A worker constrained flexible job shop scheduling problem with sequence-dependent setup times

Dominik Kress · **David Müller** · **Jenny Nossack**

November 8, 2018

Abstract We consider a flexible job shop scheduling problem with sequence-dependent setup times that incorporates heterogeneous machine operator qualifications by taking account of machine- and operator-dependent processing times. We analyze two objective functions, minimizing the makespan and minimizing the total tardiness, and present exact and heuristic decomposition based solution approaches. These approaches divide the scheduling problem into a vehicle routing problem with precedence constraints and an operator assignment problem, and connect these problems via logic inequalities. We assess the quality of our solution methods in an extensive computational study that is based on randomly generated as well as real-world problem instances.

Keywords Scheduling · Flexible job shop · Decomposition · Logic inequalities · Vehicle routing

1 Introduction

The well known *job shop scheduling problem* (JSP) is composed of a set of jobs and a set of machines (see, e.g., Błażewicz et al., 2007). Each job consists of a set of operations that must be processed in a given sequence in order to complete the job. That is, the processing of an operation of a job cannot be started before the preceding operation of that job is completed. There are no precedence relations among the operations

D. Kress (✉) · D. Müller
Management Information Science
University of Siegen
Kohlbettstr. 15, 57068 Siegen, Germany
E-mail: {dominik.kress, david.mueller}@uni-siegen.de

J. Nossack
Center for Advanced Studies in Management
HHL Leipzig
Jahnallee 59, 04109 Leipzig, Germany
E-mail: jenny.nossack@googlemail.com

of different jobs. Each operation must be processed on a specific machine and is associated with a corresponding processing time. Operations may not be preempted and each machine can process only one operation at a time. Given these restrictions, the problem is to determine sequences of the operations that are associated with the machines with the objective of optimizing some performance measure. It is well known that the JSP is strongly NP-hard for minimizing the makespan or the total tardiness (Graham et al., 1979; Lenstra and Rinnooy Kan, 1979).

Real-world manufacturing systems are usually more complex than the systems that can be represented by the classical JSP (see, e.g., Günther and Lee, 2007). Factory work floors, for example, oftentimes feature multiple machines of the same type as well as multi-purpose machines that allow for processing different types of operations. This is taken account of in the *flexible job shop scheduling problem* (FJSP), which generalizes the JSP by assuming that each operation must be processed by exactly one machine out of a given set of *eligible machines* (Brucker and Schlie, 1990; Hurink et al., 1994). Additionally, machines must oftentimes be prepared in order to be able to process a specific operation. The time needed for this preparation is referred to as a *setup time*. The importance of explicitly incorporating setup times into real-world scheduling problems has been discussed in the literature since the mid-1960s (see Allahverdi, 2015; Allahverdi and Soroush, 2008; Allahverdi et al., 1999, 2008). One distinguishes two classes of setup times (see, e.g., Allahverdi et al., 1999). If the setup time needed for some operation solely depends on the operation itself, it is referred to as sequence-independent. If it additionally depends on the immediately preceding operation that has been processed by the machine, it is referred to as sequence-dependent. Furthermore, real-world manufacturing systems oftentimes feature a heterogenous workforce which, for example, induces the need to take account of differently skilled *machine operators or workers* (De Bruecker et al., 2015).

1.1 Problem Setting and Motivation

In this paper, we address a FJSP with sequence-dependent setup times that takes account of differing worker skills. We will refer to this problem as the *worker constrained FJSP with sequence-dependent setup times*, and denote it by WSFJSP.

Our research is motivated by a real-world scheduling problem that has been brought to our attention during a project with a manufacturing company located in North Rhine-Westphalia, Germany. The company is specialized in the production of construction components – primarily cardan shaft mounts – made of aluminium, stainless steel, and steel. Its customers are mainly automotive suppliers. The products are fabricated in predefined lots that we will refer to as *jobs*. That is, each job is composed of multiple items of a specific product. The raw materials (usually aluminum profiles) are delivered by suppliers and subsequently run through various manufacturing operations, e.g. sawing, lathing, milling and punching, in predefined sequences in order to complete the final products. The items of each lot are jointly stored and moved in steel box pallets. Therefore, a specific operation of a job must be completed for the entire lot before the processing of the next operation of the job can be started. The production process of an exemplary job is illustrated in Fig. 1.

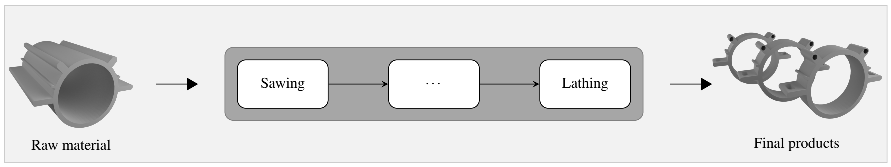


Fig. 1 Exemplary production process

There are several kinds of multi-purpose machines that are capable of performing different manufacturing operations, so that each operation of a job is associated with a set of eligible machines as described above. Cleaning operations, the change of tools, and process configurations result in sequence-dependent setup times between two consecutive operations processed on the same machine. Setup operators are not considered to be scarce. Machine operators, on the other hand, are considered to be a scarce resource and possess differing skills that our industry partner implements by making use of worker-dependent processing times. The processing time of a specific operation therefore depends on both, the machine chosen from the set of eligible machines and the worker assigned to the machine. A worker is assigned to an operation for its entire processing time and can only process one operation at a time.

Currently, the scheduling of the production processes at our industry partner is a daily manual task with a planning horizon of about one week. We were asked to implement scheduling algorithms that are capable of taking account of larger planning horizons. Additionally, we were asked to analyze two different objectives. First, based on the assumption that each job is associated with a due date at which the production of the job is intended to be completed, the aim is to minimize the total tardiness. Second, we were asked to provide results for minimizing the makespan. Clearly, due to the computational complexity of the JSP for both of these objectives, both variants of WSFJSP are strongly NP-hard.

According to the classical three-field notation by Graham et al. (1979) that was adapted by Allahverdi (2015) and Błażewicz et al. (1983) to include setup information and additional resource constraints, the two variants of WSFJSP considered in this article fall into the categories $FJ|res1 \cdot 1, ST_{sd}|C_{max}$ and $FJ|res1 \cdot 1, ST_{sd}|\sum T_i$, respectively.

1.2 Related Literature

The WSFJSP combines two variants of the FJSP that have been addressed in the literature, i.e. the FJSP with sequence-dependent setup times and the FJSP with explicit incorporation of machine operators. We will denote these variants by SFJSP and WFJSP, respectively, and summarize the relevant literature regarding these settings in this section. Note that machine scheduling problems with two types of resources, e.g. machines and machine operators, are sometimes also referred to as dual-resource constrained (DRC) systems (see, e.g. Treleven, 1989; Xu et al., 2011).

Surveys on scheduling problems with setup considerations for various machine environments are provided by Allahverdi et al. (1999, 2008) and Allahverdi (2015).

With respect to flexible job shops, Shen et al. (2017) consider the SFJSP with the objective of minimizing the makespan. They present a mixed-integer programming (MIP) formulation and a tabu search algorithm. Corresponding ant colony optimization approaches are presented by Zhang and Liu (2012) and Rossi (2014). Alternative MIP models are given by Nourali et al. (2012) and Saidi-Mehrabad and Fattahi (2007). The latter authors furthermore suggest another tabu search algorithm. Defersha and Chen (2010) additionally consider time lag requirements and machine release dates. They present a MIP model and a genetic algorithm. Some articles deal with objective functions that differ from minimizing the makespan. Mousakhani (2013), for example, aims at minimizing the total tardiness. The author presents a MIP model and proposes a metaheuristic algorithm based on iterated local search. Özgüven et al. (2012) present MIP formulations for the SFJSP with the objective of minimizing a weighted sum of the makespan and a specific measure for the degree of unbalancedness of the machine workloads. Similarly, Bagheri and Zandieh (2011) consider minimizing a weighted sum of the makespan and the mean tardiness. They present a variable neighborhood search algorithm.

The existing articles on the WFJSP with the objective of minimizing the makespan focus on the development of metaheuristic approaches. Examples include Lei and Guo (2014) (variable neighbourhood search), Yazdani et al. (2015) (simulated annealing, vibration damping optimization), Zhang et al. (2015) (particle swarm optimization), and Zheng and Wang (2016) (fruit fly optimization). Paksi and Ma'ruf (2016) analyze the objective of minimizing the total tardiness and propose a genetic algorithm. Multiple objectives are considered by Lang and Li (2011) and Lei and Tan (2016), the former of which also take account of uncertain processing times.

There exist only a few papers in the scheduling literature that explicitly take account of setup considerations as well as machine operator related constraints. Venditti et al. (2010) and Behnamian (2014) consider open shop and flow shop settings, respectively. Chen et al. (2003) address a FJSP which is very closely related to our setting. The authors consider machine operator related constraints and group-dependent setup times, where setup operations are necessary whenever switching between predefined groups of operations. Furthermore, they allow the generation of so called transfer lots. That is, each lot (as defined above for the WSFJSP) may be divided into multiple transfer lots, each of which can move to the next operation as soon as all parts within that transfer lot are completed. The objective function relates to maximizing the on-time delivery of products and the reduction of inventory and the number of setups. The problem is solved via a heuristic framework that makes use of a decomposition of the overall problem into smaller subproblems.

1.3 Contribution and Overview

Based on the above literature review, it can be concluded that FJSPs with sequence-dependent setup times and explicit incorporation of machine operators have received little attention. We will therefore contribute to the literature by analyzing a corresponding setting, namely the WSFJSP, which – as outlined above – is based on a real-world scheduling problem. We will propose an exact solution approach that decomposes the WSFJSP into a *vehicle routing problem (VRP) with precedence constraints* and a *worker assignment problem*. These models are combined by using *logic inequalities*. This exact approach will turn out to outperform an integrated MIP model and is able to solve small instances to optimality. Moreover, in order to be able to determine feasible solutions for medium and large instances within reasonable time, we will present heuristic algorithms based on the above decomposition. Our approaches are evaluated based on randomly generated test instances as well as real-world test instances that are based on data that has been provided by our industry partner.

The remainder of this article is structured as follows. In Section 2, we provide a formal definition of the WSFJSP and introduce the notation. The exact decomposition approach is presented in Section 3, while the heuristics are subject of Section 4. The computational tests are presented in Section 5. Finally, Section 6 concludes the paper.

2 Notation and Detailed Problem Description

We are given a set $I = \{1,\dots,n\}$ of jobs. Each job $i \in I$ is associated with a set of q_i operations $O_i = (i_1,\dots,i_{q_i})$ that have to be sequenced on a set M of machines and that may not be preempted. The sets O_i are assumed to be ordered for all jobs $i \in I$, which relates to the fact that, for any pair of operations $i_j, i_k \in O_i$ with $j < k$, i_j must be completed before the processing of i_k may start. Each operation $i_j \in O_i$, $i \in I$, must be processed by exactly one machine out of a set of eligible machines $M_{i_j} \subseteq M$ in order to be completed. To simplify the notation, we define $M_{i_j,k_l} := M_{i_j} \cap M_{k_l}$ for all $i,k \in I$, $i_j \in O_i$, $k_l \in O_k$. An operation can only be processed by a machine if exactly one worker out of a given set W of workers is assigned to the machine during the entire processing time of the operation. Each machine and each worker can process at most one operation at a time. A job is completed if all of its operations are completed. The completion time of an operation $i_j \in O_i$ of job $i \in I$ is denoted by C_{i_j}. The completion time of job $i \in I$ is denoted by C_i. Obviously, $C_i = C_{i_{q_i}}$ for all $i \in I$. The processing time of an operation $i_j \in O_i$ of a job $i \in I$, which we denote by $p_{i_j}^{m,w} \in \mathbb{Q}_0^+ \cup \{\infty\}$, is assumed to vary over different machines $m \in M_{i_j}$ and workers $w \in W$, which enables us to take account of differently skilled workers. We assume that all workers are available during the entire planning horizon. In case of a shift-based system and a planning horizon larger than one shift, we must therefore assume that all shifts are staffed with equally skilled workers, so that they can replace each other at shift changeovers. $p_{i_j}^{m,w} = \infty$ represents the case that worker $w \in W$ is not allowed to process an operation $i_j \in O_i$ of a job $i \in I$ on machine $m \in M_{i_j}$. We assume that for each operation $i_j \in O_i$ of a job $i \in I$ and each corresponding machine $m \in M_{i_j}$, there exists at least one worker $w \in W$ that can process the operation within finite time. Moreover, we assume that sequence-dependent setup times $s_{i_j,k_l}^{m} \in \mathbb{Q}_0^+$ occur when an operation $k_l \in O_k$, $k \in I$, is processed immediately after an operation $i_j \in O_i$, $i \in I$, on machine $m \in M_{i_j,k_l}$. Note that these setup times may differ among the machines $m \in M_{i_j,k_l}$. Setup operations do not require the assignment of a worker. Note that we consider an offline setting where all jobs, machines, and workers are available at the beginning of the planning horizon. Furthermore, there are no precedence relations between jobs.

The WSFJSP is to find a schedule, i.e. an allocation of operations to machines, a sequence of the operations that have been allocated to each machine, and a corresponding assignment of workers to operations, that is feasible with respect to the restrictions stated above, such that an objective function is addressed. We will consider two objectives. First, we will consider the minimization of the makespan $C_{max} := \max_{i \in I} C_i$. Second, given a due date $d_i \in \mathbb{Q}_0^+$ for each job $i \in I$, we will consider the minimization of the total tardiness $\sum_{i \in I} T_i$, where the tardiness T_i of job $i \in I$ is defined as $T_i := \max\{C_i - d_i, 0\}$.

Our notation is summarized in Table 1.

