	35	∞
	$O_{3,23}$	73	∞	∞	∞	∞
...	....	...	...	...	...	...

Table 12: First instance of Dauzère-Pérès and Paulli [40] with three out of ten jobs

3.5 FJSSP-instances by Chambers and Barnes

Chambers and Barnes (1996) constructed 21 instances for the FJSSP [10]. The instances are divided into three sets where the instances of each set are based on a particular JSSP instance:

1. The "mt10" instances are based on the JSSP instance "10x10x10" introduced by Fisher and Thompson in [20].
2. The "setb4" and "seti5" instances are based on the JSSP instances "LA24", and "LA40" respectively, introduced by Lawrence in [38].

These JSSP instances were transformed into FJSSP instances by "replicating" machines according to deliberations about which machines a real-life production planner would intuitively replicate: Replication policy "p" states that the machine(s) with the most cumulative processing time over all operations (which can be calculated ex ante) should be replicated. According to replication policy "c", those machines with the largest number of critical-path operations, as identified by the best solutions of the original JSSP instances, should be replicated. These two replication policies are combined according to table 13 to generate seven FJSSP instances out of each of the three JSSP instances.

Instance model	Policies	Explanation
-x	p1	the machine with the greatest processing time is replicated once
-xx	p1,p1	the machine with the greatest processing time is replicated twice
-xxx	p1,p1,p1	the machine with the greatest processing time is replicated thrice
-xy	p1,p2	the machines with the greatest and second-greatest processing times are replicated once each
-xyz	p1,p2,p3	the machines with the greatest,second-greatest, and third-greatest processing times are replicated once each
-cM_c	c1	the machine with the greatest number of critical operations M_c is replicated once
-cc	c1,c2	the machines with the greatest and second-greatest number of critical operations are replicated once each

Table 13: Combined replication policies for the instances by Chambers and Barnes [10]

Since the processing times of the operations do not depend on the machines processing them, their FJSSP specification follows a similar concept as the MPM-JSSP introduced by Hurink et al. (cf. subsection 3.3). Table 14 provides an overview of the instances which are available via the OR library. A similar design approach has been undertaken in [11] where the flexibility of the operations is increased *without* replicating machines ("flexible-routing job shop problem", FRJS): With the help of the abovementioned design criteria, *assignabilities* instead of machines are replicated. These instances, however, are to the best of our knowledge neither used in other publications nor available via the OR library.

Instance name	n	m	k_i	LB	UB	CP
mt10x†	10	11	10	655	918 [41]	941
mt10xx†	10	12	10	655	918 [41]	941
mt10xxx†	10	13	10	655	918 [41]	929
mt10xy†	10	12	10	655	905 [22]	915
mt10xyz†	10	13	10	655	847 [41]	858
mt10c1†	10	11	10	655	927 [22]	927
mt10cc†	10	12	10	655	910 [41]	919
setb4x‡	15	11	10	846	925 [41]	950
setb4xx‡	15	12	10	846	925 [41]	938
setb4xxx‡	15	13	10	846	925 [10]	950
setb4xy‡	15	12	10	845	916 [41]	921
setb4xyz‡	15	13	10	838	905 [41]	910
setb4c9‡	15	11	10	857	914 [22]	914
setb4cc‡	15	12	10	857	909 [10]	917
seti5x‡	15	16	15	955	1201 [41]	1232
seti5xx‡	15	17	15	955	1199 [41]	1226
seti5xxx‡	15	18	15	955	1197 [41]	1199
seti5xy‡	15	17	15	955	1136 [41]	1136
seti5xyz‡	15	18	15	955	1125 [41]	1125
seti5c12‡	15	16	15	1027	1174 [41]	1193
seti5cc‡	15	17	15	955	1136 [10]	1136

Table 14: Test instances by Chambers and Barnes [10] († based on [20], ‡ based on [38], LB from [10])

$p_{i,k,j}$		M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}
J_1	$O_{1,1}$	29	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$O_{1,2}$	∞	78	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$O_{1,3}$	∞	∞	9	∞	∞	∞	∞	∞	∞	∞	∞
	$O_{1,4}$	∞	∞	∞	36	∞	∞	∞	∞	∞	∞	36
	$O_{1,5}$	∞	∞	∞	∞	49	∞	∞	∞	∞	∞	∞
	$O_{1,6}$	∞	∞	∞	∞	∞	11	∞	∞	∞	∞	∞
	$O_{1,7}$	∞	∞	∞	∞	∞	∞	62	∞	∞	∞	∞
	$O_{1,8}$	∞	∞	∞	∞	∞	∞	∞	56	∞	∞	∞
	$O_{1,9}$	∞	∞	∞	∞	∞	∞	∞	∞	44	∞	∞
	$O_{1,10}$	∞	∞	∞	∞	∞	∞	∞	∞	∞	21	∞
J_2	$O_{2,1}$	43	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$O_{2,2}$	∞	∞	90	∞	∞	∞	∞	∞	∞	∞	∞
	$O_{2,3}$	∞	∞	∞	∞	75	∞	∞	∞	∞	∞	∞
	$O_{2,4}$	∞	∞	∞	∞	∞	∞	∞	∞	∞	11	∞
	$O_{2,5}$	∞	∞	∞	69	∞	∞	∞	∞	∞	∞	69
	$O_{2,6}$	∞	28	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$O_{2,7}$	∞	∞	∞	∞	∞	∞	46	∞	∞	∞	∞
	$O_{2,8}$	∞	∞	∞	∞	∞	46	∞	∞	∞	∞	∞
	$O_{2,9}$	∞	∞	∞	∞	∞	∞	∞	72	∞	∞	∞
	$O_{2,10}$	∞	∞	∞	∞	∞	∞	∞	∞	30	∞	∞
J_3	$O_{3,1}$	∞	91	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$O_{3,2}$	85	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$O_{3,3}$	∞	∞	∞	39	∞	∞	∞	∞	∞	∞	39
	$O_{3,4}$	∞	∞	74	∞	∞	∞	∞	∞	∞	∞	∞
	$O_{3,5}$	∞	∞	∞	∞	∞	∞	∞	∞	90	∞	∞
	$O_{3,6}$	∞	∞	∞	∞	∞	10	∞	∞	∞	∞	∞
	$O_{3,7}$	∞	∞	∞	∞	∞	∞	∞	12	∞	∞	∞
	$O_{3,8}$	∞	∞	∞	∞	∞	∞	89	∞	∞	∞	∞
	$O_{3,9}$	∞	∞	∞	∞	∞	∞	∞	∞	∞	45	∞
	$O_{3,10}$	∞	∞	∞	∞	33	∞	∞	∞	∞	∞	∞
...	...	...	...	...	...	...	...	...	...	...	...	...

