Journal of Scheduling (2019) 22:635–661
https://doi.org/10.1007/s10951-019-00607-9

Shop scheduling problems with pliable jobs

S. Knust[1] · N. V. Shakhlevich[2] · S. Waldherr[3] · C. Weiß[4]

Published online: 2 April 2019
© The Author(s) 2019

Abstract
In this paper, we study a new type of flow shop and open shop models, which handle so-called "pliable" jobs: their total processing times are given, but individual processing times of operations which make up these jobs are flexible and need to be determined. Our analysis demonstrates that many versions of flow shop and open shop problems with pliable jobs appear to be computationally easier than their traditional counterparts, unless the jobs have job-dependent restrictions imposed on minimum and maximum operation lengths. In the latter case, most problems with pliability become NP-hard even in the case of two machines.

Keywords Scheduling · Flow shop · Open shop · Identical parallel machines · Preemption

1 Introduction

In traditional flow shop and open shop models, n jobs of the set $\mathcal{J} = \{1, 2, \ldots, n\}$ are processed by m machines M_i, $1 \le i \le m$. Each job j, $1 \le j \le n$, consists of m operations O_{ij}, one operation on each machine M_i, with processing times p_{ij}, $1 \le i \le m$. A job cannot be processed by two machines at the same time and a machine cannot process two jobs simultaneously. In the flow shop model, all jobs have the same machine order, while in the open shop model the machine order is not fixed and can be chosen arbitrarily for each job. The goal is to select an order of operations on each machine and for open shops additionally the order of operations for each job, so that a given objective function f depending on job completion times is minimized.

Both models, flow shop and open shop, have a long history of study, see, e.g., Brucker (2007), Pinedo (2016). Over the last 60 years, the classical versions were extended to handle additional features of practical importance. The main extensions relevant for our study are processing with *preemption* and processing with *lot splitting* or *lot streaming*. In preemptive models, an operation can be cut into an arbitrary number of pieces which are then processed independently. In the models with lot splitting or lot streaming, operations can be divided into sublots, which can then be treated as new operations (cf. Kropp and Smunt 1990; Trietsch and Baker 1993; Chang and Chiu 2005). In the preemptive case, pieces of the same job processed by two machines cannot overlap in time, while in the case of lot splitting or lot streaming overlapping may happen if the pieces belong to different sublots.

Other concepts related to operation splitting and relocation, which have appeared more recently, deal with *flexible operations* and *operation redistribution* (cf. Gupta et al. 2004; Burdett and Kozan 2001, respectively). In the first model, jobs typically consist of more than m operations, some of which are fixed and have to be processed by dedicated machines while others are flexible and need to be assigned to one of the appropriate machines. In the second model, operation parts can be moved to neighboring machines if those machines are equipped to handle them.

✉ N. V. Shakhlevich
N.Shakhlevich@leeds.ac.uk

S. Knust
sknust@uni-osnabrueck.de

S. Waldherr
waldhers@in.tum.de

C. Weiß
christian.weiss@itwm.fraunhofer.de

[1] Institute of Computer Science, University of Osnabrück, 49069 Osnabrück, Germany

[2] School of Computing, University of Leeds, Leeds LS2 9JT, UK

[3] Department of Informatics, Technical University of Munich, 85748 Garching, Germany

[4] Department of Optimization, Fraunhofer Institute for Industrial Mathematics ITWM, 67663 Kaiserslautern, Germany

1.1 Pliability

In all models discussed above, processing times p_{ij} are given for the operations O_{ij} and these processing amounts have to be completed in full even if splitting happens and/or operation parts are moved to different machines. In this paper, we study a different way of splitting jobs, where operation lengths are not given in advance, but have to be determined. To distinguish our model from those studied previously, we introduce the notion of *pliability* (note that the term "splitting" is already reserved for other models, see, e.g., Serafini (1996), where a job can be split and then processed independently on different machines). Formally, a pliable job j is given by its total processing amount p_j, which has to be split among the m machines. Operation lengths $x_{ij} \in \mathbb{R}_{\geq 0}$ have to be determined as part of the decision making process. The combined length of all operations of job j over all machines has to match the given total processing requirement of job j:

$$\sum_{i=1}^{m} x_{ij} = p_j, \quad 1 \leq j \leq n.$$

We initiate this line of research by introducing three models with varying restrictiveness on the pliability of jobs.

(i) *Unrestricted pliability* means that we only have to fulfill

$$0 \leq x_{ij} \leq p_j, \quad 1 \leq i \leq m, \quad 1 \leq j \leq n.$$

(ii) *Restricted pliability with a common lower bound* means that the jobs need to be split complying with the restriction on a minimum length $\underline{p}$ of any operation:

$$\underline{p} \leq x_{ij} \leq p_j, \quad 1 \leq i \leq m, \quad 1 \leq j \leq n.$$

Note that for a feasible instance we must have $p_j \geq m\underline{p}$ for each job $1 \leq j \leq n$.

(iii) *Restricted pliability* means that the jobs need to be split complying with individual lower and upper bounds $\underline{p}_{ij}$, $\overline{p}_{ij}$ given for all operations:

$$\underline{p}_{ij} \leq x_{ij} \leq \overline{p}_{ij}, \quad 1 \leq i \leq m, \quad 1 \leq j \leq n.$$

Again, to assure feasibility we must have

$$\sum_{i=1}^{m} \underline{p}_{ij} \leq p_j \leq \sum_{i=1}^{m} \overline{p}_{ij} \quad \text{for all } j = 1, \ldots, n.$$

Model (i) assumes full flexibility of the machines and admits a low level of job granularity, accepting arbitrarily small operations. Although infinitesimally small operations are allowed, they are treated as zero-length operations rather than missing operations (for these concepts cf. Hefetz and Adiri 1982). They need to be allocated to the corresponding machine in a non-conflict way, and in the flow shop case the required machine order has to be respected.

Model (ii) is characterized by a limited level of job granularity defined by a common parameter $\underline{p}$ for all jobs, while the limitations in model (iii) can be different for distinct job-machine pairs. Clearly, model (i) is a special case of model (ii) with $\underline{p} = 0$, and model (ii) is a special case of model (iii) with $\underline{p}_{ij} = \underline{p}$, $\overline{p}_{ij} = p_j$. The classical flow shop and open shop problems are special cases of model (iii) with $\underline{p}_{ij} = \overline{p}_{ij} = p_{ij}$ for all $1 \leq i \leq m$, $1 \leq j \leq n$.

Extending the $\alpha|\beta|\gamma$-notation, we denote flow shop and open shop problems with unrestricted pliability of type (i) by $F|plbl|\gamma$ and $O|plbl|\gamma$. In the presence of additional restrictions of models (ii)–(iii), the given bounds are indicated in the second field as $plbl(\underline{p})$ and $plbl(\underline{p}_{ij}, \overline{p}_{ij})$, respectively.