Table 1 Notation used throughout the paper

I	set of jobs	$I = \{1, \dots, n\}$, $\lvert I \rvert = n$
M	set of machines	
W	set of workers	
O_i	set of operations of job $i \in I$	$O_i = (i_1, \dots, i_{q_i})$, $\lvert O_i \rvert = q_i$
M_{i_j}	set of eligible machines for operation $i_j \in O_i$ of job $i \in I$	$M_{i_j} \subseteq M$
M_{i_j,k_l}	intersection of M_{i_j} and M_{k_l}, where $i,k \in I$, $i_j \in O_i$, and $k_l \in O_k$	$M_{i_j,k_l} := M_{i_j} \cap M_{k_l}$
d_i	due date of job $i \in I$	$d_i \in \mathbb{Q}_0^+$
$p_{i_j}^{m,w}$	processing time of operation $i_j \in O_i$ of job $i \in I$ when processed by worker $w \in W$ on machine $m \in M_{i_j}$	$p_{i_j}^{m,w} \in \mathbb{Q}_0^+ \cup \{\infty\}$
s_{i_j,k_l}^m	setup time when processing operation $k_l \in O_k$ of job $k \in I$ immediately after operation $i_j \in O_i$ of job $i \in I$ on machine $m \in M_{i_j,k_l}$	$s_{i_j,k_l}^m \in \mathbb{Q}_0^+$
C_{i_j}	completion time of operation $i_j \in O_i$ of job $i \in I$	
C_i	completion time of job $i \in I$	$C_i = C_{i_{q_i}}$
C_{max}	makespan of the schedule	$C_{max} := \max\limits_{i \in I} C_i$
T_i	tardiness of job $i \in I$	$T_i := \max\{C_i - d_i, 0\}$

3 Decomposition Approach

There exist similarities between VRPs and machine scheduling problems, which has resulted in several articles that have addressed machine scheduling settings from a vehicle routing perspective. Bigras et al. (2008), for example, analyze relationships between single machine scheduling problems with sequence-dependent setup times and the time-dependent traveling salesman problem (TSP). Similarly, Balas et al. (2008) model single machine problems that arise in their adapted shifting bottleneck procedure for a JSP with sequence-dependent setup times as TSPs with time windows. A similar method is applied by Tran and Beck (2012), who propose a logic-based Benders decomposition approach for scheduling unrelated parallel machines with sequence-dependent setup times.

In line with the aforementioned research, we propose to solve the WSFJSP in a branch-and-cut framework by decomposing it into a VRP with precedence constraints (master problem, MP) and a worker assignment problem (subproblem). The MP explicitly addresses the allocation of operations to machines and the sequencing of operations on each machine. The effect of the assignment of workers to operations on the objective function value is embedded into the MP by making use of logic inequalities, a class of constraints that is inspired by the logic-based Benders decomposition approach by Hooker and Ottosson (2003). Within our branch-and-cut framework, these inequalities are consecutively obtained by the subproblem, which explicitly determines an assignment of workers to operations such that the makespan or the total tardiness is minimized based on a given allocation and sequencing decision of (a relaxed version of) the MP.

To ease the notation in the remainder of this paper, we define a dummy job 0 with due date $d_0 = \infty$ and exactly one operation 0_1 with $M_{0_1} = M$. We set $M_{0_1,i_j} = M_{i_j,0_1} = M_{i_j}$ for all jobs $i \in I$ and operations $i_j \in O_i$. Furthermore, we set $p_{0_1}^{m,w} = 0$ for all machines $m \in M$ and workers $w \in W$. The setup times s_{0_1,i_j}^m can take arbitrary nonnegative rational values for all machines $m \in M$, jobs $i \in I$, and operations $i_j \in O_i$, which allows for modelling the first setup operation on each machine. Furthermore, $s_{i_j,0_1}^m = 0$ for all $m \in M$, $i \in I$, and $i_j \in O_i$.

3.1 Master Problem

As mentioned above, the master problem does not explicitly take account of the assignment of workers to operations. Hence, we make use of lower bounds for the processing times, and define $p_{i_j}^{m,min} := \min_{w \in W} p_{i_j}^{m,w}$ for all jobs $i \in I \cup \{0\}$, operations $i_j \in O_i$, and machines $m \in M_{i_j}$. Furthermore, we set $p_{i_j}^{min} := \min_{m \in M_{i_j}} p_{i_j}^{m,min}$ for all jobs $i \in I \cup \{0\}$ and operations $i_j \in O_i$.

When considering each machine as a vehicle and each operation as a distinct customer that must be visited (processed) exactly once, the MP corresponds to a variant of the VRP, where precedence relations capture the fact that the operations of each job have to follow a predefined order. The dummy operation 0_1 represents the depot of the VRP and allows modelling the start and end configurations of the machines. Following well established VRP formulations, we define the MP on a weighted, directed (multi-) graph $G = (V,E)$ with vertex set $V := \bigcup_{i \in I} O_i \cup \{0_1\}$ and the edge set being composed of $2 \cdot |M_{i_j,k_l}|$ edges between any pair of vertices $i_j, k_l \in V$, $i \neq k$, and $|M_{i_j,i_v}|$ edges between any pair of vertices $i_j, i_v \in V$, $v > j$, as illustrated in Fig. 2. Here, solid lines represent exemplary edges of the multigraph for some specific el-

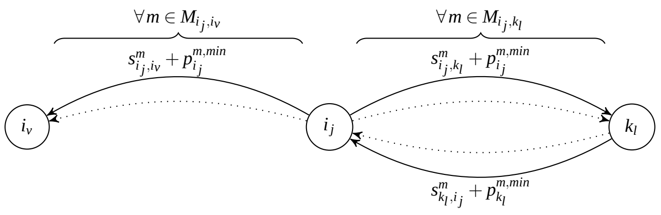

$\forall m \in M_{i_j,i_v}$
$s^m_{i_j,i_v} + p^{m,min}_{i_j}$
$\forall m \in M_{i_j,k_l}$
$s^m_{i_j,k_l} + p^{m,min}_{i_j}$
i_v
i_j
k_l
$s^m_{k_l,i_j} + p^{m,min}_{k_l}$

Fig. 2 Illustration of the graph representation, $i_v, i_j, k_l \in V$, $i \neq k$, $v > j$

igible machine, while dotted lines indicate that there potentially exist a number of additional edges that correspond to the remaining eligible machines.

Define $\bar{V} := V \setminus \{0_1\}$ and let B be a large positive number. Furthermore, for the sake of notational convenience, define $V_{i_j} := V \setminus \{i_k | k \leq j\}$ (set of potential direct successors of i_j), $\tilde{V}_{i_j} := V \setminus \{i_k | k \geq j\}$ (set of potential direct predecessors of i_j), and $\bar{V}_{i_j} := V_{i_j} \setminus \{0_1\}$ for all $i_j \in V$.

For the sake of clarity, the additional notation for the MP is summarized in Table 2.

Table 2 Additional notation for the master problem, defined on $G = (V, E)$

$p^{m,min}_{i_j}$	minimum processing time of operation $i_j \in O_i$ of job $i \in I$ processed on machine $m \in M_{i_j}$	$p^{m,min}_{i_j} := \min_{w \in W} p^{m,w}_{i_j}$
$p^{min}_{i_j}$	minimum processing time of operation $i_j \in O_i$ of job $i \in I$	$p^{min}_{i_j} := \min_{m \in M_{i_j}} p^{m,min}_{i_j}$
V	set of vertices	$V := \bigcup_{i \in I} O_i \cup \{0_1\}$
$\bar{V}$	set of vertices without depot vertex	$\bar{V} := V \setminus \{0_1\}$
V_{i_j}	set of potential direct successors of $i_j \in V$	$V_{i_j} := V \setminus \{i_k \| k \leq j\}$
$\tilde{V}_{i_j}$	set of potential direct predecessors of $i_j \in V$	$\tilde{V}_{i_j} := V \setminus \{i_k \| k \geq j\}$
$\bar{V}_{i_j}$	set of potential direct successors of $i_j \in V$ without depot vertex	$\bar{V}_{i_j} := V_{i_j} \setminus \{0_1\}$
B	large positive number	

Now, define a continuous variable $t_{i_j} \in \mathbb{R}^+_0$ for all operations $i_j \in V$. It represents the time when i_j is started to be processed on one of the machines. Moreover, define a continuous variable $C_{max} \in \mathbb{R}^+_0$ that represents the makespan, and a binary variable

$$y^m_{i_j,k_l} := \begin{cases} 1, & \text{if operation } k_l \text{ is processed directly after operation } i_j \text{ on machine } m \\ 0, & \text{else} \end{cases} \quad \forall i_j \in V, k_l \in V_{i_j}, m \in M_{i_j,k_l}. \quad (1)$$

Then, when considering makespan minimization, a MIP formulation for the MP is as follows:

$$\min C_{max} \quad (2)$$

s.t.

$$t_{i_{q_i}} + \sum_{k_l \in V_{i_{q_i}}} \sum_{m \in M_{i_{q_i},k_l}} y^m_{i_{q_i},k_l} \cdot p^{m,min}_{i_{q_i}} \leq C_{max} \qquad \forall i \in I, \quad (3)$$

$$\sum_{i_j \in \tilde{V}_{k_l}} \sum_{m \in M_{i_j,k_l}} y^m_{i_j,k_l} = 1 \qquad \forall k_l \in \bar{V}, \quad (4)$$

$$\sum_{i_j \in \bar{V}} y^m_{0_1,i_j} \leq 1 \qquad \forall m \in M, \quad (5)$$

$$\sum_{k_l \in \{a_b \in \tilde{V}_{i_j} | m \in M_{a_b}\}} y^m_{k_l,i_j} - \sum_{k_l \in \{a_b \in V_{i_j} | m \in M_{a_b}\}} y^m_{i_j,k_l} = 0 \quad \forall i_j \in V, m \in M_{i_j}, \quad (6)$$

$$t_{i_j} + s^m_{i_j,k_l} + p^{m,min}_{i_j} - t_{k_l} \leq \left(1 - y^m_{i_j,k_l}\right) B \qquad \forall i_j \in V, k_l \in \bar{V}_{i_j}, m \in M_{i_j,k_l}, \quad (7)$$

$$t_{i_j} + \sum_{k_l \in V_{i_j}} \sum_{m \in M_{i_j,k_l}} y^m_{i_j,k_l} \cdot p^{m,min}_{i_j} \leq t_{i_{j+1}} \qquad \forall\, i_j \in \bar{V} \text{with } j \leq q_i - 1, \tag{8}$$

$$t_{i_j} + p^{min}_{i_j} \leq t_{i_{j+1}} \qquad \forall\, i_j \in \bar{V} \text{with } j \leq q_i - 1, \tag{9}$$

$$\tilde{C}^h_{max} \cdot \left(1 - \sum_{(i_j,k_l,m) \in P^h} (1 - y^m_{i_j,k_l}) \right) \leq C_{max} \qquad \forall \text{ logic inequalities } h, \tag{10}$$

$$y^m_{i_j,k_l} \in \{0,1\} \qquad \forall\, i_j \in V, k_l \in V_{i_j}, m \in M_{i_j,k_l}, \tag{11}$$

$$t_{i_j} \in \mathbb{R}^+_0 \qquad \forall\, i_j \in V, \tag{12}$$

$$C_{max} \in \mathbb{R}^+_0. \tag{13}$$

The objective function (2) minimizes the makespan of the schedule. Constraints (3) set a lower bound on the makespan. Constraints (4) guarantee that each operation is scheduled exactly once. Inequalities (5) ensure that there is at most one operation that is processed first on each machine. Flow conservation is enforced by constraints (6). Constraints (7) enforce a time increase of at least $s^m_{i_j,k_l} + p^{m,min}_{i_j}$ (setup and processing) compared to t_{i_j}, if $y^m_{i_j,k_l} = 1$. Constraints (8) guarantee that an operation $i_{j+1} \in O_i$ of job $i \in I$ cannot start before its preceding operation $i_j \in O_i$ has been processed completely. Constraints (9) are redundant to constraints (8), but have shown to improve the computational performance in our tests. The logic inequalities (10) correct a potential underestimation of the objective function value that is caused by minimizing over all workers when determining the processing times that (partly) define the edge weights of the underlying graph representation. They are explained in more detail in Section 3.4, where we also introduce the related notation. Finally, constraints (11)–(13) define the domains of the variables.

Model (2)–(13) can easily be adapted for minimizing the total tardiness instead of the makespan. To do so, define a continuous variable $T_i \in \mathbb{R}^+_0$, representing the tardiness of job i, for all $i \in I$. We get:

$$\min T = \sum_{i \in I} T_i \tag{14}$$

$$\text{s.t. } (4) - (9),\ (11),\ (12),$$

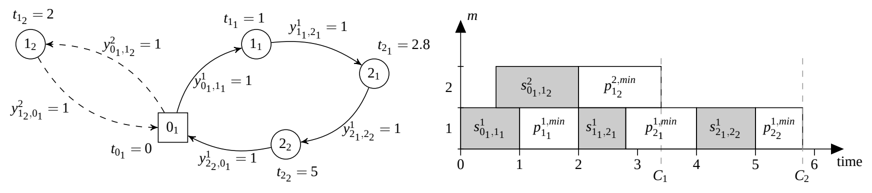
$$t_{i_{q_i}} + \sum_{k_l \in V_{i_{q_i}}} \sum_{m \in M_{i_{q_i},k_l}} y^m_{i_{q_i},k_l} \cdot p^{m,min}_{i_{q_i}} - d_i \leq T_i \quad \forall\, i \in I, \tag{15}$$

$$\tilde{T}^h \cdot \left(1 - \sum_{(i_j,k_l,m) \in P^h} (1 - y^m_{i_j,k_l}) \right) \leq \sum_{i \in I} T_i \quad \forall \text{ logic inequalities } h, \tag{16}$$

$$T_i \in \mathbb{R}^+_0 \qquad \forall\, i \in I. \tag{17}$$

The objective function (14) minimizes the total tardiness, which we will hereafter denote by T, of the jobs. Constraints (15) set a lower bound on the tardiness of each job. Constraints (16) are logic inequalities that are defined in analogy to constraints (10). Constraints (17) define the domains of the newly introduced variables.