Table 15: Instance "mt10x" by Chambers and Barnes et al. [40] with three out of ten jobs

3.6 Instances for the general FJSSP with total flexibility by Kacem et al.

Kacem et al. (2002) designed four instances for the FJSSP with total flexibility and varying numbers of operations per job [35]. The data are not available via the OR library but published in [35]. An overview of the four instances is provided in table 16. A distinct property of these data is the fact that they also contain release times r_i as earliest possible starting times of the jobs (cf. table 17). $|O_{i,k,j}|$ indicates the total number of operations. Although it is unclear how exactly the processing times were generated, a striking property of these instances is the inclusion of extremely outlying values (cf. $p_{2,3,4}$=54 in table 17).

Instance name	n	m	$\|O_{i,k,j}\|$	optimal $C_{max}(r_i)$	optimal C_{max}	CP
Instance 1 (“little size”)	4	5	12	16* [35]	11*	11*
Instance 2 (“middle size”)	10	7	29	15* [35]	11*	11*
Instance 3 (“middle size”)	10	10	30	7* [35]	7*	7*
Instance 4 (“great size”)	15	10	56	23* [35]	12*	12

Table 16: Instances for the FJSSP with total flexibility by Kacem et al. [35]

$p_{i,k,j}$		M_1	M_2	M_3	M_4	M_5	r_i
J_1	$O_{1,1}$	2	5	4	1	2	3
	$O_{1,2}$	5	4	5	7	5	
	$O_{1,3}$	4	5	5	4	5	
J_2	$O_{2,1}$	2	5	4	7	8	5
	$O_{2,2}$	5	6	9	8	5	
	$O_{2,3}$	4	5	4	54	5	
J_3	$O_{3,1}$	9	8	6	7	9	1
	$O_{3,2}$	6	1	2	5	4	
	$O_{3,3}$	2	5	4	2	4	
	$O_{3,4}$	4	5	2	1	5	
J_4	$O_{4,1}$	1	5	2	4	12	6
	$O_{4,2}$	5	1	2	1	2	

Table 17: First instance of Kacem et al. [35]

3.7 FJSSP-instances for Mathematical Programming by Fattahi et al.

Fattahi et al. introduced twenty randomly generated small and medium-sized instances [19], which are divided into 10 small instances “SFJS1” to “SFJS10” and ten medium-sized instances “MFJS1” to “MFJS10” (cf. tables 18 and 19). Due to their comparatively small sizes, they are preferably used in mathematical programming approaches [19, 55].

Instance	n	m	$\max_i h_i$	flexibility	LB	UB	CP
SFJS1	2	2	2	total	66	66* [19]	66*
SFJS2	2	2	2	partial	107	107* [19]	107*
SFJS3	3	2	2	partial	212	221* [19]	221*
SFJS4	3	2	2	partial	331	355* [19]	355*
SFJS5	3	2	2	total	107	119* [19]	119*
SFJS6	3	2	3	partial	310	320* [19]	320*
SFJS7	3	5	3	total	397	397* [19]	397*
SFJS8	3	4	3	total	216	253* [19]	253*
SFJS9	3	3	3	total	210	210* [19]	210*
SFJS10	4	5	3	partial	427	516* [19]	516*
MFJS1	5	6	3	partial	403	468* [55]	468*
MFJS2	5	7	3	partial	396	446* [55]	446*
MFJS3	6	7	3	partial	396	466* [55]	466*
MFJS4	7	7	3	partial	496	554 [5]	554
MFJS5	7	7	3	partial	414	514* [55]	514
MFJS6	8	7	3	partial	614	635 [55]	634
MFJS7	8	7	4	partial	764	879 [5]	931
MFJS8	9	8	4	partial	764	884 [5]	884
MFJS9	11	8	4	partial	807 [55]	1088 [5]	1070
MFJS10	12	8	4	partial	944	1267 [5]	1208

Table 18: Small and medium-sized instances by Fattahi et al. [19]

$p_{i,k,j}$		M_1	M_2
J_1	$O_{1,1}$	25	37
	$O_{1,2}$	32	24
J_2	$O_{2,1}$	45	65
	$O_{2,2}$	21	65

Table 19: Instance “SFJS1” of Fattahi et al.