The third field γ denotes the optimization criterion. It may include one of the traditional scheduling functions or simply "f" for an arbitrary non-decreasing objective function depending on completion times C_j of the jobs $j \in \mathcal{J}$. We distinguish between *minmax* and *minsum* objectives:

$$f_{\max} = \max_{j \in \mathcal{J}} \left\{ f_j(C_j) \right\}, \tag{1}$$

$$f_{\Sigma} = \sum_{j \in \mathcal{J}} f_j(C_j), \tag{2}$$

where $f_j(C_j)$ are non-decreasing functions. The two typical minmax examples are the makespan $C_{\max} = \max\{C_j | j \in \mathcal{J}\}$ and the maximum lateness $L_{\max} = \max\{C_j - d_j | j \in \mathcal{J}\}$, where d_j is the due date of job j. Minsum examples include the total completion time objective $\sum C_j$, total tardiness $\sum T_j$, where $T_j = \max\left\{C_j - d_j, 0\right\}$, or the total number of late jobs $\sum U_j$, where $U_j \in \{0, 1\}$, depending on whether a job is completed on time or after its due date d_j. If jobs $j \in \mathcal{J}$ have different weights w_j, then the latter functions can be extended to their weighted counterparts $\sum w_j C_j$, $\sum w_j T_j$, $\sum w_j U_j$.

1.2 Contributions

Our main results include a study of general properties of pliability models, formulating a general methodology for handling them and using it to perform a thorough complexity classification of the models. Note that all results are derived under the assumption that the jobs are available simultaneously, at zero release times.

The obtained results for flow shop and open shop problems with pliable jobs are summarized in Tables 1, 2 for the case where the number of jobs is not smaller than the number of machines, $n \geq m$, and in Table 3 for the case where there are fewer jobs than machines, $n < m$. For comparison, we also

Table 1 Open shop and flow shop problems with pliable jobs and minmax objectives, $n \geq m$

Problems	Traditional (non-pliability) models		Pliability models (this paper)					
	No pmtn	pmtn	Model (i): $plbl$		Model (ii): $plbl(\underline{p})$		Model (iii): $plbl(\underline{p}_{ij}, \overline{p}_{ij})$	
$O2\vert\circ\vert C_{\max}$	$O(n)$	$O(n)$	$O(n)$	Theorem 5	$O(n)$	Theorem 11	$O(n)$	Theorem11
$F2\vert\circ\vert C_{\max}$	$O(n\log n)$	$O(n\log n)$	$O(n)$	Theorem 5	$O(n)$	Theorem 8	NP-h‡	Theorem 10
$Om\vert\circ\vert C_{\max}$	NP-h‡ $(m \geq 3)$	$O(n^2)$	$O(n)$	Theorem 5	Open		NP-h‡ $(m \geq 3)$	Sect. 3.3
$Fm\vert\circ\vert C_{\max}$	sNP-h $(m \geq 3)$	sNP-h $(m \geq 3)$	$O(n)$	Theorem 5	$O(n)$	Theorem 8	sNP-h $(m \geq 3)$	Sect. 3.3
$O\vert\circ\vert C_{\max}$	sNP-h	$O(n^2m^2)$	$O(n)^*$	Theorem 5	Open		sNP-h	Sect. 3.3
$F\vert\circ\vert C_{\max}$	sNP-h $(m \geq 3)$	sNP-h	$O(n)^*$	Theorem 5	$O(n)^*$	Theorem 8	sNP-h $(m \geq 3)$	Sect. 3.3
$O\vert\circ\vert L_{\max}$	sNP-h $(m \geq 2)$	Linear programming	$O(n\log n)^*$	Theorem 6	Open		sNP-h $(m \geq 2)$	Sect. 3.3
$F\vert\circ\vert L_{\max}$	sNP-h $(m \geq 2)$	sNP-h $(m \geq 2)$	$O(n\log n + mn)$	Theorem 6	$O(n\log n + mn)$	Theorem 9	sNP-h $(m \geq 2)$	Sect. 3.3

sNP-h: strongly NP-hard
NP-h‡: NP-hard; open whether strongly NP-hard or pseudopolynomially solvable
*With compact encoding of the output; additional term $O(nm)$ for constructing complete schedule

Table 2 Open shop and flow shop problems with pliable jobs and minsum objectives, $n \geq m$

Problems	Traditional (non-pliability) models		Pliability models (this paper)					
	No pmtn	pmtn	Model (i): $plbl$		Model (ii): $plbl(\underline{p})$		Model (iii):$plbl(\underline{p}_{ij}, \overline{p}_{ij})$	
$O\vert\circ\vert\sum C_j$	sNP-h $(m \geq 2)$	NP-h‡ $(m \geq 2)$	$O(n\log n)^*$	Theorem 13	$O(n\log n)^*$	Theorem 15	sNP-h $(m \geq 2)$	Sect. 3.3
$F\vert\circ\vert\sum C_j$	sNP-h $(m \geq 2)$	sNP-h $(m \geq 2)$	$O(n\log n)^*$	Theorem 12	$O(n\log n)^*$	Theorem 14	sNP-h $(m \geq 2)$	Sect. 3.3
$Om\vert\circ\vert\sum w_jC_j$	sNP-h $(m \geq 2)$	sNP-h $(m \geq 2)$	NP-h†	Sect. 7.4	NP-h‡	Sect. 7.4	sNP-h $(m \geq 2)$	Sect. 3.3
$Fm\vert\circ\vert\sum w_jC_j$	sNP-h $(m \geq 2)$	sNP-h $(m \geq 2)$	NP-h†	Sect. 7.4	NP-h‡	Sect. 7.4	sNP-h $(m \geq 2)$	Sect. 3.3
$O\vert\circ\vert\sum w_jC_j$	sNP-h $(m \geq 2)$	sNP-h $(m \geq 2)$	sNP-h	Theorem 2	sNP-h	Proposition 1	sNP-h $(m \geq 2)$	Sect. 3.3
$F\vert\circ\vert\sum w_jC_j$	sNP-h $(m \geq 2)$	sNP-h $(m \geq 2)$	sNP-h	Theorem 2	sNP-h	Proposition 1	sNP-h $(m \geq 2)$	Sect. 3.3
$Om\vert\circ\vert\sum U_j$	sNP-h $(m \geq 2)$	NP-h $(m \geq 2)$	$O\left(n^{3(m-1)}\right)$	Sect. 7.4	open		sNP-h $(m \geq 2)$	Sect. 3.3
$Fm\vert\circ\vert\sum U_j$	sNP-h $(m \geq 2)$	sNP-h $(m \geq 2)$	$O\left(n^{3(m-1)}\right)$	Sect. 7.4	$O\left(n^{3(m-1)}\right)$	Sect. 7.4	sNP-h $(m \geq 2)$	Sect. 3.3
$O\vert\circ\vert\sum U_j$	sNP-h $(m \geq 2)$	sNP-h $(m \geq 2)$	NP-h‡	Theorem 2	NP-h‡	Proposition 1	sNP-h	Sect. 3.3
$F\vert\circ\vert\sum U_j$	sNP-h $(m \geq 2)$	sNP-h $(m \geq 2)$	NP-h‡	Theorem 2	NP-h‡	Proposition 1	sNP-h	Sect. 3.3

sNP-h: strongly NP-hard
NP-h†: NP-hard; pseudopolynomially solvable
NP-h‡: NP-hard; open whether strongly NP-hard or pseudopolynomially solvable
*With compact encoding of the output; additional term $O(nm)$ for constructing complete schedule