We denote a relaxed version of the MP, which results from considering only a (potentially empty) subset of constraints (10) or (16), by RMP. Based on a given solution of the MP or a RMP, we can construct a directed graph with vertex set V that includes only those edges of G that are associated to a positive variable (1). We will refer to this graph as the *supporting graph* of the solution. The supporting graph of any feasible solution of the MP or a RMP is composed of at most $|M|$ edge-disjoint cycles, each of which includes the depot vertex 0_1, such that every vertex of the set $\bar{V}$ is included in exactly one of the cycles. Fig. 3 illustrates the supporting graph of a feasible solution of the MP (Fig. 3a) and the corresponding Gantt chart that represents the processing of the operations on the machines (Fig. 3b) for an example instance with two jobs and two machines. The solid and dashed lines of Fig. 3a represent



(a) Supporting graph (b) Gantt chart

Fig. 3 Representation of a feasible solution of the MP

positive variables (1) that are associated to the first or second machine, respectively. The values of the variables (12) are given next to the corresponding vertices of the set V. Hence, in the depicted solution, machine 1 processes operations 1_1, 2_1, and 2_2, while machine 2 solely processes operation 1_2. As illustrated in the Gantt chart in Fig. 3b, the precedence constraints among the operations of job 1 result in idle time that precedes the setup operation which is necessary to process 1_2 on machine 2.

3.2 Subproblem

Based on a feasible allocation and sequencing decision of the MP or a RMP and the true processing times, the subproblem determines an assignment of workers to operations such that the makespan or the total tardiness is minimized. It is important to note that there always exists such a feasible assignment, because the assignment of workers to operations can only cause temporal shifts of the operations on the machines.

Let $\bar{\mathbf{y}} := (\bar{y}^m_{i_j,k_l} | i_j \in V, k_l \in V_{i_j}, m \in M_{i_j,k_l})$ denote the vector of variables (1) of a solution of the MP or a RMP and refer to the corresponding value of the objective function (2) or (14) by $\bar{C}_{max}$ and $\bar{T}$, respectively. Based on $\bar{\mathbf{y}}$, we can construct parameters that specify the machines that process the operations:

$$\bar{z}^m_{i_j} := \begin{cases} 1, & \text{if } \sum_{k_l \in \{a_b \in \bar{V}_{i_j} | m \in M_{a_b}\}} \bar{y}^m_{k_l,i_j} = 1 \\ 0, & \text{else} \end{cases} \qquad \forall i_j \in \bar{V}, m \in M_{i_j}. \tag{18}$$

In analogy to the variables of the MP, we additionally define continuous variables $\tilde{t}_{i_j} \in \mathbb{R}_0^+$ for all operations $i_j \in V$, as well as continuous variables $\tilde{C}_{max} \in \mathbb{R}_0^+$ and $\tilde{T}_i \in \mathbb{R}_0^+$ for all jobs $i \in I$. Moreover, we define the following binary variables:

$$x_{i_j}^w := \begin{cases} 1, & \text{if operation } i_j \text{ is processed by worker } w \\ 0, & \text{else} \end{cases} \qquad \forall i_j \in \bar{V}, w \in W, \tag{19}$$

$$x_{i_j,k_l}^w := \begin{cases} 1, & \text{if operations } i_j \text{ and } k_l \text{ are processed by worker } w \\ 0, & \text{else} \end{cases} \qquad \forall i_j, k_l \in \bar{V}, i_j \neq k_l, w \in W, \tag{20}$$

$$v_{i_j,k_l} := \begin{cases} 1, & \text{if the processing of operation } k_l \text{ starts before the processing of operation } i_j \text{ finishes} \\ 0, & \text{else} \end{cases} \qquad \forall i_j, k_l \in \bar{V}, i_j \neq k_l. \tag{21}$$

We furthermore define $x_{0_1}^w := 1$ for all workers $w \in W$.

The subproblem for makespan minimization is then defined as follows:

$$\min \tilde{C}_{max} \tag{22}$$

s.t.

$$\tilde{t}_{i_{q_i}} + \sum_{m \in M_{i_{q_i}}} \sum_{w \in W} \bar{z}_{i_{q_i}}^m \cdot x_{i_{q_i}}^w \cdot p_{i_{q_i}}^{m,w} \leq \tilde{C}_{max} \qquad \forall i \in I, \tag{23}$$

$$\sum_{w \in W} x_{i_j}^w = 1 \qquad \forall i_j \in \bar{V}, \tag{24}$$

$$\tilde{t}_{i_j} + s_{i_j,k_l}^m + p_{i_j}^{m,w} - \tilde{t}_{k_l} \leq \left(1 - \bar{y}_{i_j,k_l}^m \cdot x_{i_j}^w\right) B \qquad \forall i_j \in V, k_l \in \bar{V}_{i_j}, m \in M_{i_j,k_l}, w \in W, \tag{25}$$

$$\tilde{t}_{i_j} + \sum_{m \in M_{i_j}} \sum_{w \in W} \bar{z}_{i_j}^m \cdot x_{i_j}^w \cdot p_{i_j}^{m,w} \leq \tilde{t}_{i_{j+1}} \qquad \forall i_j \in \bar{V} \text{with } j \leq q_i - 1, \tag{26}$$

$$x_{i_j,k_l}^w \geq x_{i_j}^w + x_{k_l}^w - 1 \qquad \forall i_j, k_l \in \bar{V}, i_j \neq k_l, w \in W, \tag{27}$$

$$\tilde{t}_{i_j} + \sum_{m \in M_{i_j}} \sum_{w \in W} \bar{z}_{i_j}^m \cdot x_{i_j}^w \cdot p_{i_j}^{m,w} - \tilde{t}_{k_l} \leq B \cdot v_{i_j,k_l} \qquad \forall i_j, k_l \in \bar{V}, i_j \neq k_l, \tag{28}$$

$$x_{i_j,k_l}^w \leq 2 - v_{i_j,k_l} - v_{k_l,i_j} \qquad \forall i_j, k_l \in \bar{V}, i_j \neq k_l, w \in W, \tag{29}$$

$$x_{i_j}^w \in \{0,1\} \qquad \forall i_j \in V, w \in W, \tag{30}$$

$$x_{0_1}^w = 1 \qquad \forall w \in W, \tag{31}$$

$$x_{i_j,k_l}^w \in \{0,1\} \qquad \forall i_j, k_l \in \bar{V}, i_j \neq k_l, w \in W, \tag{32}$$

$$v_{i_j,k_l} \in \{0,1\} \qquad \forall i_j, k_l \in \bar{V}, i_j \neq k_l, \tag{33}$$

$$\tilde{t}_{i_j} \in \mathbb{R}_0^+ \qquad \forall i_j \in V, \tag{34}$$

$$\tilde{C}_{max} \in \mathbb{R}_0^+. \tag{35}$$

Given the vector $\bar{\mathbf{y}}$ of a solution of the MP or a RMP, the objective function (22) minimizes the makespan of the schedule. Constraints (23) bound the makespan from below. Restrictions (24) ensure that each operation is assigned to exactly one worker. Constraints (25) and (26) are defined in analogy to restrictions (7) and (8) of the MP. Inequalities (27) and (28) ensure that the variables (20) and (21) are set to one when needed. Based on these variables, constraints (29) guarantee that each worker is assigned to at most one operation at a time. Finally, constraints (30)–(35) define the domains of the variables.

For the case of minimizing the total tardiness, the subproblem is similarly defined as follows:

$$\min \tilde{T} = \sum_{i \in I} \tilde{T}_i \tag{36}$$

$$\text{s.t. } (24) - (34),$$

$$\tilde{t}_{i_{q_i}} + \sum_{m \in M_{i_{q_i}}} \sum_{w \in W} \bar{z}^m_{i_{q_i}} \cdot x^w_{i_{q_i}} \cdot p^{m,w}_{i_{q_i}} - d_i \leq \tilde{T}_i \quad \forall i \in I, \tag{37}$$

$$\tilde{T}_i \in \mathbb{R}^+_0 \quad \forall i \in I. \tag{38}$$

Here, the objective function (36) minimizes the total tardiness of the jobs based on the vector $\bar{\mathbf{y}}$ of a solution of the MP or a RMP. Constraints (37) bound the tardiness of each job from below, and constraints (38) define the domains of the tardiness variables.

3.3 Subtour Elimination Cuts

As described in Section 3.1, the supporting graph of any feasible solution of the MP or a RMP is composed of a set of at most $|M|$ edge-disjoint cycles, each of which includes the depot vertex 0_1, such that every vertex of the set $\bar{V}$ is included in exactly one of the cycles. For fractional solutions, that may arise during the solution process of a standard MIP solver that relaxes the integrality constraints of the MP or a RMP, we define the supporting graph to include all vertices of the set V as well as all edges of G that are associated with positive relaxed variables (1). Here, the above property needs not be true, which is illustrated in Fig. 4 based on a fractional solution of a RMP of an example instance with two machines and two jobs, both of which consist of two operations (refer to the previous example in Fig. 3a for an explanation of the illustration). Each machine of the example instance is eligible for processing all operations. As can be seen, the supporting graph of the depicted fractional solution is composed of four edge-disjoint cycles, two of which do not contain the depot vertex 0_1.

We make use of the above fact to separate subtour elimination constraints that we add to our branch-and-cut framework in addition to the logic inequalities. More specifically, when a fractional solution arises at a node of the branch-and-bound tree, we construct the corresponding supporting graph as described above and determine the set A of connected components of the undirected pendent of this graph. To do so, we make use of a depth-first search approach provided by the Boost Graph Library

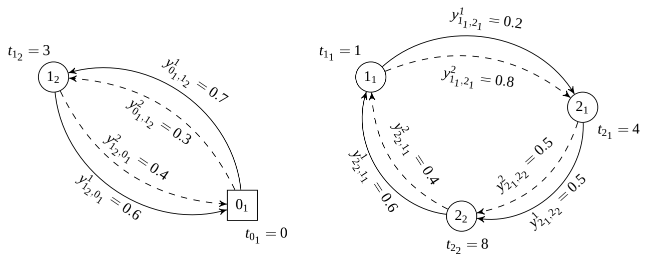


Fig. 4 Supporting graph of a fractional solution of a RMP of an example instance

(Boost, 2018). Each element of the set A is a set of vertices, each of which is included in the respective connected component. If $|A| > 1$, the subtour elimination constraints

$$\sum_{i_j \in S} \sum_{k_l \in V \setminus S} \sum_{m \in M_{i_j,k_l}} y^m_{i_j,k_l} \geq 1 \quad \forall S \in A \tag{39}$$

are generated and included in the branch-and-cut procedure.

3.4 Branch-and-Cut Framework

We propose to solve the WSFJSP in a branch-and-cut framework offered by a standard MIP solver. Modern solvers allow their users to guide a branch-and-cut solution process via so called callbacks. We make use of these callbacks to consecutively generate and add logic inequalities (10) or (16) and subtour elimination constraints (39) after having started the solver on the RMP with an empty set of logic inequalities. This is illustrated in Algorithm 1.

The logic inequalities are indexed by h and are derived by solving subproblems in step 3 of Algorithm 1. The basic idea is as follows. The RMP makes use of lower bounds of the processing times, so that the objective function value of a potential incumbent solution may be incorrect with respect to the real processing times. The corresponding logic inequality assures that, for this specific solution, the objective function value of the RMP implicitly takes account of the worker assignment restrictions and the real processing times which are explicitly modelled in the subproblem.

The overall process of handling a potential incumbent solution with objective value $\bar{C}_{max}$ ($\bar{T}$) in step 3 of Algorithm 1 is as follows. First, the corresponding subproblem is generated and solved by a standard solver. The resulting objective function value is referred to as $\tilde{C}_{max}$ ($\tilde{T}$). If the subproblem is feasible and $\bar{C}_{max} = \tilde{C}_{max}$ ($\bar{T} = \tilde{T}$), no logic inequality is violated and the branch-and-cut process continues without any modification. If, however, the subproblem is feasible and $\bar{C}_{max}$ ($\bar{T}$) is smaller than $\tilde{C}_{max}$ ($\tilde{T}$), we generate a logic inequality, add it to the RMP, and proceed with the branch-and-cut solution process.

The process of separating subtour elimination cuts in step 4 of Algorithm 1 proceeds as outlined in Section 3.3. Note, however, that we do not construct these cuts for all potential fractional solutions in order to balance the computational effort needed for their separation and their use within the branch-and-cut procedure. Instead, we

```
Input: Instance $Inst$ of WSFJSP, parameter $\rho_s$
Output: Optimal solution of $Inst$
1. Initialize $h := 0$ (counter for logic inequalities).
2. Call standard branch-and-cut solver for the RMP of $Inst$ with an empty set of logic
   inequalities. Whenever a potential incumbent solution with vector $\bar{\mathbf{y}}$ and objective function
   value $\bar{C}_{max}$ ($\bar{T}$) is found, go to 3 (solve subproblem and potentially generate and add a logic
   inequality). Whenever a fractional solution arises at a node of the branch-and-bound tree and
   the node cannot be pruned, go to 4 (potentially generate and add subtour elimination
   constraints) with a given probability $\rho_s$.
3. Determine parameters (18) and solve the subproblem based on $\bar{\mathbf{y}}$ by making use of a standard
   solver. The resulting objective function value is referred to as $\tilde{C}_{max}$ ($\tilde{T}$).
   (a) If the subproblem is feasible and $\bar{C}_{max} < \tilde{C}_{max}$ ($\bar{T} < \tilde{T}$), set $h := h+1$, $\tilde{C}^h_{max} := \tilde{C}_{max}$
       ($\tilde{T}^h := \tilde{T}$), and construct the logic inequality
           $$\tilde{C}^h_{max} \cdot (1 - \sum_{(i_j,k_l,m)\in P^h} (1 - y^m_{i_j,k_l})) \leq C_{max}$$
       (in case of makespan minimization), or
           $$\tilde{T}^h \cdot (1 - \sum_{(i_j,k_l,m)\in P^h} (1 - y^m_{i_j,k_l})) \leq \sum_{i\in I} T_i$$
       (in case of minimization of the total tardiness). Here
           $$P^h := \left\{(i_j,k_l,m)|\bar{y}^m_{i_j,k_l} = 1, i_j \in V, k_l \in V_{i_j}, m \in M_{i_j,k_l}\right\}.$$
       Add this constraint to the branch-and-cut process and continue in 2.
   (b) If the subproblem is feasible and $\bar{C}_{max} = \tilde{C}_{max}$ ($\bar{T} = \tilde{T}$), the potential incumbent solution
       becomes the new incumbent solution and the branch-and-cut process in 2 continues.
4. Construct the supporting graph of the fractional solution and determine the connected
   components $A$ of this graph. If $|A| > 1$, generate subtour elimination constraints (39). Add
   these constraints to the branch-and-cut process and continue in 2.
```

Algorithm 1 Branch-and-cut framework for WSFJSP

generate subtour elimination cuts with a given probability ρ_s for each fractional solution.