4 New instances for the FJSSPWC

4.1 Desired properties

In this article, we are introducing a new, practice-oriented specification of the FJSSP where similar machines are pooled to work centers [3, 45, 48, 51]. The following properties have been set in accordance with deliberations about an assumed industrial manufacturing environment in the metalworking industry [14, 28, 36, 46]:

1. "Similar" machines are grouped to work centers. Similarity in this context means that if one operation is assignable to a particular machine, it is also assignable to every other machine of that respective work center [51]. This property models the fact that particular operations primarily depend upon which technology may be used rather than which machine is able to process them: For instance, it is rather unlikely to occur that out of four *similar* drilling machines in a work center only two machines may be able to fulfill a particular drilling operation at all.

2. The first and last work center are obligatory in that every first operation of each job has definitely to be processed on one of the machines of the first work center, and every last operation of each job has definitely to be processed on one of the machines of the last work center. This is often observable in industrial practice when special preparing (cleaning, warming) procedures and special closing procedures (deburring, polishing) are necessary for all jobs.

3. The number of possible work center assignments is randomly fixed for every operation (except from the first and last operation of every job): Some tasks (e. g. removing material) may be done by several (machining) technologies, e. g. turning, milling, water jet or laser cutting. Others, like drilling a hole with a small diameter, only by a very particular technology.

4. The processing times $p_{i,k,j}$ were varied from 10 to 30 time units following a uniform distribution (cf. instances mk11 to mk15 by [7]). We neither considered outlying values nor broad ranges in processing times, as these are often due to disruptions, which will be considered within "robust scheduling" approaches. The time units may represent minutes, e. g. if a palm-sized metal part is to be machined (including machine setup, fixing the metal part, inspecting and measuring the part afterwards). Unlike the assumption of the MPM-JSSP that the processing times do not depend upon which machine has been selected, the randomization of processing times in the FJSSPWC allows for the fact that, in practice, a particular operation may be processed either by faster (newer) or slower (older) machines with different processing times resulting.

5. The numbers of jobs and work centers are varying from 10 to 100, and from 5 to 15, respectively. For every parameter combination of numbers of jobs and numbers of work centers five different instances with different possible assignments and processing times have been generated. Thus, three different sizes of manufacturing settings can be modeled: "Small", "medium-sized", and "large".

6. The numbers of operations per job h_i are not varied: Every job consists of exactly five operations ($h_i = h = 5$). The principle of "parsimony" in the generation of experimental data prohibits any unnecessary inclusion of variability [24]. Since our primary concerns in research are the consequences of introducing work centers (besides robust scheduling issues), an inclusion of a varying number of operations per job was not considered necessary. Furthermore, the amount of five operations per job seemed to be a reasonable value [23, 29].

7. The numbers of machines per work center are not varied: Every work center consists of exactly four machines [3]. If the setting of the work centers and machines is fixed among several instances, these instances can be regarded as distinct expressions of the same problem and thus analyzed statistically in a consolidated manner. Moreover, in a setting where the number of machines per work center is varying, bottleneck work centers might occur leading to a strong bias and/or obvious fixing of tangled decision variables.

The last two properties have also been fixed with respect to the fact that one does not strive to generate synthetic experimental data as a pure image of (often too simple or otherwise insufficient) real-life data, but to provide data that is both "typical" of real-life industry on the one hand and customized for the particular experiments to conduct on the other hand [24]. Furthermore, we attempted to equally consider faster and slower production resources on the one hand, and more or less broadly applicable technologies on the other hand in order to obtain computationally and scientifically challenging data. In practice, both effects may overlap such that, in a real-life manufacturing setting, there would sometimes be newer high-technology machines besides older, technologically restricted machines. In such a setting, "good" solutions might much easier or even trivially be found.
Table 20 provides an overview of the 60 generated instances. Note that the attributes "small", "medium" and "large" do not refer to the size of the respective instances but indicate small, medium-sized, and large production sites according to the number of work centers.
Despite the fact that above-mentioned design properties were primarily set with respect to an assumed industrial manufacturing setting, these newly designed instances do also allow for the modeling of whole supply chain networks where "work centers" would represent entire companies. In such a context, however, the geographical distances would possibly have to be considered. Furthermore, in the case of independent companies, there may not be any globally available information about jobs and/or production resources such that multi-agent optimization approaches could be favored over holistic heuristic approaches.
To our knowledge, only De Giovanni and Pezzella have modeled something like machine pools in flexible job shop scheduling [18]. However, they are dealing with a problem where several flexible manufacturing systems, each of which consists of not necessarily similar machines, are considered together in a so-called "distributed and flexible job-shop scheduling problem" (DFJS). In this problem, incoming jobs have to be processed in total by exactly one machine on either one of the technologically capable FMS such that the set and sequence of the operations of one particular job depends on the assigned FMS. Moreover, an additional handling time is considered from the input/output interface to/from the FMS. Due to these properties, the authors gener-

ated instances by replicating the data of common FMS instances two, three, and four times so that different sets of FMS are modeled.