Table 3 Open shop and flow shop problems with pliable jobs, $n < m$ (results from Sect. 8)

Problems	Complexity
$O\|plbl\|f$, $O\|plbl(\underline{p})\|f$ for any regular objective f	$O(1)^*$
$F\|plbl\|C_{\max}$	$O(n)^*$
$F\|plbl\|f$ for any regular objective f	$O(n \log n)^*$
$F\|plbl(\underline{p})\|C_{\max}$	$O(n)^*$
$F\|plbl(\underline{p})\|L_{\max}$	$O(n \log n)^*$
$F\|plbl(\underline{p})\|\sum C_j$	$O(n \log n)^*$
$F\|plbl(\underline{p})\|\sum w_j C_j$	$O(n^2)^*$
$F\|plbl(\underline{p})\|\sum U_j$	$O(n \log n)^*$
$F\|plbl(\underline{p})\|\sum T_j$	$O(n^4 \sum p_j)^*$

*With compact encoding of the output; additional term $O(nm)$ for constructing complete schedule

provide related results for traditional (non-pliability) models; for references see, e.g., Brucker (2007), Pinedo (2016).

Note that the input for problems with pliability of type (i) and (ii) consists of $O(n)$ entries, while the output consists of $O(nm)$ entries if it is needed to produce a full characterization of a schedule, specifying starting/completion times of all operations. Whenever an optimal schedule has a special structure and it is possible to derive formulae for computing starting/completion times of individual operations, we list reduced time complexities in Tables 1, 2, 3, associated with finding characteristics of a single operation and for required auxiliary preprocessing. An additional term $O(nm)$ has to be added if an optimal solution should be specified in full (see the entries in the table marked by an asterisk).

The remainder of the paper is organized as follows. In Sect. 2 we provide an overview of related models studied in the literature. General properties of flow shop and open shop models with pliability are discussed in Sect. 3. Our main focus is on the case where the number of jobs is not smaller than the number of machines, $n \geq m$: pliability models with minmax objectives are discussed in Sects. 4, 5, and 6; results for models with minsum objectives are presented in Sect. 7. The situation $n < m$ is discussed in Sect. 8. Concluding remarks are summarized in Sect. 9.

2 Related work

The study of shop scheduling models with pliability is motivated by scenarios where jobs are processed by machines of different types in a flow shop or open shop manner, and transition from one machine to another requires intermediate actions, which can be performed by either of two consecutive machines. Those actions may be related to quality control, preprocessing, postprocessing, or setup operations. Alternatively, operators specializing on serving particular machines may be able to perform not only the main operations they are trained for, but also additional operations on adjacent machines, thus reducing possible delays and idle times in the system. In manufacturing applications, not only the operators can be flexible, but machines as well if they are designed to perform operations of different types. Various examples of flexible machines, such as CNC machines and machines producing printed circuit boards, are reviewed by Crama and Gultekin (2010), and examples of flexible operations are presented in the context of assembly line scheduling, see, e.g., Ostolaza et al. (1990), McLain et al. (1992), Anuar and Bukchin (2006) and Askin and Chen (2006). Note that the nature of processing in assembly lines and in flow shops and open shops is quite different.

For the shop models we consider, the most relevant results are known in the area of flow shop scheduling, for (a) models with *flexible operations* and (b) models with *operation redistribution*. In the models of type (a), there are fixed operations processed by dedicated machines and flexible ones which can be processed (without preemption) by one of the two adjacent machines. In the models of type (b), any flexible operation can be preempted and parts of it can be relocated to an adjacent machine.

Comparing the pliability model with models (a) and (b), we summarize below the main common points and the points of difference.

- In both models (a) and (b), it is usually assumed that machines operate in the flow shop manner. In comparison, the nature of the pliability model is rather general: it is relevant for any shop model, including flow shop and open shop, the two models considered in this paper. It can be also generalized for the job shop model, although this is beyond the scope of our paper.
- All models, including the pliability one, deal with flexible allocation of operations or their parts. The level of flexibility is slightly different in the three models. In model (a), each flexible operation has to be allocated to one machine in full. In model (b), flexible operations can be split at any point of time (see Burdett and Kozan 2001). Still there is a limitation on the machine choice for the allocation: only an adjacent machine in the flow shop chain of machines can be selected. Additionally, for some operations on a given machine, it may only be allowed to share them with one of the two neighboring machines (either the previous or the next machine). In the pliability model, every machine must get at least the minimum workload associated with job j (namely, 0, $\underline{p}$ or $\underline{p}_{ij}$, depending on the model type), with full freedom for the distribution of the remaining workload.

- In models (a) and (b), processing times p_{ij} are given for all operations; for the pliability model the total job lengths p_j are given and operations' lower and upper bounds.

The difference between the three models discussed above becomes more apparent if the number of machines is $m \geq 3$. The models with $m = 2$ machines are closely related, especially if the flexible operation can be started on the first machine and then preempted at any point in time to be restarted on the second machine. Indeed, in such an instance of model (a) with preemption of the flexible operation, every job j consists of three operations of lengths p_{1j}, p_{*j} and p_{2j}. The first and the last operations are fixed and have to be processed by machines M_1 and M_2, while the middle one can be split into at most two parts and distributed between M_1 and M_2. In an equivalent instance of the pliability model, job j is defined by its total processing time $p_j = p_{1j} + p_{*j} + p_{2j}$ and lower and upper bounds on operation lengths, $\underline{p}_{1j} = p_{1j}$, $\underline{p}_{2j} = p_{2j}$ and $\overline{p}_{1j} = \overline{p}_{2j} = p_j$. On the other hand, the flow shop model of type (b) with relocatable operations of lengths p_{1j}, p_{2j} associated with machines M_1 and M_2 has similarities with the pliability model of type (i) with $\underline{p}_{1j} = \underline{p}_{2j} = 0$ and $\overline{p}_{1j} = \overline{p}_{2j} = p_{1j} + p_{2j}$. Interestingly, the preemptive version of model (a) has not been considered in the literature, although it is observed by Lin et al. (2016) that this case would be an interesting extension of the model.