4 Decomposition Based Heuristics

Due to the computational complexity of WSFJSP, it cannot be expected that our branch-and-cut framework is able to solve medium- to large-sized instances to optimality within reasonable time. In this section, we will therefore introduce two heuristic approaches that are based on the above decomposition. First, in Section 4.1, we will present a simple approach for generating a feasible initial solution of WSFJSP. Then, in Section 4.2, we will develop a more sophisticated improvement procedure which is inspired by an idea presented by Della Croce et al. (2014) for a flow shop scheduling problem with two machines.

4.1 Generating an Initial Solution

In line with the decomposition introduced in Section 3, we generate an initial solution of a given instance of the WSFJSP by a hierarchical approach composed of two steps. First, based on the lower bounds of the processing times introduced in Section 3.1, our approach allocates operations to machines and decides on the sequences of the operations on the machines. This step corresponds to finding a solution to the RMP with an empty set of logic inequalities. Second, given this solution, the approach assigns workers to operations based on the correct processing times, which corresponds to determining a solution to the subproblem and, thus, to the given instance of the WSFJSP.

4.1.1 Allocation and Sequencing

The allocation and sequencing decisions are made by a priority-rule based heuristic that follows an algorithmic idea of Giffler and Thompson (1960) for the classical JSP. Our approach is outlined in detail in Algorithm 2.

Basically, the algorithm iteratively (loop 7–21) allocates operations that can start being processed (when applying lower bounds of the processing times as introduced in Section 3.1) at the respective point of time (represented by “release times”, see lines 3–4 and 12–19), when taking account of the corresponding precedence constraints. Among all operations that compete for the same machine (chosen in lines 8–9) in some iteration, exactly one operation is chosen based on a priority rule (line 10). As there exists a vast amount of potential priority rules (see, for example, Haupt, 1989), we rely on a comparative study by Sels et al. (2012), who analyze the performance of multiple priority rules for JSPs and FJSPs under different objective functions, including settings with setup times. Based on the results of this analysis, we decided to make use of two priority rules. In case of the objective of minimizing the makespan, we apply a combination of the *flow due date* (FDD), the *most work remaining* (MWKR), and the *shortest setup time* (SS) priority rules. Formally, after having decided on a machine $m^* \in M$, we determine the last operation $a_b(m^*) \in V$ that is processed by m^* in its current processing sequence, where $a_b(m^*) = 0_1$ in case of the empty sequence. Among all relevant candidate operations, our priority rule then selects an operation i_j with smallest value

$$\frac{\sum\limits_{1\leq k\leq j} p_{i_k}^{min}}{\sum\limits_{j\leq k\leq q_i} p_{i_k}^{min}} + s_{a_b(m^*),i_j}^{m^*}.$$

Similarly, in case of the objective of minimizing the total tardiness, we consider a combination of the SS and the *earliest due date* (EDD) priority rules, where an operation i_j with smallest value

$$d_i + s_{a_b(m^*),i_j}^{m^*}$$

among the relevant candidate operations is selected.

```
Input: RMP of an instance $Inst$ of WSFJSP with an empty set of logic inequalities
Output: Feasible solution of RMP
Initialize $U := \bar{V}$; // Set of operations that have not been sequenced
Initialize $L_m := 0 \; \forall m \in M$; // Load of machine $m$
Initialize $r^m_{i_1} := s^m_{0_1,i_1} \; \forall i \in I, m \in M_{i_1}$; // Earliest starting time of operation $i_1$ of
    job $i$ on machine $m$
Initialize $r^m_{i_j} := \infty \; \forall i_j \in \bar{V} \setminus \{i_1 | i \in I\}, m \in M_{i_j}$; // Earliest starting time of operation
    $i_j$ of job $i$ on machine $m$
Initialize $a_b(m) := 0_1 \; \forall m \in M$; // Last operation sequenced on machine $m$
Initialize $c_i := 0 \; \forall i \in I$; // Completion time of the operation of job $i$ that has
    been completed last
repeat
    Determine $C^* := \min_{i_j \in U, m \in M_{i_j}} r^m_{i_j} + p^{m,min}_{i_j}$; // Smallest possible completion time
        of operations that have not been sequenced on their eligible
        machines
    Let $m^*$ denote a machine on which $C^*$ is a possible completion time;
    Among all operations that have not been sequenced, $i_j \in U$, with $r^{m^*}_{i_j} < C^*$, choose an
      operation $i^*_j$ based on a priority rule and sequence it on machine $m^*$, starting its processing
      at time $r^{m^*}_{i^*_j}$;
    Update $L_{m^*} := r^{m^*}_{i^*_j} + p^{m^*,min}_{i^*_j}$, $a_b(m^*) := i^*_j$, and $c_{i^*} := L_{m^*}$;
    // Update earliest starting times of operations that have not been
        sequenced
    forall $i \in I$ do
        if $i = i^*$ and $i^*_j$ has a succeeding operation then
            update $r^m_{i^*_{j+1}} := \max\{L_{m^*}, L_m + s^m_{a_b(m),i^*_{j+1}}\} \; \forall m \in M_{i^*_{j+1}}$;
        end
        else if $i \neq i^*$ then
            $r^{m^*}_{i_k} := \max\{c_i, L_{m^*} + s^{m^*}_{i^*_j,i_k}\} \; \forall i_k \in U$ with $r^{m^*}_{i_k} < \infty$;
        end
    end
    Update $U := U \setminus \{i^*_j\}$;
until $U = \emptyset$;
```

Algorithm 2 Obtaining an initial allocation and sequencing decision

4.1.2 Worker Assignment

After having determined an initial allocation and sequencing decision, i.e. a feasible solution of a RMP as, for instance, constructed by Algorithm 2, we proceed by computing a corresponding feasible worker assignment and the resulting solution of the given instance of the WSFJSP with a *beam search* approach.

In general, beam search uses a graph representation of a solution process and applies breadth-first search with a filtering process to only expand the β (*beam width*) most promising nodes of the graph in each iteration. It was first used by Lowerre (1976). Our implementation of a beam search approach for the subproblem first sorts the operations of all jobs in non-decreasing order of the points in time when their processing is started in a feasible solution of a RMP. It then proceeds by assigning workers to the operations (and potentially shifting the corresponding starting times of the operations based on the correct values of the processing times) in this order by constructing a search tree in a breadth-first search manner as illustrated in Fig. 5. That is, the algorithm decides on the worker assignment for a single operation on

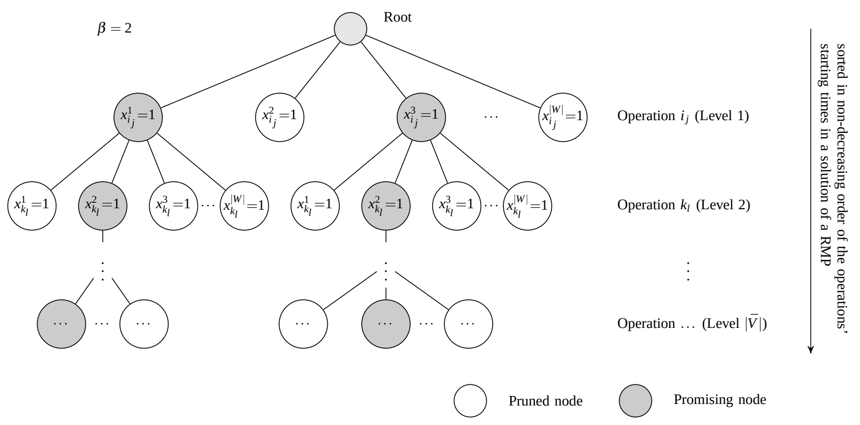

$\beta = 2$
Root
$x_{i_j}^1=1$
$x_{i_j}^2=1$
$x_{i_j}^3=1$
$x_{i_j}^{|W|}=1$
Operation i_j (Level 1)
$x_{k_l}^1=1$
$x_{k_l}^2=1$
$x_{k_l}^3=1$
$x_{k_l}^{|W|}=1$
Operation k_l (Level 2)
Operation … (Level $|\bar{V}|$)
sorted in non-decreasing order of the operations' starting times in a solution of a RMP
Pruned node
Promising node

Fig. 5 Illustration of the beam search algorithm for the worker assignment problem

each level of the tree and hereafter selects the β most promising nodes for further consideration, while all other nodes are pruned. The root node represents a situation where no worker has been assigned. A node is evaluated based on the earliest possible completion time of the corresponding operation that results from the worker assignment decisions given in the node. On the last level of the search tree, the algorithm determines the objective function value, i.e. the makespan or the total tardiness, of the β most promising nodes and selects a solution with minimum value.

4.2 Decomposition Based Improvement Procedure

Our improvement procedure is based on dividing the relevant time horizon into time windows of equal length (see Section 4.2.1). Its main idea is to iterate over pairs of time windows and – for each pair – make use of a given reference solution of a RMP (which is altered in the course of the procedure, starting with the solution determined by Algorithm 2) to decide on fixing a subset of variables (1), i.e. fixing subsequences of operations on machines. The resulting problems are referred to as *fixed relaxed master problems* (FRMPs, see Section 4.2.2). They have a reduced number of free variables, which allows to quickly determine (potentially improved) feasible solutions with a standard MIP solver by solely reoptimizing the subsequences of operations that are (started to be) processed within the time windows in the reference solution. The worker assignment decisions are handled by solving the resulting subproblems either heuristically by the beam search approach introduced in Section 4.1.2 or as in our exact approach.

4.2.1 Time Windows

The length *twLen* of the time windows is a crucial parameter of our improvement procedure. On one hand, large time windows result in a large amount of free variables and, consequently, induce a relatively large computational effort needed to solve the resulting FRMPs with a MIP solver. On the other hand, the solution space is less restricted when considering large time windows, which allows for potentially computing high quality solutions. In order to balance this trade-off, it is reasonable to take account of the average processing time of all operations when computing *twLen*, so that we define a multiset $P := \{p_{i_j}^{m,w} | i_j \in \bar{V}, m \in M_{i_j}, w \in W, 0 < p_{i_j}^{m,w} < \infty\}$, and set

$$twLen = \left\lceil \frac{\sum\limits_{p\in P} p}{|P|} \right\rceil .$$

We then define time windows $tw_i := [(i-1)\cdot twLen, i\cdot twLen]$, $i = 1,2,\ldots$. The number of relevant time windows is not fixed, but is implicitly specified by the current reference solution within our improvement procedure (see Section 4.2.3).

4.2.2 Fixed Relaxed Master Problem

Algorithm 3 presents our method of determining the variables (1) that are considered as fixed in the FRMP that is based on a reference solution Sol_{ref}^{RMP} of a RMP for a given pair of time windows tw_a and tw_b, $a \neq b$. Here, $\mathbf{t^{ref}} := (t_{i_j}^{ref} | i_j \in V)$ refers to the time variables of Sol_{ref}^{RMP}, while $\mathbf{y^{ref}} := (y_{i_j,k_l}^{ref,m} | i_j \in V, k_l \in V_{i_j}, m \in M_{i_j,k_l})$ denotes the vector of variables (1) of Sol_{ref}^{RMP}. The algorithm iterates over all machines and all pairs of operations that are consecutively sequenced on these machines in the given solution (loop 4–18). If, for a given pair of operations, none of the starting times of these operations lie within one of the time windows (line 7), the corresponding variable (1) will later be fixed to one (line 8). Based on this decision, a number of additional, directly related variables (1) will necessarily later be fixed to zero (lines 9–16). Within the algorithm, the variables that will later be fixed are stored in the sets F_0 (fix to zero) and F_1 (fix to one) via their indices.

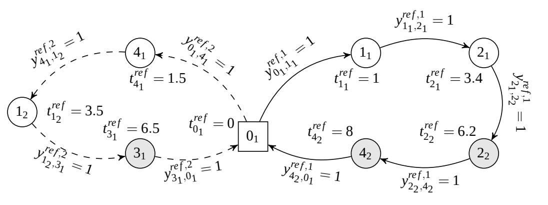
Consider an exemplary reference solution of a RMP as given in Fig. 6 (refer to Fig. 3 for or an explanation of the illustrations) and assume that the time windows $tw_1 = [0,3]$ and $tw_2 = [3,6]$ are considered. When, for example, analyzing $y_{2_2,4_2}^{ref,1}$, the algorithm inserts $(2_2, 4_2, 1)$ into the set F_1, i.e. the decision to process operation 4_2 directly after operation 2_2 on machine 1 will later be fixed, because both starting times $t_{2_2}^{ref}$ and $t_{4_2}^{ref}$ lie outside of the considered time windows in the reference solution. The set F_0 is updated accordingly. When, on the other hand, analyzing $y_{2_1,2_2}^{ref,1}$, no subsequence is fixed, because $t_{2_1}^{ref}$ lies inside of time window tw_2.