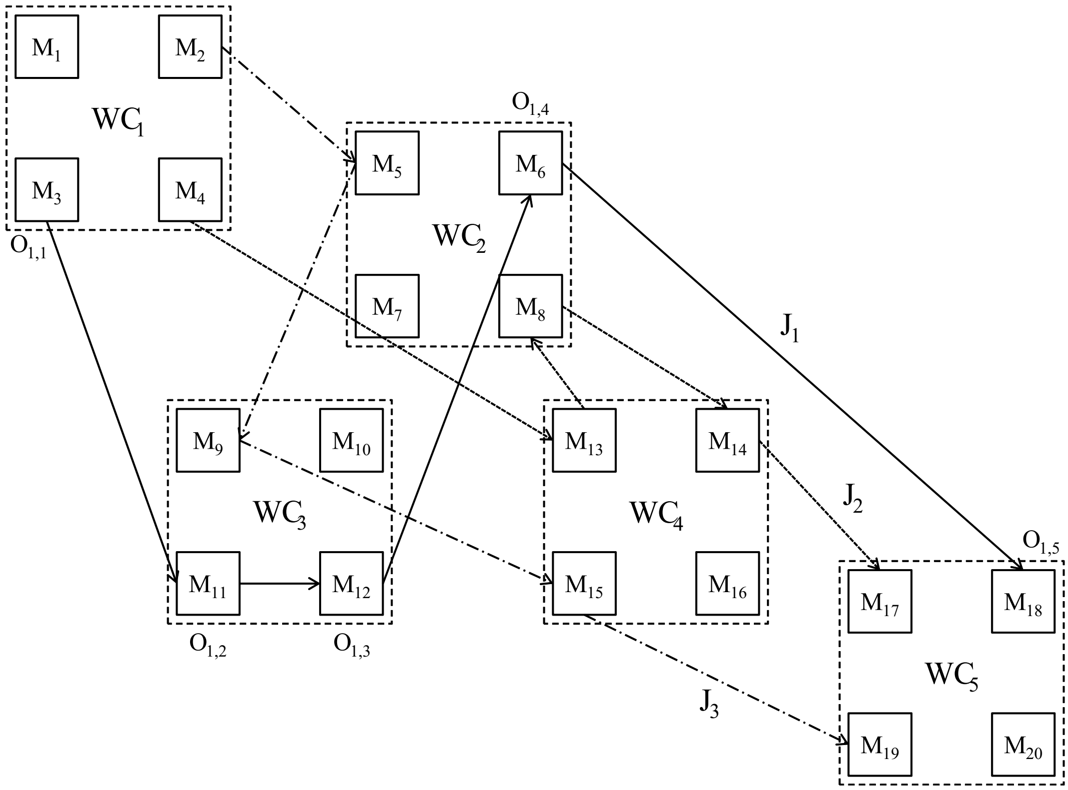

Figure 1: Problem illustration with five work centers and the first three jobs of instance 1 (arbitrarily assigned and sequenced, cf. table 21)

Instance	Name	n	$\lvert WC \rvert$	Seed	LB	UB (10 min CP)	UB (60 min CP)
1	Sm01.1	10	5	1	70	91	91
2	Sm01.2	10	5	2	75	91	91
3	Sm01.3	10	5	3	79	91	91
4	Sm01.4	10	5	4	76	98	97
5	Sm01.5	10	5	5	71	91	91
6	Sm02.1	20	5	6	78	134	131
7	Sm02.2	20	5	7	84	133	130
8	Sm02.3	20	5	8	76	132	128
9	Sm02.4	20	5	9	74	133	129
10	Sm02.5	20	5	10	81	136	133
11	Sm03.1	50	5	11	163	281	259
12	Sm03.2	50	5	12	157	274	251
13	Sm03.3	50	5	13	160	294	252
14	Sm03.4	50	5	14	164	295	258
15	Sm03.5	50	5	15	159	289	262
16	Sm04.1	100	5	16	327	576	566
17	Sm04.2	100	5	17	320	573	535
18	Sm04.3	100	5	18	321	567	555
19	Sm04.4	100	5	19	323	582	532
20	Sm04.5	100	5	20	322	570	522
21	Med01.1	10	10	21	78	85	85
22	Med01.2	10	10	22	69	87	87
23	Med01.3	10	10	23	72	86	86
24	Med01.4	10	10	24	70	87	87
25	Med01.5	10	10	25	80	87	87
26	Med02.1	20	10	26	70	127	122
27	Med02.2	20	10	27	81	134	132
28	Med02.3	20	10	28	73	128	123
29	Med02.4	20	10	29	75	134	125
30	Med02.5	20	10	30	80	134	127
31	Med03.1	50	10	31	79	292	272
32	Med03.2	50	10	32	77	292	259
33	Med03.3	50	10	33	77	279	245
34	Med03.4	50	10	34	78	286	265
35	Med03.5	50	10	35	79	268	253
36	Med04.1	100	10	36	152	551	531
37	Med04.2	100	10	37	153	548	536
38	Med04.3	100	10	38	151	540	527
39	Med04.4	100	10	39	153	556	516
40	Med04.5	100	10	40	156	562	521
41	Lar01.1	10	15	41	68	87	87
42	Lar01.2	10	15	42	75	87	87
43	Lar01.3	10	15	43	68	86	86
44	Lar01.4	10	15	44	68	85	85
45	Lar01.5	10	15	45	68	87	87
46	Lar02.1	20	15	46	73	128	124
47	Lar02.2	20	15	47	76	132	126
48	Lar02.3	20	15	48	74	142	134
49	Lar02.4	20	15	49	66	126	121
50	Lar02.5	20	15	50	73	137	131
51	Lar03.1	50	15	51	75	290	259
52	Lar03.2	50	15	52	76	264	255
53	Lar03.3	50	15	53	74	283	257
54	Lar03.4	50	15	54	75	303	267
55	Lar03.5	50	15	55	77	294	256
56	Lar04.1	100	15	56	99	545	538
57	Lar04.2	100	15	57	99	536	535
58	Lar04.3	100	15	58	100	538	531
59	Lar04.4	100	15	59	99	535	532
60	Lar04.5	100	15	60	101	551	537