The traditional, unsplittable flow shop problem (a) with flexible operations is NP-hard for the makespan objective even in its simplest setting with $m = 2$ machines, see Gupta et al. (2004). It remains NP-hard even if the job sequence is fixed (cf. Lin et al. 2016). Therefore, the study of models with flexible operations focuses on approximability results (Gupta et al. 2004), pseudopolynomial-time algorithms (Lin et al. 2016), construction heuristics and local search methods (Ruiz-Torres et al. 2010, 2011). The models often incorporate special features such as limitations on the buffer capacities used for handling jobs in-between the machines, requirements to optimize workstation utilization or throughput rate, etc. The main special case of the flexible model, for which efficient algorithms have been developed, is the one with identical jobs, see Crama and Gultekin (2010), Gultekin (2012).

Flow shop problems with redistribution are less studied (compared to the model with flexible operations), especially in terms of a complexity analysis. Burdett and Kozan (2001) consider several scenarios where adjacent machines can perform the same tasks and parts of an operation may be shifted to the upstream or downstream machine. Besides proposing a MILP formulation, heuristics are described and empirically evaluated. In Bultmann et al. (2018a), a very general framework for flexibility is introduced. Similar to the model with pliable jobs, the processing times of the operations are not fixed in advance, but lower and upper bounds on the processing times are specified for consecutive machines. A decomposition algorithm is proposed, using a local search procedure on the set of all permutations where optimal corresponding processing times are efficiently computed in a second step. In Bultmann et al. (2018b), a similar approach can be found for a synchronous flow shop environment with pliable jobs.

Our study continues the line of research on flow shop models with flexibility and relocation, and extends it also to open shop counterparts.

3 General properties and reductions

In this section, we explore the links between the pliability models and classical scheduling models: flow shop, open shop and single-stage scheduling with parallel machines. Furthermore, we establish some key properties of the pliability models and discuss their implications.

3.1 Unrestricted pliability

In order to address type (i) problems $O|plbl|f$ and $F|plbl|f$, it is often useful to relax the requirement of dedicated machines typical for open shops and flow shops and to consider identical parallel machines instead. The pliability condition, that allows to determine the actual processing times of the operations, can then be interpreted as processing with preemption. The resulting problem is denoted by $P|pmtn|f$, where P denotes "identical parallel machines" and *pmtn* denotes preemption. In this problem, jobs may be split into multiple parts and these job parts can be processed on different machines.

Clearly, if in a feasible schedule for problem $P|pmtn|f$ every machine processes exactly n job parts, one for each job, then that schedule also represents a feasible open shop schedule. Alternatively, if every machine processes each job at most once, then the schedule can be converted into a feasible open shop schedule by introducing zero-length operations for missing operations at the beginning of a schedule. We will call a schedule for problem $P|pmtn|f$ with exactly nm job parts, some of which may be of zero length, an "open shop type" schedule, or an *O-type* schedule for short.

In a "flow shop type" schedule, or an *F-type* schedule for short, each machine processes exactly one part of each job, as in an O-type schedule; additionally jobs visit the machines in a flow shop manner, moving from machine M_i to M_{i+1}, $1 \leq i \leq m - 1$. As for an O-type schedule, some of the nm operations may be of zero length. However, in F-type schedules, those zero-length operations might appear in the middle of the schedule and zero-length operations which appear on a machine M_i with $2 \leq i \leq m$ cannot usually be moved to the beginning of the schedule instead. Therefore, as opposed to the open shop case, for F-type schedules such zero-length operations may create idle times on upstream or downstream machines and have an impact on the completion time of the job they belong to.

In the case of *permutation schedules*, with all machines processing the jobs in the same order, the notion of an F-type schedule coincides with the notion of a "*Permutation Flow Shop-like schedule*", introduced by Prot et al. (2013).

For a scheduling problem $\alpha|\beta|\gamma$, let $\mathcal{S}(\alpha|\beta|\gamma)$ denote the set of its feasible solutions. Since any feasible solution to $F|plbl|f$ is also feasible for $O|plbl|f$, and in its turn any feasible solution to $O|plbl|f$ is feasible for $P|pmtn|f$, we conclude:

$$\mathcal{S}(F|plbl|f) \subseteq \mathcal{S}(O|plbl|f) \subseteq \mathcal{S}(P|pmtn|f). \tag{3}$$

In what follows we revise known algorithms and NP-hardness results for problem $P|pmtn|f$ with the focus on optimal schedules of O- and F-type. The existence of an optimal F-type schedule for problem $P|pmtn|f$ with any non-decreasing objective function f was proved by Prot et al. (2013).

Theorem 1 (Prot et al. 2013) *For problem $P|pmtn|f$ with any non-decreasing objective function $f \in \{f_{\max}, f_{\Sigma}\}$, there exists an optimal F-type schedule.*

Clearly, due to the inclosed structure of solution regions (3), an optimal schedule for problem $P|pmtn|f$, which is of F-type, is also an optimal schedule for problems $F|plbl|f$ and $O|plbl|f$ with pliable jobs. It follows that for problems $P|pmtn|f$, $O|plbl|f$ and $F|plbl|f$ there exists a common optimal schedule and it is of F-type. Thus the optimal objective values for these three problems are the same, and the following corollary holds.

Corollary 1 *Any optimal schedule for problem $P|pmtn|f$ which is of F-type is also optimal for problems $F|plbl|f$ and $O|plbl|f$. Any optimal schedule for problem $F|plbl|f$ or $O|plbl|f$ is also optimal for problem $P|pmtn|f$.*

We use Corollary 1 in order to transfer complexity results from scheduling problems with parallel machines to shop scheduling with pliability.

Consider first the case when a particular version of problem $P|pmtn|f$ is NP-hard. Then, the corresponding versions of $F|plbl|f$ and $O|plbl|f$ are also NP-hard since otherwise, due to the second part of Corollary 1, a polynomial-time algorithm for $F|plbl|f$ or $O|plbl|f$ would also solve problem $P|pmtn|f$ in polynomial time. We combine this observation with the known NP-hardness results for $P|pmtn|f$ (see Brucker 2007; Lawler et al. 1993).

Theorem 2 *Problems $F2|plbl|f$ and $O2|plbl|f$ with $f \in \{\sum w_jC_j, \sum T_j, \sum w_jU_j\}$ are NP-hard in the ordinary sense, and they are NP-hard in the strong sense if $f = \sum w_jT_j$.*

Problems $F|plbl|f$ and $O|plbl|f$ with $f = \sum U_j$ are NP-hard in the ordinary sense and they are NP-hard in the strong sense if $f = \sum w_jC_j$.