In order to obtain the mathematical model of the FRMP based on a given reference solution Sol_{ref}^{RMP} and the result of Algorithm 3, the following constraints are added to the RMP:

$$y_{i_j,k_l}^{m} = 1 \quad \forall\, (i_j, k_l, m) \in F_1,$$

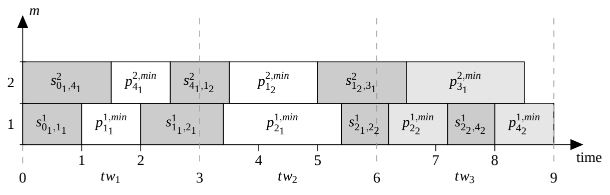
```
   Input: Solution $Sol_{ref}^{RMP}$ ($\mathbf{y^{ref}}$, $\mathbf{t^{ref}}$) of a RMP of an instance $Inst$ of WSFJSP, time windows $tw_a$
          and $tw_b$, $a \neq b$
   Output: Sets $F_0$ and $F_1$
1  Initialize $F_1 := \emptyset$; // Set of indices of variables that will be fixed to one
2  Initialize $F_0 := \emptyset$; // Set of indices of variables that will be fixed to zero
3  Initialize $\hat{t}_{i_j} := t_{i_j}^{ref}$ $\forall i_j \in V$; // Auxiliary variables
4  forall $i_j \in V$, $k_l \in V_{i_j}$, $m \in M_{i_j,k_l}$ with $y_{i_j,k_l}^{ref,m} = 1$ do
       // Update auxiliary variables if operation $0_1$ is involved
5      if $k_l = 0_1$ then $\hat{t}_{k_l} := \hat{t}_{i_j} + p_{i_j}^{m,min}$;
6      else if $i_j = 0_1$ then $\hat{t}_{i_j} := 0$;
       // Check time windows $tw_a$ and $tw_b$ and update $F_0$ and $F_1$
7      if $\hat{t}_{i_j} \notin tw_a$ and $\hat{t}_{i_j} \notin tw_b$ and $\hat{t}_{k_l} \notin tw_a$ and $\hat{t}_{k_l} \notin tw_b$ then
8          $F_1 := F_1 \cup \{(i_j, k_l, m)\}$;
9          forall $q_v \in V_{i_j} \setminus \{k_l\}$ do
10             if $i_j \neq 0_1$ then $F_0 := F_0 \cup \{(i_j, q_v, m') | m' \in M_{i_j,q_v}\}$;
11             else if $m \in M_{q_v}$ then $F_0 := F_0 \cup \{(0_1, q_v, m)\}$;
12         end
13         forall $q_v \in \tilde{V}_{k_l} \setminus \{i_j\}$ do
14             if $k_l \neq 0_1$ then $F_0 := F_0 \cup \{(q_v, k_l, m') | m' \in M_{q_v,k_l}\}$;
15             else if $m \in M_{q_v}$ then $F_0 := F_0 \cup \{(q_v, 0_1, m)\}$;
16         end
17     end
18 end
```

Algorithm 3 Determining the set of variables to be fixed



$y_{1_1,2_1}^{ref,1} = 1$
$y_{4_1,1_2}^{ref,2} = 1$
$y_{0_1,4_1}^{ref,2} = 1$
$y_{0_1,1_1}^{ref,1} = 1$
4_1
1_1
2_1
$t_{4_1}^{ref} = 1.5$
$t_{1_1}^{ref} = 1$
$t_{2_1}^{ref} = 3.4$
$y_{2_1,2_2}^{ref,1} = 1$
1_2
$t_{1_2}^{ref} = 3.5$
$t_{3_1}^{ref} = 6.5$
$t_{0_1}^{ref} = 0$
0_1
$t_{4_2}^{ref} = 8$
$t_{2_2}^{ref} = 6.2$
3_1
4_2
2_2
$y_{1_2,3_1}^{ref,2} = 1$
$y_{3_1,0_1}^{ref,2} = 1$
$y_{4_2,0_1}^{ref,1} = 1$
$y_{2_2,4_2}^{ref,1} = 1$

(a) Supporting graph



(b) Gantt chart

Fig. 6 Exemplary reference solution of a RMP

$$y_{i_j,k_l}^m = 0 \quad \forall (i_j, k_l, m) \in F_0.$$

A MIP solver will therefore reoptimize the subsequences of operations that are started to be processed within the given time windows in Sol_{ref}^{RMP} when being called on this FRMP.

4.2.3 Heuristic Framework

We are now ready to present the details of our decomposition based improvement procedure in Algorithm 4. The algorithm stores solutions of RMPs, FRMPs, and the WSFJSP instance in the data structures defined in Table 3. To ease the notation, we

Table 3 Solutions stored by the heuristic framework

Sol^{RMP}_{best}	Best solution found for RMP
Sol^{RMP}_{ref}	Current reference solution of RMP
Sol^{FRMP}	Current solution of FRMP
Sol	Current solution of WSFJSP
Sol_{best}	Best solution found for WSFJSP

refer to the value of the objective function of a solution Sol by using an additional label, i.e. $\ddot{Sol}$, throughout the algorithm.

In the *initialization phase* (step 1) of the algorithm, the length of the time windows is calculated as described in Section 4.2.1, the RMP is initialized with an empty set of logic inequalities, and an initial solution of the WSFJSP instance is determined as described in Section 4.1. During the process of generating this solution, the reference solution Sol^{RMP}_{ref} is set to the solution determined by Algorithm 2. It is potentially altered multiple times in the course of the algorithm (steps 2e i and 2f). In the *iteration phase* (step 2), the algorithm first computes the current pair of time windows (step 2a) based on the reference solution. If all pairs have been traversed, the algorithm exits (step 2a iii). Otherwise, the current pair of time windows is discarded with probability $1-\rho_{tw}$. Then, in steps 2b and 2c, the FRMP is constructed based on the current set of logic inequalities (see Section 4.2.2) and a solution Sol^{FRMP} of the resulting FRMP is computed by some algorithm. If promising, i.e. if $\ddot{Sol}^{FRMP} < \ddot{Sol}_{best}$, a feasible solution of the subproblem is computed by some algorithm (step 2e ii) and the best known solution of the WSFJSP instance is potentially updated (step 2e iii). A logic inequality is generated and added to the RMP (step 2e iv) if this is necessary. This process of generating the FRMP and solving the corresponding subproblem is repeated at most λ times for a given pair of time windows (steps 2e iii and 2e v).

5 Computational Study

In order to evaluate the performance of our solution approaches, we conducted extensive computational tests. The tests were performed on a PC with an Intel® Core™ i7-4770 CPU, running at 3.4 GHz, with 16 GB of RAM under a 64-bit version of Windows 8. All algorithms were implemented in C++ (Microsoft Visual Studio 2015). We used IBM ILOG CPLEX in version 12.7 as a MIP solver.

We implemented six approaches:

1. E-DM refers to the exact branch-and-cut approach (Algorithm 1), using CPLEX as a standard solver.

```
Input: Instance $Inst$ of WSFJSP, parameters $\rho_{tw}$ and $\lambda$
Output: Solution $Sol_{best}$ of $Inst$
1. Initialization:
   (a) Compute and initialize $twLen$.
   (b) Initialize the RMP of $Inst$ with an empty set of logic inequalities.
   (c) Determine $Sol^{RMP}_{ref}$ by calling Algorithm 2 on the RMP. Set $Sol^{RMP}_{best} := Sol^{RMP}_{ref}$.
   (d) Compute a feasible solution for the subproblem based on $Sol^{RMP}_{ref}$ with the beam search
       approach of Section 4.1.2. Retrieve the corresponding feasible solution $Sol$ of the
       WSFJSP instance. Set $Sol_{best} := Sol$.
   (e) Initialize $counter := 1$, $a := 1$, and $b := 1$.
2. Iteration:
   (a) Define current pair of time windows as follows:
       i. Set $b := b+1$.
       ii. If $(b-1) \cdot twLen$ is not smaller than the completion time of the last operation which
           is completed in $Sol^{RMP}_{ref}$, set $a := a+1$ and $b := a+1$.
       iii. If $(b-1) \cdot twLen$ is not smaller than the completion time of the last operation which
            is completed in $Sol^{RMP}_{ref}$, exit algorithm.
       iv. Define $tw_a := [(a-1) \cdot twLen, a \cdot twLen]$ and $tw_b := [(b-1) \cdot twLen, b \cdot twLen]$.
       v. Go to step 2a i with probability $1-\rho_{tw}$.
   (b) Determine $F_0$ and $F_1$ by calling Algorithm 3 on $Sol^{RMP}_{ref}$.
   (c) Construct FRMP based on RMP (including the current set of logic inequalities), $F_0$, and
       $F_1$. Compute a feasible solution $Sol^{FRMP}$.
   (d) If $\ddot{Sol}^{FRMP} \leq \ddot{Sol}^{RMP}_{best}$, update $Sol^{RMP}_{best} := Sol^{FRMP}$.
   (e) If $\ddot{Sol}^{FRMP} < \ddot{Sol}_{best}$, do the following:
       i. Update $Sol^{RMP}_{ref} := Sol^{FRMP}$.
       ii. Compute a feasible solution for the subproblem based on $Sol^{RMP}_{ref}$ and retrieve the
           corresponding WSFJSP solution $Sol$.
       iii. If $\ddot{Sol} < \ddot{Sol}_{best}$, update $Sol_{best} := Sol$. Else, set $counter := counter + 1$.
       iv. If $\ddot{Sol}^{RMP}_{ref} < \ddot{Sol}$, generate the corresponding logic inequality (10) or (16) and add it
           to RMP.
       v. If $counter \leq \lambda$, go to step 2b.
   (f) Update $Sol^{RMP}_{ref} := Sol^{RMP}_{best}$ and set $counter := 1$.
   (g) Go to step 2a.
```

Algorithm 4 Heuristic framework for WSFJSP

2. E-IM refers to calling CPLEX in its standard settings on an integrated MIP model for WSFJSP. This approach is intended to be a benchmark for evaluating the performance of E-DM. The integrated model is presented in Section 5.1.
3. H-DM refers to the heuristic framework (Algorithm 4), using CPLEX in step 2c (determining feasible solutions of FRMPs, 10 seconds time limit) and in step 2e ii (determining feasible solutions of subproblems, the time limit is set to the remaining time with respect to the limit set for the overall heuristic framework).
4. H-DMB refers to the heuristic framework (Algorithm 4), using CPLEX in step 2c (10 seconds time limit) and applying the beam search approach of Section 4.1.2 in step 2e ii.
5. H-HIER refers to an adapted version of the heuristic framework (Algorithm 4, CPLEX time limit of 10 seconds in step 2c), where the subproblem is solved only once at the very end of the procedure. That is, steps 1d and 2e are replaced by a single call of the beam search approach on the best reference solution after having iterated over all relevant pairs of time windows.

6. LS refers to a local search procedure inspired by Mastrolilli and Gambardella (2000), which we use as a benchmark heuristic when evaluating the heuristic framework. It is presented in detail in Section 5.3.3.

Whenever calling CPLEX on a FRMP within H-DM, H-DMB, or H-HIER, we provide the vector $\mathbf{y}^{\mathbf{ref}}$ of the corresponding solution Sol_{ref}^{RMP} as a warm start. Similarly, in H-DM, when solving a subproblem with CPLEX, we provide a lower bound on the optimal objective function value, i.e. the objective function value of the corresponding FRMP. Furthermore, we provide CPLEX with the vector $\mathbf{t}^{\mathbf{ref}}$ of the corresponding solution Sol_{ref}^{RMP}, which allows to quickly check if this partial solution remains feasible for the subproblem.

We set the large integer B to $\sum_{i_j \in \bar{V}} (p_{i_j}^{max} + s_{i_j}^{max})$ in constraint (7) of the MP and to $\sum_{i_j \in \bar{V}} (\tilde{p}_{i_j}^{max} + s_{i_j}^{max})$ in constraints (25) and (28) of the subproblem as well as in the relevant constraints of the integrated MIP model. Here, $s_{i_j}^{max} := \{s_{k_l,i_j}^m | k_l \in \tilde{V}_{i_j}, m \in M_{i_j,k_l}\}$, $p_{i_j}^{max} := \max\{p_{i_j}^{m,min} | m \in M_{i_j}\}$, and $\tilde{p}_{i_j}^{max} := \max\{p_{i_j}^{m,w} | p_{i_j}^{m,w} \neq \infty, m \in M_{i_j}, w \in W\}$, for all $i_j \in \bar{V}$.

5.1 Integrated MIP model

E-IM is based on an integrated MIP model for WSFJSP. This model is in line with the formulations of the MP and the subproblem in Section 3, but replaces the variables (19) with variables

$$x_{i_j}^{m,w} := \begin{cases} 1, & \text{if operation } i_j \text{ is processed by worker } w \text{ on machine } m \\ 0, & \text{else} \end{cases} \qquad \forall i_j \in \bar{V}, m \in M_{i_j}, w \in W.$$

The model for minimizing the makespan is as follows.

$$\min C_{max} \tag{40}$$

$$\text{s.t. } (4)-(6),\ (9),\ (11)-(13),\ (29),\ (32)-(33),$$

$$t_{i_{q_i}} + \sum_{m \in M_{i_{q_i}}} \sum_{w \in W} x_{i_{q_i}}^{m,w} \cdot p_{i_{q_i}}^{m,w} \leq C_{max} \qquad \forall i \in I, \tag{41}$$

$$\sum_{i_j \in \{a_b \in \tilde{V}_{k_l} | m \in M_{a_b}\}} y_{i_j,k_l}^m = \sum_{w \in W} x_{k_l}^{m,w} \qquad \forall k_l \in \bar{V}, m \in M_{k_l}, \tag{42}$$

$$t_{i_j} + s_{i_j,k_l}^m + \sum_{w \in W} x_{i_j}^{m,w} \cdot p_{i_j}^{m,w} - t_{k_l} \leq \left(1 - y_{i_j,k_l}^m\right) B \qquad \forall i_j \in V, k_l \in \bar{V}_{i_j}, m \in M_{i_j,k_l}, \tag{43}$$

$$t_{i_j} + \sum_{m \in M_{i_j}} \sum_{w \in W} x_{i_j}^{m,w} \cdot p_{i_j}^{m,w} \leq t_{i_{j+1}} \qquad \forall i_j \in \bar{V} \text{with } j \leq q_i - 1, \tag{44}$$

$$x_{i_j,k_l}^w \geq \sum_{m \in M_{i_j}} x_{i_j}^{m,w} + \sum_{m \in M_{k_l}} x_{k_l}^{m,w} - 1 \qquad \forall i_j, k_l \in \bar{V}, i_j \neq k_l, w \in W, \tag{45}$$

$$t_{i_j} + \sum_{m \in M_{i_j}} \sum_{w \in W} x_{i_j}^{m,w} \cdot p_{i_j}^{m,w} - t_{k_l} \leq B \cdot v_{i_j,k_l} \qquad \forall i_j, k_l \in \bar{V}, i_j \neq k_l, \tag{46}$$

$$x_{0_1}^{m,w} = 0 \qquad \forall m \in M_{i_j}, w \in W, \qquad (47)$$

$$x_{i_j}^{m,w} \in \{0,1\} \qquad \forall i_j \in \bar{V}, m \in M_{i_j}, w \in W. \qquad (48)$$

The objective function and most constraints are either identical to the ones in the MP and the subproblem or have been adapted in a straightforward manner. Constraint (42) connects the sequencing variables with the worker assignment variables. The MIP model for minimizing the total tardiness modifies the objective function and constraint (41) in analogy to Section 3. We do not present it in detail for the sake of brevity.