Table 20: Test instances for the FJSSP with work centers

4.2 Generation procedure

In order to determine how many and which machines may process the various operations, four randomization processes were necessary ($|WC|$ being the total number of work centers):

1. The first and last operations $O_{i,1}$ and $O_{i,5}$ of each job J_i definitely have to pass the first work center (WC_{first}), or last work center (WC_{last}) respectively (cf. subsection 4.1 number 2): The corresponding processing times were set by a uniform distribution within [10;30]. For these operations ($O_{i,1}$ and $O_{i,5}$ $\forall i = 1...n$) and work centers (WC_{first} and WC_{last}) no other assignments were allowed.

For the remaining operations and work centers three randomization processes were necessary. Note that in the following steps, the first and last operations of each job ($O_{i,1}$ and $O_{i,5}$ $\forall i = 1...n$) as well as the first and last work centers (WC_{first} and WC_{last}) have been excluded ($|WC|$ being the total number of work centers):

2. For each of the operations $O_{i,2}...O_{i,4}$ of each job J_i, the numbers of assignable work centers have been determined by a random number in $[1; (|WC| - 2)]$.

3. Subsequently, random numbers in $[2; (|WC| - 1)]$ were used to decide which work center to set assignable. Within this randomization process two conditions were necessary. Firstly, the random number must be within $[2; (|WC| - 1)]$ (condition I). Secondly, the number must not have been drawn before (condition II). For every operation to assign, random numbers were generated until both conditions held. This process was repeated until the number of work centers to assign (cf. step 2) was met for the respective operation.

4. A third randomization process determines the processing times of the enabled assignments by a uniform distribution in [10;30].

For every instance, the seed for the initialization of the random-functions was set according to the values in table 20. The table also contains lower bounds (LB) computed according to [27] and best known upper bounds (UB) obtained by the ILOG Constraint Programming (CP) engine [30–32] within 10 and 60 minutes. The instance generator was coded in VB.NET. As an example, instance 1 (Sm01.1) is provided in table 21. In figure 1, a visualization is presented where the first three jobs of instance 1 are arbitrarily assigned. For a complete solution, a sequencing of the operations assigned to the same machine would be necessary.