Whenever a job sequence is known for an optimal F-type schedule for problem $P|pmtn|f$, an optimal allocation of jobs to the machines can be obtained as a solution to the following model, see Prot et al. (2013):

$$\begin{array}{ll}
\min \; f(C_1, C_2, \ldots, C_n) & \\
\text{s.t.} & \\
\sum_{i=1}^{m} x_{ij} = p_j, & 1 \le j \le n, \\
t_{ij} + x_{ij} \le t_{i+1,j}, & 1 \le i \le m-1, \; 1 \le j \le n, \\
t_{ij} + x_{ij} \le t_{i,j+1}, & 1 \le i \le m, \; 1 \le j \le n-1, \\
t_{mj} + x_{mj} = C_j, & 1 \le j \le n, \\
x_{ij}, t_{ij} \ge 0, & 1 \le i \le m, \;\; 1 \le j \le n, \\
C_j \ge 0, & 1 \le j \le n.
\end{array} \tag{4}$$

Here, it is assumed that the jobs are numbered in the order they appear in an optimal schedule, x_{ij} is the processing time of operation O_{ij}, t_{ij} is its starting time, and C_j is the completion time of job j. The range of functions that allow fixing the job sequence includes $f = C_{\max}$ (the job order can be arbitrary), $f = L_{\max}$ (earliest due date first; see Sahni 1979) and $f = \sum C_j$ (shortest processing time first; see Conway et al. 1967). Thus, in the case of $f \in \left\{C_{\max}, L_{\max}, \sum C_j\right\}$ problems $O|plbl|f$ and $F|plbl|f$ are solvable via linear programming. However, special properties of these pliability problems allow us to develop faster algorithms. We present them in Sects. 4.1, 4.2 and 7.1.

3.2 Restricted pliability with a common lower bound

In this section, we establish two important properties for the type (ii) flow shop model $F|plbl(\underline{p})|f$ with a common lower bound $\underline{p}$ on operation lengths. Unfortunately these properties cannot be easily generalized to the open shop version of the problem, $O|plbl(\underline{p})|f$. They also do not hold for the most general model of type (iii), where job splitting has to respect individual lower and upper bounds on operation lengths.

In Sect. 3.2.1, we prove that for problem $F|plbl(\underline{p})|f$ with an arbitrary (not necessarily non-decreasing) objective function f, there exists an optimal permutation schedule. Note that in Prot et al. (2013) Theorem 1 was proved for a more restricted type (i) problem ($\underline{p} = 0$) with a non-decreasing objective function. Their technique cannot be reused for our proof, as it involves cutting jobs into arbitrarily small pieces.

In Sect. 3.2.2, we present the common methodology for solving problem $F|plbl(\underline{p})|f$ with $f \in \{C_{\max}, L_{\max}, \sum C_j\}$

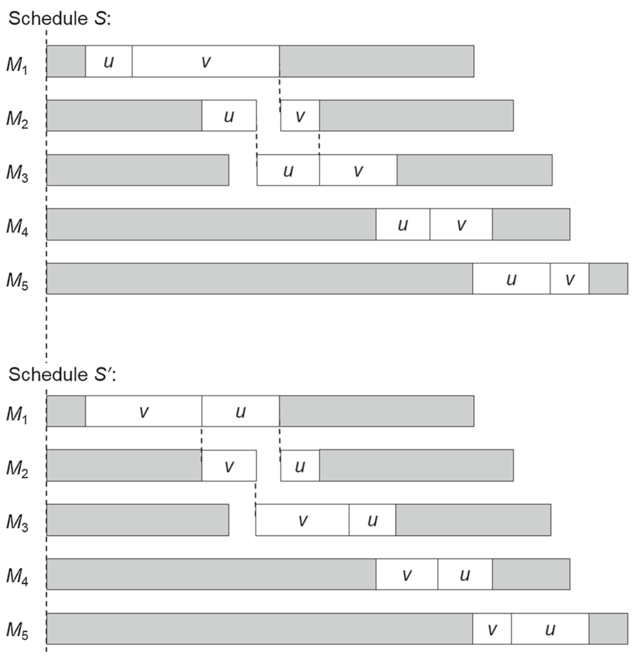

Fig. 1 Adjacent jobs swap: initial schedule S (above) and modified schedule S' (below); gray boxes represent fixed parts of schedules S and S' where the jobs $\mathcal{J}\backslash\{u, v\}$ are processed

based on job disaggregation and decomposing the problem into two subproblems. The implementation details of that methodology are then elaborated in Sects. 5.1, 5.2 and 7.2.

3.2.1 The existence of an optimal permutation schedule for $F|plbl(\underline{p})|f$

We start with an auxiliary property based on *adjacent swaps*. By that property, the order of any two adjacent jobs u and v in a permutation schedule can be reversed without making changes to the rest of the schedule. To achieve this, the operation lengths of jobs u and v may be redistributed, if necessary.

Lemma 1 *Given a feasible permutation schedule S for problem $F|plbl(\underline{p})|f$ where job u is sequenced immediately before job v in the permutation, there exists another feasible permutation schedule S' where u is sequenced immediately after v, while all remaining jobs are scheduled in the same time slots as in S. For schedule S',*

$$C'_j = C_j,\ j \in \mathcal{J}\backslash\{u, v\},$$
$$C'_u \le C_v,$$
$$C'_v \le C_v.$$

Lemma 1 is illustrated in Fig. 1 and proved in Appendix 1. Note that the proof uses the property that there is a common lower bound $\underline{p}_{ij} = \underline{p}$ for the operation lengths and that the remaining processing time $p_j - m\underline{p}$ of a job j can be distributed among the operations without restrictions. This does not work for problem $F|plbl(\underline{p}_{ij}, \overline{p}_{ij})|f$.

Theorem 3 *Given an arbitrary schedule S for problem $F|plbl(\underline{p})|f$, there exists a permutation schedule S' in which every job has the same completion time on machine M_m as in S.*

Proof The proof is done by induction on the number of machines m. For $m = 1$ the statement is obvious. Consider $m \ge 2$, assuming that the statement of the theorem holds for $m - 1$ machines.

Let S be a non-permutation schedule on m machines. We split S into two subschedules $S(M_1, \ldots, M_{m-1})$ and $S(M_m)$ defined over the corresponding machine sets. For the instance of the problem defined by $S(M_1, \ldots, M_{m-1})$, the induction hypothesis holds: there exists a permutation schedule $S'(M_1, \ldots, M_{m-1})$ such that each job has the same completion time on machine M_{m-1} as in $S(M_1, \ldots, M_{m-1})$.

Let $\hat{S}$ be a complete schedule defined as a combination of $S'(M_1, \ldots, M_{m-1})$ with the final part $S(M_m)$ of the original schedule S. Schedule $\hat{S}$ is feasible, since by the induction hypothesis each job completes on machine M_{m-1} at the same time in both schedules, $S(M_1, \ldots, M_{m-1})$ and $S'(M_1, \ldots, M_{m-1})$.

If $\hat{S}$ is a permutation schedule, then no further action is needed. Otherwise consider $S'(M_1, \ldots, M_{m-1})$ and apply a sequence of adjacent swaps, described in the proof of Lemma 1. The swaps eventually result in the same job order as in $S(M_m)$. We demonstrate that each swap on the first $m - 1$ machines does not cause conflicts with operations in $S(M_m)$.