5.2 Instance Generation and Parameter Settings

Our computational tests were performed on randomly generated test instances as well as real-world test instances based on data of our industry partner.

5.2.1 Random Testbed

Our random testbed is composed of three classes of instance sets. These classes differ in the size of the included instances (small, medium, and large), which is expressed by an identifier $size \in \{s,m,l\}$. Each class is composed of eight sets of test instances with differing numbers of jobs $|I|$, machines $|M|$, and workers $|W|$. Each set is composed of ten instances and is denoted by $size_{|I|,|M|,|W|}$, where $size \in \{s,m,l\}$. For each instance, the number of operations q_i of jobs $i \in I$, the number of eligible machines $|M_{i_j}|$ for operations $i_j \in O_i$ of jobs $i \in I$, and the integer setup times s_{i_j,k_l}^m (including s_{0_1,k_l}^m) when processing operation $k_l \in O_k$ of job $k \in I$ immediately after operation $i_j \in O_i$ of job $i \in I$ on machine $m \in M_{i_j,k_l}$ are drawn from uniform distributions over the intervals given in Table 4. Our process of generating the processing times of the operations is

Table 4 Random testbed

small instances					medium instances					large instances				
set	q_i	$\lvert M_{i_j}\rvert$	p_{i_j}	s_{i_j,k_l}^m	set	q_i	$\lvert M_{i_j}\rvert$	p_{i_j}	s_{i_j,k_l}^m	set	q_i	$\lvert M_{i_j}\rvert$	p_{i_j}	s_{i_j,k_l}^m
$s_{3,2,2}$	$[2,3]$	$[2,2]$	$[3,6]$	$[1,4]$	$m_{10,5,4}$	$[2,3]$	$[2,3]$	$[4,8]$	$[1,5]$	$l_{20,15,12}$	$[5,10]$	$[4,6]$	$[5,15]$	$[1,10]$
$s_{4,2,2}$	$[2,3]$	$[2,2]$	$[3,6]$	$[1,4]$	$m_{10,5,5}$	$[2,3]$	$[2,3]$	$[4,8]$	$[1,5]$	$l_{20,15,15}$	$[5,10]$	$[4,6]$	$[5,15]$	$[1,10]$
$s_{5,3,3}$	$[2,3]$	$[2,2]$	$[3,6]$	$[1,4]$	$m_{10,8,6}$	$[4,6]$	$[3,4]$	$[4,8]$	$[1,5]$	$l_{25,20,16}$	$[5,10]$	$[5,7]$	$[5,15]$	$[1,10]$
$s_{6,3,3}$	$[2,3]$	$[2,2]$	$[3,6]$	$[1,4]$	$m_{10,8,8}$	$[4,6]$	$[3,4]$	$[4,8]$	$[1,5]$	$l_{25,20,20}$	$[5,10]$	$[5,7]$	$[5,15]$	$[1,10]$
$s_{7,4,3}$	$[2,3]$	$[2,2]$	$[3,6]$	$[1,4]$	$m_{15,10,8}$	$[4,6]$	$[3,5]$	$[5,10]$	$[1,7]$	$l_{30,20,16}$	$[5,7]$	$[5,7]$	$[5,15]$	$[1,10]$
$s_{7,4,4}$	$[2,3]$	$[2,2]$	$[3,6]$	$[1,4]$	$m_{15,10,10}$	$[4,6]$	$[3,5]$	$[5,10]$	$[1,7]$	$l_{30,20,20}$	$[5,7]$	$[5,7]$	$[5,15]$	$[1,10]$
$s_{8,4,3}$	$[2,3]$	$[2,2]$	$[3,6]$	$[1,4]$	$m_{20,10,8}$	$[4,6]$	$[3,5]$	$[5,10]$	$[1,7]$	$l_{40,20,16}$	$[4,6]$	$[5,7]$	$[5,15]$	$[1,10]$
$s_{8,4,4}$	$[2,3]$	$[2,2]$	$[3,6]$	$[1,4]$	$m_{20,10,10}$	$[4,6]$	$[3,5]$	$[5,10]$	$[1,7]$	$l_{40,20,20}$	$[4,6]$	$[5,7]$	$[5,15]$	$[1,10]$

as follows. We first draw auxiliary integer parameters p_{i_j} for all jobs $i \in I$ and operations $i_j \in O_i$ from uniform distributions over the intervals given in Table 4. Based on these parameters, we construct varying processing times over the corresponding eligible machines $m \in M_{i_j}$ by drawing integer values $p_{i_j}^m$ from uniform distributions over $[\lfloor 0.9 \cdot p_{i_j} + 0.5 \rfloor, \lceil 1.1 \cdot p_{i_j} - 0.5 \rceil]$ and, in the last step, we incorporate dependencies on the workers $w \in W$ by drawing integer values $p_{i_j}^{m,w}$ from uniform distributions over $[\lfloor 0.9 \cdot p_{i_j}^m + 0.5 \rfloor, \lceil 1.1 \cdot p_{i_j}^m - 0.5 \rceil]$. Finally, the integer due dates d_i are drawn from uniform distributions over the interval $[\lfloor \mu_i \cdot \left(1 - \frac{\Phi}{2}\right) + 0.5 \rfloor, \lceil \mu_i \cdot \left(1 + \frac{\Phi}{2}\right) - 0.5 \rceil]$ for all jobs $i \in I$ (cf. Vilcot and Billaut, 2008). Here,

$$\mu_i := \left(1 + \frac{\Omega \cdot |I|}{|M|}\right) \cdot \frac{\sum\limits_{p \in P_i} p}{|P_i|} \quad \forall i \in I,$$

where P_i is a multiset $\{p_{i_j}^{m,w} | i_j \in O_i, m \in M_{i_j}, w \in W, 0 < p_{i_j}^{m,w} < \infty\}$ for all jobs $i \in I$. We set $\Phi = 0.5$ and $\Omega = 0.3$ for all small instances, $\Phi = 0.5$ and $\Omega = 0.4$ for all medium instances, and $\Phi = 0.3$ and $\Omega = 0.5$ for all large instances.

5.2.2 Real-World Instances

As mentioned in Section 1.1, our research is motivated by a real-world problem setting. Our industry partner provided us with data on its production processes, including the processing times of the manufacturing operations on their eligible machines and the sequence-dependent setup times, as well as the customer demands (including due dates) over a time period of several months. Additionally, we received the relevant information on worker qualifications needed to derive all relevant processing times. Based on this data, we constructed ten realistic scheduling scenarios for testing our algorithms. Each scenario consists of a set of jobs relating to the company's product portfolio, including the respective lot sizes and their due dates. Currently, the product portfolio consists of more than one hundred products and the company uses 16 different multi-purpose machines for processing the manufacturing operations. The number of eligible machines for the operations varies between one and two. The machine operators work on a shift-based system. Each shift is staffed with nine workers of similar qualifications. We will therefore assume that nine representative machine workers are available at all time in all of our scenarios.

In line with the random testbed, we denote each scheduling scenario – or problem instance – by $r_{|I|,|M|,|W|}$. Table 5 lists all instances, including their total number of operations. Note that the number of machines is smaller than 16 in the three

Table 5 Real-world instances

instance	$r_{14,13,9}$	$r_{15,11,9}$	$r_{16,14,9}$	$r_{25,16,9}$	$r_{35,16,9}$	$r_{45,16,9}$	$r_{55,16,9}$	$r_{60,16,9}$	$r_{70,16,9}$	$r_{80,16,9}$
$\sum_{i \in I} \lvert O_i \rvert$	34	34	38	63	84	106	136	146	162	188

smallest instances. This is because not all machines are needed for producing the respective products of the scenario. These three scenarios are representative instances that feature planning horizons of at most one week, which is what our industry partner is currently capable of scheduling manually. All remaining instances have been constructed to analyze the capability and performance of our algorithms for larger planning horizons in a real-world context. As mentioned in Section 1.1, our industry partner wishes to compute schedules on a daily basis. Hence, when wanting to take account of these planning horizons, our approaches will have to be embedded into a rolling horizon planning approach. While we will not explicitly analyze this type of approach in our computational study, we note at this point that all of our algorithms are flexible enough to support a rolling horizon procedure that allows rescheduling decisions if the WSFJSP parameters are set to appropriate values. Most important, by appropriately setting the setup times associated to the dummy operation, one can take account of machines that are currently processing operations at the beginning of the planning horizon. Our model also allows for interrupting the processing of these operations (as sometimes decided by experts of the manufacturing companies) by appropriately (re-)defining the set of operations and the associated setup times.

The detailed data of all instances is available in supplementary files that accompany this paper. The processing times of the operations for the representative workers on the machines vary between 15 and 5280 minutes (1090 minutes on average). The setup times range from zero to 300 minutes. Finally, the due dates of the jobs vary between one and six weeks.

5.2.3 Parameter Settings

With respect to the parameters of Algorithms 1 and 4, we set $\rho_s = 0.005$ and $\lambda = 3$. The remaining parameters were set based on the size of the instances (see Table 6). Finally, we set a time limit of 3,600 seconds for each call of an algorithm. Note,

Table 6 Setup of the algorithms

	small instances	medium and real-world instances	large instances
ρ_{tw}	1	0.4	0.2
β	25	16	8

however, that we check the current runtime of the algorithmic framework solely at the beginning of step 2b of Algorithm 4, so that the overall runtime of the framework upon termination may slightly exceed the time limit.

5.3 Results and Analysis

This section presents and analyzes the results of our computational study. Section 5.3.1 analyzes the performance of the exact approaches on the random testbed. Hereafter, Section 5.3.2 presents the corresponding results of the heuristic approaches H-DM, H-DMB, and H-HIER. In Section 5.3.3, we describe LS and evaluate the aforementioned heuristics against this procedure on the random testbed. Section 5.3.4 focuses on the results for the real-world instances.

5.3.1 Exact Approaches

Table 7 presents the computational results for the exact approaches E-IM and E-DM for the small test instances of our random testbed. It includes information about

Table 7 Performance of exact approaches for small instances

	C_{max}				T			
	E-IM		E-DM		E-IM		E-DM	
set	opt. [%]	t_{avg} [s]	opt. [%]	t_{avg} [s]	opt. [%]	t_{avg} [s]	opt. [%]	t_{avg} [s]
$s_{3,2,2}$	100	0.9	100	0.3	100	1	100	0.5
$s_{4,2,2}$	100	47.5	100	13.3	100	58.2	100	19.7
$s_{5,3,3}$	100	36.8	100	5	100	51.1	100	12.4
$s_{6,3,3}$	60	756.7	100	425.7	70	882.8	80	482.6
$s_{7,4,3}$	10	2142.5	70	913.1	20	1952.9	40	944.3
$s_{7,4,4}$	80	1161.6	100	112.2	50	1074.2	90	839.3
$s_{8,4,3}$	-	-	20	2773.1	-	-	-	-
$s_{8,4,4}$	-	-	40	622.7	-	-	10	3132.4

the percentage of instances within each set that were solved to optimality within the given time limit (columns "opt.") as well as the average computational time needed to compute the optimal solutions (columns "t_{avg}") for both objectives, minimizing the makespan C_{max} or the total tardiness T. The decomposition based approach E-DM clearly outperforms the integrated approach E-IM both with respect to the number of instances solved to optimality and the runtimes. While this is true for both objectives, the benefit of using the decomposition based approach is more pronounced in case of minimizing the makespan. The maximum number of logic inequalities generated over all calls of E-DM for the small test instances was 1,503 (235) for the objective of minimizing the makespan (total tardiness). On average, a logic inequality was generated and added to the branch-and-cut process in 29% (38%) of the calls of step 3 of Algorithm 1.

Unfortunately, neither of the exact approaches was able to solve instances of medium or large size to optimality within the time limit. Table 8 therefore focuses on illustrating the percentage of instances for which E-DM found a feasible solution. It can be seen that E-DM returns feasible solutions for instances up to the size of the

Table 8 Capability of finding feasible solutions with E-DM within the time limit of 3,600 seconds

small instances			medium instances		
set	C_{max} [%]	T [%]	set	C_{max} [%]	T [%]
$s_{3,2,2}$	100	100	$m_{10,5,4}$	100	100
$s_{4,2,2}$	100	100	$m_{10,5,5}$	100	100
$s_{5,3,3}$	100	100	$m_{10,8,6}$	90	100
$s_{6,3,3}$	100	100	$m_{10,8,8}$	90	100
$s_{7,4,3}$	100	100	$m_{15,10,8}$	-	-
$s_{7,4,4}$	100	100	$m_{15,10,10}$	-	10
$s_{8,4,3}$	100	100	$m_{20,10,8}$	-	-
$s_{8,4,4}$	100	100	$m_{20,10,10}$	-	-

ones in the set $m_{10,8,8}$ and for one instance in the set $m_{15,10,10}$.

We conclude that, while E-DM is certainly not a reasonable choice when facing large instances of WSFJSP in practice, it clearly outperforms E-IM and is able to compute benchmark solutions that allow to assess the performance of heuristic approaches for instances of medium size.