$p_{i,k,j}$	W_1				W_2				W_3				W_4				W_5			
	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}	M_{17}	M_{18}	M_{19}	M_{20}
J_1 $O_{1,1}$	26	24	13	26	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
$O_{1,2}$	∞	∞	∞	∞	∞	∞	∞	∞	28	25	12	30	∞	∞	∞	∞	∞	∞	∞	∞
$O_{1,3}$	∞	∞	∞	∞	∞	∞	∞	∞	13	11	30	22	22	30	20	24	∞	∞	∞	∞
$O_{1,4}$	∞	∞	∞	∞	16	17	16	25	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
$O_{1,5}$	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	21	15	27	22
J_2 $O_{2,1}$	12	14	20	23	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
$O_{2,2}$	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	12	14	19	27	∞	∞	∞	∞
$O_{2,3}$	∞	∞	∞	∞	29	16	22	17	∞	∞	∞	∞	18	25	18	12	∞	∞	∞	∞
$O_{2,4}$	∞	∞	∞	∞	∞	∞	∞	∞	12	24	12	11	10	13	25	10	∞	∞	∞	∞
$O_{2,5}$	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	17	13	27	30
J_3 $O_{3,1}$	22	14	21	28	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
$O_{3,2}$	∞	∞	∞	∞	12	12	17	25	22	29	23	20	20	15	19	14	∞	∞	∞	∞
$O_{3,3}$	∞	∞	∞	∞	12	22	23	28	28	20	29	22	15	18	27	11	∞	∞	∞	∞
$O_{3,4}$	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	21	22	11	10	∞	∞	∞	∞
$O_{3,5}$	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	16	24	13	19
J_4 $O_{4,1}$	29	19	28	28	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
$O_{4,2}$	∞	∞	∞	∞	17	27	15	13	22	13	18	18	13	30	29	17	∞	∞	∞	∞
$O_{4,3}$	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	25	29	10	24	∞	∞	∞	∞
$O_{4,4}$	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	12	21	18	10	∞	∞	∞	∞
$O_{4,5}$	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	29	21	25	12
J_5 $O_{5,1}$	12	12	16	29	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
$O_{5,2}$	∞	∞	∞	∞	12	15	17	23	29	27	11	28	∞	∞	∞	∞	∞	∞	∞	∞
$O_{5,3}$	∞	∞	∞	∞	19	15	15	13	14	19	21	12	∞	∞	∞	∞	∞	∞	∞	∞
$O_{5,4}$	∞	∞	∞	∞	13	20	11	19	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
$O_{5,5}$	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	23	29	12	28
J_6 $O_{6,1}$	12	20	20	30	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
$O_{6,2}$	∞	∞	∞	∞	19	30	18	20	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
$O_{6,3}$	∞	∞	∞	∞	20	26	27	27	∞	∞	∞	∞	27	11	14	14	∞	∞	∞	∞
$O_{6,4}$	∞	∞	∞	∞	28	24	27	25	13	24	21	28	16	28	20	22	∞	∞	∞	∞
$O_{6,5}$	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	22	16	19	28
J_7 $O_{7,1}$	26	10	25	14	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
$O_{7,2}$	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	25	11	25	30	∞	∞	∞	∞
$O_{7,3}$	∞	∞	∞	∞	∞	∞	∞	∞	12	23	10	13	∞	∞	∞	∞	∞	∞	∞	∞
$O_{7,4}$	∞	∞	∞	∞	15	21	22	22	15	17	30	20	25	10	27	17	∞	∞	∞	∞
$O_{7,5}$	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	16	17	13	19
J_8 $O_{8,1}$	15	16	30	13	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
$O_{8,2}$	∞	∞	∞	∞	∞	∞	∞	∞	22	16	20	23	24	15	22	30	∞	∞	∞	∞
$O_{8,3}$	∞	∞	∞	∞	17	16	25	18	22	11	29	15	23	28	22	14	∞	∞	∞	∞
$O_{8,4}$	∞	∞	∞	∞	11	12	20	19	21	23	17	30	27	23	20	26	∞	∞	∞	∞
$O_{8,5}$	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	29	24	18	30
J_9 $O_{9,1}$	11	30	30	19	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
$O_{9,2}$	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	21	28	28	11	∞	∞	∞	∞
$O_{9,3}$	∞	∞	∞	∞	22	25	11	11	20	19	11	30	∞	∞	∞	∞	∞	∞	∞	∞
$O_{9,4}$	∞	∞	∞	∞	∞	∞	∞	∞	27	16	20	12	23	23	18	21	∞	∞	∞	∞
$O_{9,5}$	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	24	15	16	12
J_{10} $O_{10,1}$	25	11	28	23	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
$O_{10,2}$	∞	∞	∞	∞	11	14	12	28	29	14	11	12	∞	∞	∞	∞	∞	∞	∞	∞
$O_{10,3}$	∞	∞	∞	∞	∞	∞	∞	∞	28	30	15	28	∞	∞	∞	∞	∞	∞	∞	∞
$O_{10,4}$	∞	∞	∞	∞	24	11	30	29	30	16	12	30	20	25	29	17	∞	∞	∞	∞
$O_{10,5}$	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	23	11	20	27

Table 21: Instance 1 (Sm01.1) with ten jobs and five work centers

5 Conclusions

In this technical report, we provided a survey of common test instances for the different specifications of the FJSSP. These instances obey similar designing principles: The similarity of machines is modeled by a varying degree of flexibility of the different operations. Some of the instances assume that the processing times of the operations do not depend upon which machine they have been assigned to. Furthermore, the processing times often differ considerably within one particular instance. Due to these very special properties of the existing instances, we generated new instances for a realistic, practice-oriented specification of the FJSSP where similar machines are pooled to work centers, and the first and last work center are obligatory ("flexible job schop scheduling problem with work centers", FJSSPWC). These new instances cover plenty of real-life manufacturing environments: Most of the operations may be processed on various machines some of which are faster or slower than others. Moreover, many operations may be processed by different technologies: Metal may be removed by turning, milling, drilling, or other machining procedures. Eventually, the instances allow for preparing (cleaning, warming) and closing (deburring, polishing) procedures often observable in practice. Although these instances are designed by practice-oriented aspects, they are still scientifically and computationally challenging: In contrast to many of the existing instances, none of the new instances could be solved to optimality by a state-of-the-art Constraint Programming engine within a realistic computation time. Besides its original shop-floor conception, the pooling of production resources also allows for the modeling of whole supply chain networks where "work centers" would represent entire companies.

This technical report as well as the instances for the FJSSPWC alongside the existing FJSSP instances are all available via the following url:

`http://www.nbn-resolving.de/urn:nbn:de:gbv:705-opus-29827`

References

[1] J. Adams, E. Balas, and D. Zawack. The Shifting Bottleneck Procedure for Job Shop Scheduling. *Management Science*, 34(3):391–401, 1988.

[2] D. Applegate and W. Cook. A Computational Study of the Job-Shop Scheduling Problem. *ORSA Journal on Computing*, 3(2), 1991.

[3] D. Arnold, H. Isermann, A. Kuhn, H. Tempelmeier, and K. Furmans, editors. *Handbuch Logistik*. Springer, 3rd edition, 2008.

[4] A. Bagheri and M. Zandieh. Bi-Criteria Flexible Job-Shop Scheduling with Sequence-Dependent Setup Times – Variable Neighborhood Search Approach. *Journal of Manufacturing Systems*, 30(1):8–15, 2011.