Assume u is sequenced immediately before v in $S'(M_1, \ldots, M_{m-1})$, but somewhere after v in $S(M_m)$. Then, by Lemma 1, after swapping u and v on the first $m - 1$ machines, the completion time of job v on machine M_{m-1} is at most as large as before, and hence job v is not postponed on M_m. By the same lemma, the completion time of job u on M_{m-1} remains no larger than the completion time of job v before the swap. This means that u finishes on M_{m-1} before v starts on M_m, and therefore before u starts on M_m.

Performing at most $O(n^2)$ swaps in the part $S'(M_1, \ldots, M_{m-1})$, we get a permutation schedule on m machines without changing the completion times on machine M_m. □

It is worth noting that in the proof of Theorem 3, the schedule transformations keep the operations on the last machine unchanged. This implies that an optimal permutation schedule exists for any objective function depending on job completion times, monotone or non-monotone. Note also that Theorem 3 does not hold for the more general problem $F|plbl(\underline{p}_{ij}, \overline{p}_{ij})|f$ since for the special case $F||C_{\max}$ with more than three machines there exist instances for which only non-permutation schedules are optimal (see, e.g., Potts et al. 1991).

3.2.2 A job disaggregation approach for problem $F|plbl(\underline{p})|f$

In this section, we introduce the disaggregation approach, which serves as a common methodology for solving problem $F|plbl(\underline{p})|f$ with $f \in \{C_{\max}, L_{\max}, \sum C_j\}$. It provides the tool for constructing optimal schedules and for justifying their optimality. Problem-specific details on how the methodology can be implemented are presented in Sects. 5.1, 5.2 and 7.2.

The main idea is to define for an instance I of problem $F|plbl(\underline{p})|f$ two auxiliary instances by disaggregating the jobs into two parts: instance I^e of type (ii) with *equal* processing times and instance I^d of type (i) with *diminished* processing times. Optimal solutions to the two instances are then found and combined, delivering a solution to the initial problem.

Definition 1 For an instance I of problem $F|plbl(\underline{p})|f$, there are two associated instances:

> Instance I^e of type (ii) with processing times $p_j^e = m\underline{p}$ for all jobs $j \in J$ and with the same lower bound $\underline{p}$ as in the original instance I,
> Instance I^d of type (i) with processing times $p_j^d = p_j - m\underline{p}$, for all jobs $j \in J$, and zero lower bounds.

Let S^d and S^e be two feasible schedules for instances I^d and I^e which satisfy the following conditions, see Fig. 2 for an illustration.

(D1) S^d and S^e are permutation schedules with the same job sequence $(1, 2, \dots, n)$.
(D2) S^e has a staircase structure, uniquely defined by completion times $C_{ij}^e = (i + j - 1)\,\underline{p}$ of its operations O_{ij}. In that schedule, machine M_i is idle in the time interval $\left[0, (i-1)\,\underline{p}\right]$.
(D3) In S^d, every machine operates without idle times from time 0 until all assigned operations are completed; some operations in S^d may be of zero length.

Here, and in the remainder of the paper, we use the notion of an *idle time of machine* M_i if it occurs before the last job is completed on M_i.

Denote operation lengths in schedules S^d and S^e by p_{ij}^d and p_{ij}^e, respectively, where $p_{ij}^e = \underline{p}$. Schedules S^d and S^e satisfying properties (D1)-(D3) can be easily combined to produce a permutation schedule S for the original instance I, as illustrated in Fig. 2. The job order remains the same as in schedules S^d and S^e, while the aggregate operation lengths p_{ij} are defined as

$$p_{ij} = p_{ij}^d + p_{ij}^e.$$

Theorem 4 *Let S^d and S^e be feasible schedules for instances I^d and I^e satisfying conditions (D1)-(D3). Then, there exists an aggregate schedule S for the original instance I with*

$$C_{ij} = C_{ij}^d + (i + j - 1)\,\underline{p}, \tag{5}$$

where C_{ij}^d and $C_{ij}^e = (i + j - 1)\,\underline{p}$ are completion times of operations O_{ij} in S^d and S^e.

Conversely, if in a permutation schedule S for instance I with the job order $(1, 2, \dots, n)$ there are no idle times except for time intervals $\left[0, (i-1)\,\underline{p}\right]$ (as in Condition (D2) for schedule S^e), then S can be decomposed into two schedules S^e and S^d such that conditions (D1)-(D3) and relation (5) holds.

Proof Consider schedules S^d, S^e and their disjunctive graph representation shown in Fig. 3. In that graph, nodes (i, j) correspond to operations O_{ij}, $1 \le i \le m$, $1 \le j \le n$. The nodes are associated with weights: p_{ij}^d for the graph representing S^d and $p_{ij}^e = \underline{p}$ for the graph representing S^e. The length of a path in the graph is defined as the sum of weights of the nodes on the path. The completion time of any operation O_{ij} is calculated as the length of a longest path from the source node $(1, 1)$ to node (i, j). Combining S^d and S^e implies increasing the weights of all nodes in the graph for S^d by the same amount $\underline{p}$. Since any path from $(1, 1)$ to (i, j) includes exactly $i + j - 1$ nodes, the structure of a longest path does not change, and its length increases by $(i + j - 1)\underline{p}$, so that (5) holds for the aggregate schedule.

Similar arguments justify the reverse statement on decomposing S into S^e and S^d. $\square$

3.3 Restricted pliability with individual lower and upper bounds

The following proposition establishes basic reductions for type (iii) problems.

Proposition 1 *For flow shop and open shop problems, the following reductions hold:*

$$\alpha||f \propto \alpha|plbl(\underline{p}_{ij}, \overline{p}_{ij})|f, \tag{6}$$

$$\alpha|plbl|f \propto \alpha|plbl(\underline{p})|f \propto \alpha|plbl(\underline{p}_{ij}, \overline{p}_{ij})|f, \tag{7}$$

where $\alpha \in \{F, O\}$.

Here $A \propto B$ indicates that problem A polynomially reduces to problem B, see Garey et al. (1976).