5.3.2 Heuristic Approaches

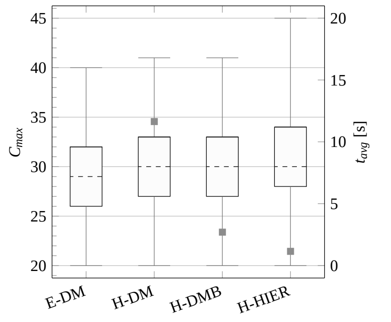
The performance of the heuristic approaches H-DM, H-DMB, and H-HIER in comparison to E-DM is illustrated in Table 9 for the small test instances. For each objective function, each instance set, and each solution approach, the table presents information on the average objective function values of the best solutions returned by the respective algorithms (columns "C_{max}^{avg}" and "T^{avg}") as well as the average runtimes for computing these solutions (columns "t_{avg}"). Entries "tl" denote average computational times that correspond to or exceed the time limit of 3,600 seconds. Bold entries highlight the best heuristic approaches with respect to the average solution quality.

Table 9 Performance of heuristic approaches for small instances

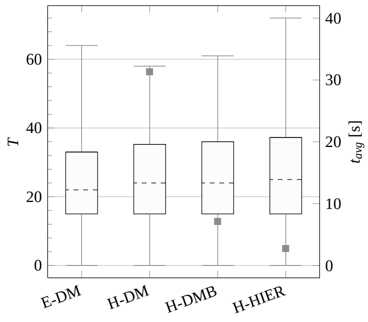
	C_{max}								T							
	E-DM		H-DM		H-DMB		H-HIER		E-DM		H-DM		H-DMB		H-HIER	
set	C_{max}^{avg}	t_{avg} [s]	C_{max}^{avg}	t_{avg} [s]	C_{max}^{avg}	t_{avg} [s]	C_{max}^{avg}	t_{avg} [s]	T^{avg}	t_{avg} [s]	T^{avg}	t_{avg} [s]	T^{avg}	t_{avg} [s]	T^{avg}	t_{avg} [s]
$s_{3,2,2}$	23.9	0.3	**24.2**	0.2	24.3	0.2	24.5	0.1	8.6	0.5	**8.6**	0.2	**8.6**	0.2	8.7	0.2
$s_{4,2,2}$	30	13.3	**30.4**	0.4	30.5	0.4	30.7	0.3	16.2	19.7	**17.3**	0.5	17.4	0.4	17.4	0.3
$s_{5,3,3}$	27.1	5	**27.7**	0.5	**27.7**	0.4	28	0.3	14.3	12.4	**15.4**	1	15.5	0.9	16	0.4
$s_{6,3,3}$	29.9	425.7	**31**	1.5	31.1	1.2	31.8	0.9	23.8	1106.1	26.3	2.3	**25.8**	1.8	26.6	1.3
$s_{7,4,3}$	31	1719.2	**32.1**	22	32.9	3.8	33.5	1.1	29.3	2537.7	**30.9**	42.5	31.4	9.7	35.3	3.2
$s_{7,4,4}$	27.5	112.2	**28.2**	1.4	**28.2**	1.3	28.4	0.8	24.4	1115.4	**26.2**	6.4	26.5	6.8	26.9	2.7
$s_{8,4,3}$	34.5	3436.6	**35.4**	63.2	36	11.0	37.6	3.4	44.7	tl	**45.7**	187.6	46.6	27.1	53.7	6.5
$s_{8,4,4}$	29.7	2409.1	31.1	4	**31**	3.4	31.1	2.3	34.7	3553.3	**36.6**	10.2	37	10.1	36.8	7.5

Fig. 7 complements Table 9 by presenting boxplots of the objective function values returned by the algorithms (left ordinate) as well as data points (gray squares) on the overall average of the corresponding runtimes (right ordinate) for the heuristic approaches. Each boxplot depicts the first quartile (bottom of the box), the third quartile (top of the box), the median (dotted line within the box), the minimum (bottom whisker), and the maximum (top whisker) of the objective function values.

Based on Table 9 and Fig. 7, we observe that the solution quality of all heuristic approaches is competitive when compared with the (mostly optimal, see Section 5.3.1) solutions computed by E-DM for both objectives. That is, all heuristics provide high quality solutions. H-DM tends to provide the best solutions at the cost of substantially larger running times than the ones of the other heuristic approaches. This is a result of determining optimal solutions to the subproblems within the heuristic framework. Applying beam search (H-DMB) results in slightly worse solutions when compared with the solutions determined by H-DM but clearly pays off with respect to the running times. A similar but less pronounced effect can be observed when comparing H-DMB and H-HIER.



(a) Minimizing the makespan



(b) Minimizing the total tardiness

Fig. 7 Computational results for small instances

Table 10 Performance of heuristic approaches for medium instances

	C_{max}								T							
	E-DM		H-DM		H-DMB		H-HIER		E-DM		H-DM		H-DMB		H-HIER	
set	C_{max}^{avg}	t_{avg} [s]	C_{max}^{avg}	t_{avg} [s]	C_{max}^{avg}	t_{avg} [s]	C_{max}^{avg}	t_{avg} [s]	T^{avg}	t_{avg} [s]	T^{avg}	t_{avg} [s]	T^{avg}	t_{avg} [s]	T^{avg}	t_{avg} [s]
$m_{10,5,4}$	41.9	tl	**42.2**	158.8	43.6	14.3	44.4	5.8	43	tl	**40.4**	543.5	42.4	31.8	45.5	7
$m_{10,5,5}$	38.2	tl	**39.3**	12.5	39.6	11.5	39.7	6.4	22.9	tl	**24.3**	32.9	24.7	26.2	27.3	7
$m_{10,8,6}$	165	tl	57.7	tl	**56.4**	49.6	57.8	13.5	746.3	tl	25.5	3157.4	**17.8**	75.1	19	17.2
$m_{10,8,8}$	198.1	tl	52.9	317.1	**52.6**	32.9	53.4	15.7	656.2	tl	5.8	901.4	**4.6**	53.8	9.3	14.4
$m_{15,10,8}$	-	-	84.2	tl	**78.9**	186.3	80	45.3	-	-	154.5	tl	**66.4**	417.2	84.2	106.5
$m_{15,10,10}$	-	-	80.8	3105.2	**75.7**	118.1	77.7	47.9	2367	tl	64.1	2361.1	**34.5**	255.2	42.8	85.4
$m_{20,10,8}$	-	-	107.3	tl	**102**	525.8	103.3	140.7	-	-	286.2	tl	**164.5**	895.3	178	216.5
$m_{20,10,10}$	-	-	98.6	tl	**93.8**	409.9	**93.8**	156.5	-	-	224	3287.5	**143**	618.7	156.5	233.6

Table 10 presents the computational results for the instances of medium size. In contrast to the results for the small instances, H-DMB now clearly outperforms H-DM, both with respect to solution quality and computational time. Again, this mainly results from solving the subproblems to optimality within H-DM, which causes high computational effort for increasing instance sizes, so that less pairs of time windows are traversed within the given time limit. Moreover, note that all heuristics outperform E-DM for most instance sets. When comparing H-DMB and H-HIER, we observe larger differences in the average solution qualities as in case of the small test instances, particularly for the case of minimizing total tardiness. Nevertheless, the runtimes are in ranges that allow the use of both H-DMB and H-HIER in real-world scenarios. The same results hold for the large test instances, as can be seen in Table 11, where we restrict ourselves to comparing H-DMB and H-HIER due to the above results for the medium instances.

Summing up the above results, the frequency of computing solutions of subproblems within the heuristic framework as well as the algorithms applied to determine these solutions are components of major importance when wanting to balance the trade-off between runtime and solution quality. When facing medium or large instances, it does not pay off to try to solve the subproblems to optimality, but one must make use of a heuristic approach. Calling this approach more often has a positive

Table 11 Performance of H-DMB and H-HIER for large instances

	C_{max}				T			
	H-DMB		H-HIER		H-DMB		H-HIER	
set	C_{max}^{avg}	t_{avg} [s]	C_{max}^{avg}	t_{avg} [s]	T^{avg}	t_{avg} [s]	T^{avg}	t_{avg} [s]
$l_{20,15,12}$	**146.6**	217.3	149.7	54.1	**37.7**	466.7	55.8	113.3
$l_{20,15,15}$	**150.2**	206.1	152	62.5	**32.7**	393.8	53.5	124.9
$l_{25,20,16}$	**141.3**	461.6	144.5	95.3	**17.4**	934.4	32.4	213.2
$l_{25,20,20}$	**137.5**	295.1	139.7	101.1	**29**	480.8	39.8	193.2
$l_{30,20,16}$	**125.2**	357.6	127.7	95.3	**146.8**	889.3	181	208.1
$l_{30,20,20}$	**119**	246	120.1	98.8	**90.4**	721.5	145.6	208.4
$l_{40,20,16}$	**131.8**	588.8	134.8	135.6	**247.6**	1320.4	285.4	306.7
$l_{40,20,20}$	**126.2**	359.8	127.9	117.2	**247.3**	998.1	302.1	287.8

effect on solution quality at the cost of larger runtimes. Moreover, as to be expected, we observe that the staffing level, i.e. the ratio of the number of workers and the number of machines, influences the performance of the algorithms. Large staffing levels allow to compute better solutions with less computational effort.

In order to analyze the sensitivity of our heuristic framework to a change of the initial reference solution of RMP (determined in step 1c of Algorithm 4), we have implemented an additional simple priority rule that can be used in Algorithm 2 (line 10). In contrast to the priority rules described in Section 4.1.1, that make use of most of the relevant problem parameters, this priority rule simply selects the operation with *smallest job index* (SI) among the operations that have not been sequenced and that can start to be processed at the respective point of time. Table 12 compares the performance of the heuristic framework for the case of applying SI and the standard setting introduced in Section 4.1.1. For the sake of brevity, we restrict ourselves to presenting

Table 12 Sensitivity of H-DMB and H-HIER to a change of the initial reference solution

	C_{max}								T							
	standard setting				SI				standard setting				SI			
	H-DMB		H-HIER		H-DMB		H-HIER		H-DMB		H-HIER		H-DMB		H-HIER	
class	C_{max}^{avg}	t_{avg} [s]	C_{max}^{avg}	t_{avg} [s]	C_{max}^{avg}	t_{avg} [s]	C_{max}^{avg}	t_{avg} [s]	T^{avg}	t_{avg} [s]	T^{avg}	t_{avg} [s]	T^{avg}	t_{avg} [s]	T^{avg}	t_{avg} [s]
small	30.2	2.7	30.7	1.2	30.7	2.9	31.5	1.3	26.1	7.1	27.7	2.8	29.8	8.1	31.7	3
medium	67.8	168.5	68.8	54	72.5	120.2	75.8	30.2	62.2	296.7	70.3	85.9	114.9	302.3	133.9	87.4
large	134.7	341.5	137.1	95	166.8	446.3	175.6	122.8	106.1	775.6	137	206.9	355.5	910.3	408.7	228.8

the average objective function values and computational times over all instances of the three instance classes (small, medium, and large) for H-DMB and H-HIER. We observe large differences of the average objective function values, particulary for the case of minimizing total tardiness, so that we conclude that it pays off to make use of a well designed procedure to determine high quality initial allocation and sequencing decisions in our heuristic framework.

In light of the fact that H-DMB has shown to be a promising approach but uses a heuristic to determine solutions for the subproblems in step 2e ii of Algorithm 4, one may wonder if the use of logic inequalities within this setup of the heuristic framework actually pays off. Table 13 therefore illustrates the effect of adding logic inequalities by comparing the standard setting of H-DMB (see Section 4.2.3) with the setting where the generation of logic inequalities, i.e. step 2e iv of Algorithm 4, is disabled. Again, we restrict ourselves to presenting average values for the three

Table 13 Effect of using logic inequalities in H-DMB

	C_{max}					T				
	standard setting			no logic inequalities		standard setting			no logic inequalities	
class	C_{max}^{avg}	t_{avg} [s]	LI [%]	C_{max}^{avg}	t_{avg} [s]	T^{avg}	t_{avg} [s]	LI [%]	T^{avg}	t_{avg} [s]
small	30.2	2.7	39	30.4	3.5	26.1	7.1	47	26.8	8.8
medium	67.8	168.5	91	68.5	157.9	62.2	296.7	92	63.2	280
large	134.7	341.5	95	135	357	106.1	775.6	88	113.7	849.9

classes of instances. For the standard setting, the columns "LI" present the percentage of executions of step 2e of Algorithm 4 that result in the generation of a logic inequality. The results indicate that the use of logic inequalities within H-DMB has a positive effect on the objective function values and tends to reduce (or only slightly increase) runtimes for the small and large (medium) instances.

5.3.3 Evaluation of the Heuristic Framework against a Local Search Heuristic

As indicated above, we implemented a local search approach (LS) inspired by Mastrolilli and Gambardella (2000) in order to gain additional insights into the performance of our heuristic framework. In line with Algorithm 4, LS is called on a feasible allocation and sequencing decision, referred to as Sol^{RMP}, initially determined by Algorithm 2. In the course of the algorithm, Sol^{RMP} is altered in a first-fit manner and initially evaluated by dropping the worker assignment restrictions and making use of the lower bounds of the processing times introduced in Section 3.1. Given some current solution Sol^{RMP}, LS restricts the search process by computing a *critical path* of the so called *solution graph* of this solution as described by Mastrolilli and Gambardella (2000) (see also Błażewicz et al., 2007), where sequence-dependent setup times are taken account of in a straightforward manner. Basically, a critical path identifies a sequence of operations, any of which fulfills the property that a delayed start of its processing would immediately cause an increase of the makespan (without considering worker assignment restrictions) under the given allocation and sequencing decision. Note that we make use of the makespan criterion independently of the considered objective function when computing critical paths. A neighbor of the current solution Sol^{RMP} is then determined by moving an operation of the critical path to another feasible position on the same machine or to a feasible position in the sequence of operations on any other eligible machine. As outlined above, the objective function value (makespan or total tardiness) of a neighbor (as well as the initial solution determined by Algorithm 2) is first evaluated based on the lower bounds of the processing times. If this objective function value is smaller than the one of the currently best known solution of the WSFJSP instance (including worker restrictions), a feasible worker assignment and the corresponding objective function value is determined with the beam search approach of Section 4.1.2. If the resulting objective function value remains smaller than the one of the currently best WSFJSP solution, Sol^{RMP} and the overall best WSFJSP solution are updated and LS proceeds by computing a critical path for Sol^{RMP}. LS terminates if none of the neighbors results in an improved objective function value.