[5] A. Bagheri, M. Zandieh, I. Mahdavi, and M. Yazdani. An Artificial Immune Algorithm for the Flexible Job-Shop Scheduling Problem. *Future Generation Computer Systems*, 26(4):533–541, 2010.

[6] J. Beasley. OR-Library. `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`.

[7] P. Brandimarte. Routing and Scheduling in a Flexible Job Shop by Tabu Search. *Annals of Operations Research*, 41(3):157–183, 1993.

[8] P. Brandimarte and M. Calderini. A Hierarchical Bicriterion Approach to Integrated Process Plan Selection and Job Shop Scheduling. *International Journal of Production Research*, 33(1):161–181, 1995.

[9] J. Carlier and E. Pinson. An Algorithm for Solving the Job-Shop Problem. *Management Science*, 35(2):164–176, 1989.

[10] J. B. Chambers and J. W. Barnes. Flexible Job Shop Scheduling by Tabu Search. *The University of Texas, Austin, TX, Technical Report Series ORP96-09, Graduate Program in Operations Research and Industrial Engineering*, 1996.

[11] J. B. Chambers and J. W. Barnes. Tabu Search for the Flexible-Routing Job Shop Problem. *The University of Texas, Austin, TX, Technical Report Series ORP96-10, Graduate Program in Operations Research and Industrial Engineering*, 1996.

[12] F. T. S. Chan, T. C. Wong, and L. Y. Chan. Flexible Job-Shop Scheduling Problem Under Resource Constraints. *International Journal of Production Research*, 44(11):2071–2089, 2006.

[13] H. Chen, J. Ihlow, and C. Lehmann. A Genetic Algorithm for Flexible Job-Shop Scheduling. In *IEEE International Conference on Robotics & Automation, Detroit, MI, USA*, pages 1120–1125, 1999.

[14] G. Chryssolouris, K. Dicke, and M. Lee. An Approach to Short Interval Scheduling for Discrete Parts Manufacturing. *International Journal of Computer Integrated Manufacturing*, 4(3):157–168, 1991.

[15] R. W. Conway, W. L. Maxwell, and L. W. Miller. *Theory of Scheduling*. Addison-Wesley, Reading, MA, 1967.

[16] S. Dauzère-Pérès and J. Paulli. Solving the General Multiprocessor Job-Shop Scheduling Problem. Technical report, Rotterdam School of Management, Erasmus Universiteit Rotterdam, 1994.

[17] S. Dauzère-Pérès and J. Paulli. An Integrated Approach for Modeling and Solving the General Multiprocessor Job-Shop Scheduling Problem Using Tabu Search. *Annals of Operations Research*, 70:281–306, 1997.

[18] L. De Giovanni and F. Pezzella. An Improved Genetic Algorithm for the Distributed and Flexible Job-Shop Scheduling Problem. *European Journal of Operational Research*, 200(2):395–408, 2010.

[19] P. Fattahi, M. S. Mehrabad, and F. Jolai. Mathematical Modeling and Heuristic Approaches to Flexible Job Shop Scheduling Problems. *Journal of Intelligent Manufacturing*, 18(3):331–342, 2007.

[20] H. Fisher and G. L. Thompson. Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules. In J. F. Muth and G. L. Thompson, editors, *Industrial Scheduling*, pages 225–251. Prentice-Hall, 1963.

[21] J. Gao, M. Gen, L. Sun, and X. Zhao. A Hybrid of Genetic Algorithm and Bottleneck Shifting for Multiobjective Flexible Job Shop Scheduling Problems. *Computers & Industrial Engineering*, 53(1):149–162, 2007.

[22] J. Gao, L. Sun, and M. Gen. A Hybrid Genetic and Variable Neighborhood Descent Algorithm for Flexible Job Shop Scheduling Problems. *Computers & Operations Research*, 35(9):2892–2907, 2008.

[23] M. J. Geiger and S. Petrovic. An Interactive Multicriteria Optimisation Approach to Scheduling. *Artificial Intelligence Applications and Innovations. IFIP International Federation for Information Processing*, 154:475–484, 2004.

[24] N. G. Hall and M. E. Posner. The Generation of Experimental Data for Computational Testing in Optimization. In T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, editors, *Experimental Methods for the Analysis of Optimization Algorithms*, pages 73–101. 2010.

[25] N. B. Ho and J. C. Tay. GENACE: An Efficient Cultural Algorithm for Solving the Flexible Job-Shop Problem. In *IEEE Congress on Evolutionary Computation (CEC2004)*, volume 2, pages 1759–1766, 2004.

[26] N. B. Ho and J. C. Tay. Evolving Dispatching Rules for Solving the Flexible Job-Shop Problem. In *IEEE Congress on Evolutionary Computation*, volume 3, pages 2848–2855, 2005.

[27] N. B. Ho and J. C. Tay. Solving Multiple-Objective Flexible Job Shop Problems by Evolution and Local Search. *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, 38(5):674–685, 2008.

[28] D. J. Hoitomt, P. B. Luh, S. Bailey, and S. LoStocco. A Practical System for Scheduling Manufacturing Job Shops. In *Proceedings of the Third International Conference on Computer Integrated Manufacturing, 1992*, pages 384–392, 1992.