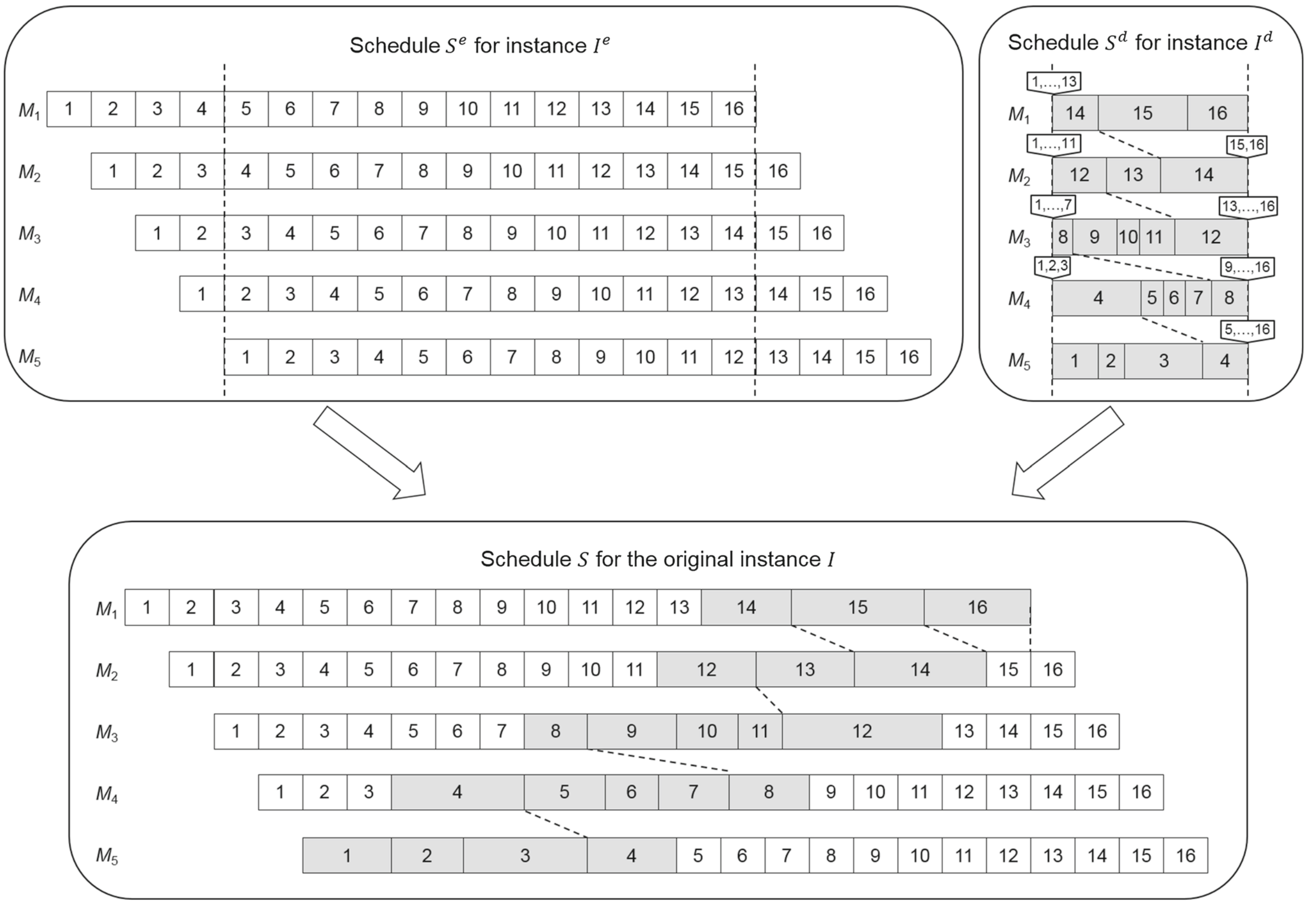
Schedule S^e for instance I^e
Schedule S^d for instance I^d
Schedule S for the original instance I
M_1
M_2
M_3
M_4
M_5

Fig. 2 Schedules S^e and S^d for instances I^e and I^d, and an aggregate schedule S

$$\begin{tikzcd}
M_1: & (1,1) \arrow[r] \arrow[d] & (1,2) \arrow[r] \arrow[d] & \cdots \arrow[r] & (1,n) \arrow[d] \\
M_2: & (2,1) \arrow[r] \arrow[d] & (2,2) \arrow[r] \arrow[d] & \cdots \arrow[r] & (2,n) \arrow[d] \\
 & \cdots \arrow[d] & \cdots \arrow[d] & \cdots & \cdots \arrow[d] \\
M_m: & (m,1) \arrow[r] & (m,2) \arrow[r] & \cdots \arrow[r] & (m,n)
\end{tikzcd}$$

Fig. 3 The disjunctive graph representation of schedules S^d and S^e

Reduction (6) follows from the fact that the pliability problem $\alpha|plbl(\underline{p}_{ij}, \overline{p}_{ij})|f$ with $\underline{p}_{ij} = \overline{p}_{ij}$, $1 \le i \le m$, $1 \le j \le n$, coincides with the traditional flow shop problem (if $\alpha = F$) or open shop problem (if $\alpha = O$). The chain of reductions (7) reflects the fact that pliability model (i) is a special case of model (ii), which in its turn is a special case of model (iii).

Using (6), we can transfer all NP-hardness results known for $F||f$ and $O||f$ to the corresponding pliability problems of type (iii), concluding that problem $O3|plbl(\underline{p}_{ij}, \overline{p}_{ij})|C_{\max}$ is NP-hard in the ordinary sense, while problems $F3|plbl(\underline{p}_{ij}, \overline{p}_{ij})|C_{\max}$, $O|plbl(\underline{p}_{ij}, \overline{p}_{ij})|C_{\max}$ and $\alpha 2|plbl(\underline{p}_{ij}, \overline{p}_{ij})|f$ with $\alpha \in \{F, O\}$ and $f \in \left\{L_{\max}, \sum C_j\right\}$ are NP-hard in the strong sense.

Similarly, using (7), we transfer the NP-hardness results for $f \in \{\sum w_j C_j, \sum T_j, \sum w_j T_j, \sum U_j, \sum w_j U_j\}$, discussed in Sect. 3.1 in relation to type (i) problems $\alpha|plbl|f$ to the pliability problems of types (ii) and (iii). Note that for the problems of type (iii) these results are dominated by those obtained through reduction (6).

4 Unrestricted pliability: Minmax objectives

In this section, we apply the methodology from Sect. 3.1 to develop efficient algorithms for problems $F|plbl|f_{\max}$ and $O|plbl|f_{\max}$ with unrestricted pliability. To this end, we consider the relaxed problem $P|pmtn|f$ and construct optimal F- and O-type schedules for it.