Table 14 illustrates the average runtimes and objective function values determined by LS in comparison with H-DMB and H-HIER, i.e. the most promising approaches identified in Section 5.3.2, for the three instance classes. As can be seen, H-DMB and

Table 14 Performance of LS on random testbed

	C_{max}						T					
	H-DMB		H-HIER		LS		H-DMB		H-HIER		LS	
class	C_{max}^{avg}	t_{avg} [s]	C_{max}^{avg}	t_{avg} [s]	C_{max}^{avg}	t_{avg} [s]	T^{avg}	t_{avg} [s]	T^{avg}	t_{avg} [s]	T^{avg}	t_{avg} [s]
small	30.2	2.7	30.7	1.2	32	0.1	26.1	7.1	27.7	2.8	34.4	0.1
medium	67.8	168.5	68.8	54	71.4	10.7	62.2	296.7	70.3	85.9	119.8	7.4
large	134.7	341.5	137.1	95	138.7	66.2	106.1	775.6	137	206.9	234.4	87.6

H-HIER outperform LS with respect to the average objective function values. This effect is especially pronounced for the objective of minimizing total tardiness, which is probably caused by restricting LS to compute neighbors based on critical paths as outlined above. The average computational times of LS are smaller than the ones of H-DMB and H-HIER. However, as our practical application allows runtimes in the ranges of the ones of the latter approaches, this effect is less of an issue. Hence, the remainder of this computational study focusses on the decomposition based solution approaches.

5.3.4 Results for Real-World Instances

Table 15 presents the results on the performance of E-DM as well as the heuristic approaches for the real-world problem instances. For each instance and objective, the table provides results on the objective function values of the solutions determined by the algorithms. Bold elements highlight the best approaches in each row of the table. Note that, in contrast to the results for the random testbed, the objective function values represent minutes of real time (see Section 5.2.2).

Recall that the three smallest instances are representative instances that our industry partner is currently capable of scheduling manually. For all of these instances, H-DM or H-DMB are able to compute solutions in at most 26 seconds while performing similar to the exact approach E-DM and, in case of H-DM, even determining optimal solutions for two instances when minimizing total tardiness. Even in case of medium time horizons, H-DM is able to determine optimal solutions for minimizing total tardiness within reasonable time, i.e. instances $r_{25,16,9}$, $r_{35,16,9}$ and $r_{55,16,9}$.

Table 15 Computational results for real-world instances

	C_{max}								T							
	E-DM		H-DM		H-DMB		H-HIER		E-DM		H-DM		H-DMB		H-HIER	
inst.	C_{max}	t [s]	C_{max}	t [s]	C_{max}	t [s]	C_{max}	t [s]	T	t [s]	T	t [s]	T	t [s]	T	t [s]
$r_{14,13,9}$	**6256**	tl	**6256**	14.3	**6274**	15.9	6676	10.2	**0***	18.5	**0***	3.6	170	4.2	1293	2.2
$r_{15,11,9}$	5375	tl	**5272**	13.5	5392	17.5	5475	3.3	**0***	6.8	**0***	0.4	**0***	0.4	**0***	0.2
$r_{16,14,9}$	**7584**	tl	**7584**	18.7	**7584**	3	**7584**	0.2	**864**	tl	**864**	25.2	**864**	5.2	**864**	0.2
$r_{25,16,9}$	8392	tl	**8180**	2190.4	8551	157.2	11596	66.2	2150	tl	**0***	33.3	227	36.8	1716	12.1
$r_{35,16,9}$	11993	tl	**9045**	tl	9333	358.7	9572	76.6	19337	tl	**0***	50.9	**0***	18.8	322	3
$r_{45,16,9}$	25019	tl	**12722**	tl	12820	378.6	12884	95.5	105019	tl	233	822.9	**0***	40.9	761	11.7
$r_{55,16,9}$	29126	tl	15288	tl	**14608**	916.1	16391	225.4	-	-	**0***	1906.4	**0***	310	289	20.5
$r_{60,16,9}$	33993	tl	18953	tl	**17180**	1444.4	18953	253.1	327669	tl	947	tl	**190**	1300.8	1900	14.9
$r_{70,16,9}$	34716	tl	21202	tl	**19590**	1691.2	21202	280.6	353140	tl	4675	tl	**669**	823.7	878	67.8
$r_{80,16,9}$	-	-	23492	tl	**21869**	2694.4	23998	490.7	-	-	10230	tl	**4670**	1152.5	5880	105.8

*: optimal objective function value

When wanting to make quick decisions for medium and large time horizons, however, it seems appropriate to make use of H-DMB or H-HIER, both of which are capable of computing high quality solutions. For these instances, H-DMB clearly outperforms the other approaches with respect to solution quality, which is in line with our results for the random testbed.

Summing up, we can conclude that our algorithms are well suited for daily use when schedules for time horizons of about one week must be computed at our industry partner. Furthermore, when using H-DMB and H-HIER, our industry partner will be able to make scheduling decisions for demand forecasts covering several weeks or months.

6 Summary

In this paper, we have introduced a worker constrained flexible job shop scheduling problem with sequence-dependent setup times that takes account of heterogeneous machine operator qualifications. We have analyzed two objective functions, minimizing the makespan and the total tardiness, and proposed to solve the problem in a branch-and-cut framework by decomposing it into a vehicle routing problem with precedence constraints (master problem) and a worker assignment problem (subproblem) that are connected via logic inequalities. In addition to this exact approach, we have presented decomposition based heuristic approaches. In an extensive computational study, we have shown that our exact decomposition approach outperforms an integrated approach and that it allows to compute benchmark solutions for assessing the performance of heuristic approaches for instances of medium size. Our heuristic approaches have shown to provide high-quality solutions within reasonable time and have proven well suited for daily use at our industry partner, who provided us with real-world data. When setting up the decomposition based heuristics, the decisions on the frequency of computing solutions of subproblems as well as on the algorithms applied to determine these solutions are of major importance for finding a suitable trade-off between runtime and solution quality.

Acknowledgements This work has been supported by the European Union and the state North Rhine-Westphalia through the European Fund for Regional Development (EFRD). It has been conducted as part of the project "EKPLO: Echtzeitnahes kollaboratives Planen und Optimieren" (EFRE-0800463).

References

Allahverdi A (2015) The third comprehensive survey on scheduling problems with setup times/costs. European Journal of Operational Research 246(2):345–378

Allahverdi A, Soroush HM (2008) The significance of reducing setup times/setup costs. European Journal of Operational Research 187(3):978–984

Allahverdi A, Gupta JND, Aldowaisan T (1999) A review of scheduling research involving setup considerations. Omega 27(2):219–239

Allahverdi A, Ng CT, Cheng TCE, Kovalyov MY (2008) A survey of scheduling problems with setup times or costs. European Journal of Operational Research 187(3):985–1032

Bagheri A, Zandieh M (2011) Bi-criteria flexible job-shop scheduling with sequence-dependent setup times—variable neighborhood search approach. Journal of Manufacturing Systems 30(1):8–15

Balas E, Simonetti N, Vazacopoulos A (2008) Job shop scheduling with setup times, deadlines and precedence constraints. Journal of Scheduling 11(4):253–262

Behnamian J (2014) Scheduling and worker assignment problems on hybrid flowshop with cost-related objective function. The International Journal of Advanced Manufacturing Technology 74(1-4):267–283

Bigras LP, Gamache M, Savard G (2008) The time-dependent traveling salesman problem and single machine scheduling problems with sequence dependent setup times. Discrete Optimization 5(4):685–699

Błażewicz J, Lenstra JK, Rinnooy Kan AHG (1983) Scheduling subject to resource constraints: classification and complexity. Discrete Applied Mathematics 5(1)

Błażewicz J, Ecker KH, Pesch E, Schmidt G, Węglarz J (2007) Handbook on Scheduling: from Theory to Applications. Springer, Berlin

Boost (2018) The Boost Graph Library (BGL). https://www.boost.org/doc/libs/1_67_0/libs/graph/doc/index.html, last accessed 2018-06-22

Brucker P, Schlie R (1990) Job-shop scheduling with multi-purpose machines. Computing 45(4):369–375

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

De Bruecker P, Van den Bergh J, Beliën J, Demeulemeester E (2015) Workforce planning incorporating skills: State of the art. European Journal of Operational Research 243(1):1–16

Defersha FM, Chen M (2010) A parallel genetic algorithm for a flexible job-shop scheduling problem with sequence dependent setups. The International Journal of Advanced Manufacturing Technology 49(1-4):263–279

Della Croce F, Grosso A, Salassa F (2014) A matheuristic approach for the two-machine total completion time flow shop problem. Annals of Operations Research 213(1):67–78

Giffler B, Thompson GL (1960) Algorithms for solving production-scheduling problems. Operations Research 8(4):487–503

Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG (1979) Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics 5:287–326

Günther HO, Lee TE (2007) Scheduling and control of automated manufacturing systems. OR Spectrum 29(3):373–374

Haupt R (1989) A survey of priority rule-based scheduling. OR Spektrum 11(1):3–16

Hooker JN, Ottosson G (2003) Logic-based Benders decomposition. Mathematical Programming 96(1):33–60

Hurink J, Jurisch B, Thole M (1994) Tabu search for the job-shop scheduling problem with multi-purpose machines. OR Spektrum 15(4):205–215

Lang M, Li H (2011) Research on dual-resource multi-objective flexible job shop scheduling under uncertainty. In: Proceedings of the 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce, IEEE, AIMSEC '11, pp 1375–1378

Lei D, Guo X (2014) Variable neighbourhood search for dual-resource constrained flexible job shop scheduling. International Journal of Production Research 52(9):2519–2529

Lei D, Tan X (2016) Local search with controlled deterioration for multi-objective scheduling in dual-resource constrained flexible job shop. In: Proceedings of the 28th Chinese Control and Decision Conference, IEEE, CCDC '16, pp 4921–4926

Lenstra JK, Rinnooy Kan AHG (1979) Computational complexity of discrete optimization problems. Annals of Discrete Mathematics 4:121–140

Lowerre BT (1976) The HARPY speech recognition system. PhD thesis, Carnegie-Mellon University, Pittsburgh, Pennsylvania

Mastrolilli M, Gambardella LM (2000) Effective neighbourhood functions for the flexible job shop problem. Journal of Scheduling 3(1):3–20

Mousakhani M (2013) Sequence-dependent setup time flexible job shop scheduling problem to minimise total tardiness. International Journal of Production Research 51(12):3476–3487

Nourali S, Imanipour N, Shahriari MR (2012) A mathematical model for integrated process planning and scheduling in flexible assembly job shop environment with sequence dependent setup times. International Journal of Mathematical Analysis 6(41-44):2117–2132

Özgüven C, Yavuz Y, Özbakır L (2012) Mixed integer goal programming models for the flexible job-shop scheduling problems with separable and non-separable sequence dependent setup times. Applied Mathematical Modelling 36(2):846–858

Paksi ABN, Ma'ruf A (2016) Flexible job-shop scheduling with dual-resource constraints to minimize tardiness using genetic algorithm. In: Proceedings of the 2nd International Manufacturing Engineering Conference and 3rd Asia-Pacific Conference on Manufacturing Systems, IOP Publishing, p 012060

Rossi A (2014) Flexible job shop scheduling with sequence-dependent setup and transportation times by ant colony with reinforced pheromone relationships. International Journal of Production Economics 153:253–267

Saidi-Mehrabad M, Fattahi P (2007) Flexible job shop scheduling with tabu search algorithms. The International Journal of Advanced Manufacturing Technology 32(5-6):563–570

Sels V, Gheysen N, Vanhoucke M (2012) A comparison of priority rules for the job shop scheduling problem under different flow time-and tardiness-related objective functions. International Journal of Production Research 50(15):4255–4270

Shen L, Dauzère-Pérès S, Neufeld JS (2017) Solving the flexible job shop scheduling problem with sequence-dependent setup times. European Journal of Operational Research 265(2):503–516

Tran TT, Beck JC (2012) Logic-based Benders decomposition for alternative resource scheduling with sequence dependent setups. In: Proceedings of the 20th European Conference on Artificial Intelligence, ACM, ECAI '12, pp 774–779

Treleven M (1989) A review of the dual resource constrained system research. IIE Transactions 21(3):279–287

Venditti L, Pacciarelli D, Meloni C (2010) A tabu search algorithm for scheduling pharmaceutical packaging operations. European Journal of Operational Research 202(2):538–546

Vilcot G, Billaut JC (2008) A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem. European Journal of Operational Research 190(2):398–411

Xu J, Xu X, Xie SQ (2011) Recent developments in dual resource constrained (DRC) system research. European Journal of Operational Research 215(2):309–318

Yazdani M, Zandieh M, Tavakkoli-Moghaddam R, Jolai F (2015) Two meta-heuristic algorithms for the dual-resource constrained flexible job-shop scheduling problem. Scientia Iranica - Transaction E 22(3):1242–1257

Zhang J, Liu GB (2012) Hybrid ant colony algorithm for job shop schedule with unrelated parallel machines. In: Proceedings of the Conference on Frontiers of Advanced Materials and Engineering Technology, Trans Tech Publications, pp 905–908

Zhang J, Wang W, Xu X (2015) A hybrid discrete particle swarm optimization for dual-resource constrained job shop scheduling with resource flexibility. Journal of Intelligent Manufacturing 28(8):1961–1972

Zheng XL, Wang L (2016) A knowledge-guided fruit fly optimization algorithm for dual resource constrained flexible job-shop scheduling problem. International Journal of Production Research 54(18):5554–5566