[29] J. Hurink, B. Jurisch, and M. Thole. Tabu Search for the Job-Shop Scheduling Problem with Multi-Purpose Machines. *OR Spektrum*, 15(4):205–215, 1994.

[30] International Business Machines (IBM) Corporation. *IBM ILOG CP Optimizer V2.3 User's Manual*, 2009.

[31] International Business Machines (IBM) Corporation. *Detailed Scheduling in IBM ILOG CPLEX Optimization Studio with IBM ILOG CPLEX CP Optimizer*, 2010.

[32] International Business Machines (IBM) Corporation. *IDE and OPL Language and Interfaces Examples*, 2011.

[33] B. Jurisch. *Scheduling Jobs in Shops with Multi-Purpose Machines.* Dissertation am Fachbereich Mathematik/Informatik der Universität Osnabrück, 1992.

[34] I. Kacem, S. Hammadi, and P. Borne. Approach by Localization and Multiobjective Evolutionary Optimization for Flexible Job-Shop Scheduling Problems. *IEEE Transactions on Systems, Man and Cybernetics*, 32(1):1–13, 2002.

[35] I. Kacem, S. Hammadi, and P. Borne. Pareto-Optimality Approach for Flexible Job-Shop Scheduling Problems: Hybridization of Evolutionary Algorithms and Fuzzy Logic. *Mathematics and Computers in Simulation*, 60(3-5):245–276, 2002.

[36] R. M. Kerr and R. V. Ebsary. Implementation of an Expert System for Production Scheduling. *European Journal of Operational Research*, 33(1):17–29, 1988.

[37] Y. K. Kim, K. Park, and J. Ko. A Symbiotic Evolutionary Algorithm for the Integration of Process Planning and Job Shop Scheduling. *Computers & Operations Research*, 30(8):1151–1171, 2003.

[38] S. Lawrence. Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques. *Graduate School of Industrial Administration, Carnegie-Mellon University*, 1984.

[39] I. Mahdavi, B. Shirazi, and M. Solimanpur. Development of a Simulation-Based Decision Support System for Controlling Stochastic Flexible Job Shop Manufacturing Systems. *Simulation Modelling Practice and Theory*, 18(6):768–786, 2010.

[40] M. Mastrolilli. Flexible Job Shop Problem. `http://www.idsia.ch/~monaldo/fjsp.html`.

[41] M. Mastrolilli and L. M. Gambardella. Effective Neighbourhood Functions for the Flexible Job Shop Problem. *Journal of Scheduling*, 3(1):3–20, 2000.

[42] K. Mesghouni, S. Hammadi, and P. Borne. Evolution Programs for Job-Shop Scheduling. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC '97), Orlando, FL, USA*, pages 720–724, 1997.

[43] J. Paulli. A Hierarchical Approach for the FMS Scheduling Problem. *European Journal of Operational Research*, 86(1):32–42, 1995.

[44] 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.

[45] M. L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer, 3rd edition, 2008.

[46] J. M. J. Schutten. Practical Job Shop Scheduling. *Annals of Operations Research*, 83:161–178, 1998.

[47] C. R. Scrich, V. A. Armentano, and M. Laguna. Tardiness Minimization in a Flexible Job Shop: A Tabu Search Approach. *Journal of Intelligent Manufacturing*, 15(1):103–115, 2004.

[48] M. K. Starr. *Production and Operations Management*. Cengage, 2nd edition, 2008.

[49] H. Tamaki, T. Ono, H. Murao, and S. Kitamura. Modeling and Genetic Solution of a Class of Flexible Job Shop Scheduling Problems. In *8th IEEE International Conference on Emerging Technologies and Factory Automation*, volume 2, pages 343–350, 2001.

[50] J. C. Tay and N. B. Ho. Evolving Dispatching Rules Using Genetic Programming for Solving Multi-Objective Flexible Job-Shop Problems. *Computers & Industrial Engineering*, 54(3):453–473, 2008.

[51] V. T'kindt and J.-C. Billaut. *Multicriteria Scheduling: Theory, Models and Algorithms*. Springer, Berlin, 2 edition, 2006.

[52] S. A. Torabi, B. Karimi, and S. M. T. Fatemi Ghomi. The Common Cycle Economic Lot Scheduling in Flexible Job Shops: The Finite Horizon Case. *International Journal of Production Economics*, 97(1):52–65, 2005.

[53] L.-N. Xing, Y.-W. Chen, P. Wang, Q.-S. Zhao, and J. Xiong. A Knowledge-Based Ant Colony Optimization for Flexible Job Shop Scheduling Problems. *Applied Soft Computing*, 10(3):888–896, 2010.

[54] M. Yazdani, M. Amiri, and M. Zandieh. Flexible Job-Shop Scheduling with Parallel Variable Neighborhood Search Algorithm. *Expert Systems With Applications*, 37(1):678–687, 2010.

[55] C. Özgüven, L. Özbakır, and Y. Yavuz. Mathematical Models for Job-Shop Scheduling Problems with Routing and Process Plan Flexibility. *Applied Mathematical Modelling*, 34(6):1539–1548, 2010.

[56] N. Zribi, I. Kacem, A. El Kamel, and P. Borne. Optimization by Phases for the Flexible Job-Shop Scheduling Problem. In *5th Asian Control Conference*, pages 1889–1895, 2004.