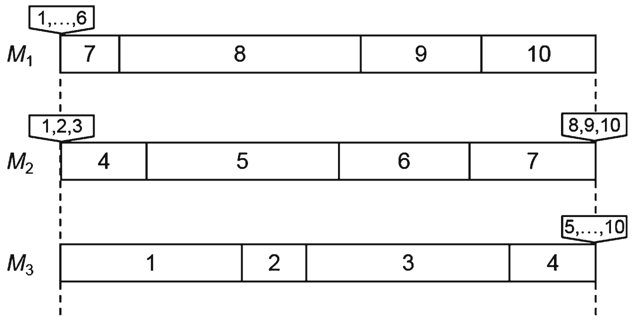

Fig. 4 An optimal schedule for $P|pmtn|C_{\max}$ of F-type and O-type, with zero-length operations

4.1 Problems $F|plbl|C_{\max}$ and $O|plbl|C_{\max}$

Consider problem $P|pmtn|C_{\max}$. An optimal schedule can be constructed in $O(n)$ time by the wrap-around algorithm (McNaughton 1959), achieving the optimum makespan value

$$C_* = \max\left\{\frac{1}{m}p(\mathcal{J}), \max_{j\in\mathcal{J}}\{p_j\}\right\}, \tag{8}$$

where $p(\mathcal{J}) = \sum_{j\in\mathcal{J}} p_j$. In order to force McNaughton's wrap-around algorithm to produce a solution of F- and O-type, suitable for problems $F|plbl|C_{\max}$ and $O|plbl|C_{\max}$, we consider the jobs in the order of their numbering and allocate them in the time window $[0, C_*]$ first on machine M_m, then on M_{m-1}, etc., until all jobs are fully allocated. Notice that machine M_m is always fully occupied in the interval $[0, C_*]$, while other machines might only be partly occupied in that interval, if $C_* = p_q$ for a job $q \in \mathcal{J}$ with the longest processing time. Note that after performing the wrap-around algorithm, there exist at most $m-1$ jobs which have operations of length greater than zero on more than one machine while all other jobs are processed on a single machine for their whole processing time. The order in which the machines are considered, gives an easy way for introducing zero-length operations, as illustrated in Fig. 4. The resulting schedule satisfies the requirements of F- and O-type schedules, has the minimum makespan C_*, and is therefore optimal for both problems, $F|plbl|C_{\max}$ and $O|plbl|C_{\max}$.

Theorem 5 *Problems $F|plbl|C_{\max}$ and $O|plbl|C_{\max}$ are solvable in $O(n)$ time.*

4.2 Problems $F|plbl|L_{\max}$ and $O|plbl|L_{\max}$

Consider the open shop problem $O|plbl|L_{\max}$ and its relaxed counterpart $P|pmtn|L_{\max}$. Our approach to find an optimal O-type schedule for the latter problem consists of two stages. First calculate an optimal value L_* of the objective using, for example, the closed form expression for L_* from Baptiste (2000). That calculation requires $O(n\log n)$ time due to the sorting of the jobs. In the second stage, adjust the due dates to $\overline{d}_j = d_j + L_*$, treat them as deadlines, and find a feasible schedule for $P|pmtn, C_j \le \overline{d}_j|-$. The fastest algorithm is due to Sahni (1979); its time complexity is $O(n\log(nm))$ or $O(n\log n)$ under our assumption $n \ge m$. It is a property of Sahni's algorithm that the resulting parallel machine schedule has at most one preemption per job, and a preempted job is not restarted on the same machine. Therefore, the schedule is of O-type, if zero-length operations are added at the beginning of the schedule.

Theorem 6 *Problem $O|plbl|L_{\max}$ is solvable in $O(n\log n)$ time.*

Consider now the flow shop problem $F|plbl|L_{\max}$, using again its relaxed counterpart $P|pmtn|L_{\max}$. The approach discussed above for constructing an O-type schedule is no longer applicable, since Sahni's algorithm used at the second stage does not guarantee that the resulting schedule is of F-type. An alternative approach for solving the second stage problem is to apply the $O(n\log n + mn)$-time algorithm by Baptiste (2000), which does find an optimal schedule of F-type, thus providing a solution to problem $F|plbl|L_{\max}$.

Theorem 7 *Problem $F|plbl|L_{\max}$ is solvable in $O(n\log n + mn)$ time.*

Interestingly, the term mn in the complexity estimate cannot be reduced, since there are instances which require $\Omega(nm)$ nonzero operations in an optimal schedule. One such instance is presented in Appendix 2. Recall that for problem $F|plbl|C_{\max}$ with the makespan objective, there exists an optimal schedule with the total number of nonzero operations bounded by $n+m-1$, see Sect. 4.1.

5 Restricted pliability with a common lower bound: Minmax objectives

In this section, we apply the methodology of Sect. 3.2 to problems $F|plbl(\underline{p})|f_{\max}$ by solving the flow shop problems $F|plbl|f_{\max}$. Furthermore, we discuss difficulties encountered for problem $O|plbl(\underline{p})|C_{\max}$.

5.1 Problem $F|plbl(\underline{p})|C_{\max}$

By Theorem 3 we limit our consideration to the class of permutation schedules and use the disaggregation technique from Sect. 3.2 to construct an optimal schedule and to justify its optimality. Given an instance I of problem $F|plbl|C_{\max}$, introduce instances I^d and I^e as in Definition 1.

Let S^d be a permutation schedule for instance I^d with every machine working contiguously from time 0 [in accordance with (D3) from Sect. 3.2.2], and let S^e be a solution to I^e in the staircase form, which uses the same job permutation as S^d [in accordance with (D1) and (D2) from Sect. 3.2.2]. Let S be the schedule for the original instance I obtained by combining S^d and S^e. By Theorem 4,

$$\begin{aligned} C_{\max}(S) = C_{mn}(S) &= C_{mn}(S^d) + (m+n-1)\,\underline{p} \\ &= C_{\max}(S^d) + (m+n-1)\,\underline{p}. \end{aligned} \tag{9}$$

Thus, if $C_{\max}(S^d)$ achieves its minimum value, then $C_{\max}(S)$ is minimum as well.

Following the approach from Sect. 4.1, construct an optimal schedule S_*^d by McNaughton's wrap-around algorithm, using an arbitrary job permutation. Note that, by construction, schedule S_*^d is of permutation type. The illustrative example presented in Fig. 2 satisfies this requirement. Without loss of generality, we assume that the jobs are sequenced in the order of their numbering, and the same job order is used in an optimal solution S_*^e to I^e.

Consider the aggregate schedule S_*, obtained as a merger of S_*^d and S_*^e. Due to (9), S_* is an optimal schedule among all permutation schedules, and by Theorem 3 it is globally optimal among all schedules.

The most time-consuming step in the described approach is the merger of S_*^d and S_*^e. Its time complexity is $O(nm)$, and it defines the overall time complexity for constructing a complete optimal schedule for $F|plbl(\underline{p})|C_{\max}$.

Following the ideas of a compact encoding of an optimal solution, known in the context of high-multiplicity scheduling problems (see, e.g., Brauner et al. 2005), we specify formulae for starting times of all operations, each of which can be computed in $O(1)$ time, provided a special $O(n)$ preprocessing is done.

At the preprocessing stage, the diminished instance I^d is analyzed and the calculations related to McNaughton's wrap-around algorithm are performed. The optimal schedule S_*^d can be specified by $m-1$ split jobs, which define three types of operations in S_*^d: zero-length *initial* operations $\mathcal{I}$, zero-length *final* operations $\mathcal{F}$, and the remaining nonzero *middle* operations $\mathcal{M}$ characterized by starting times $t_{ij}(S_*^d)$ and processing times $p_{ij}(S_*^d)$ for operations O_{ij}.

After the merger of the two schedules S_*^d and S_*^e, the aggregate initial and final operations $\mathcal{I} \cup \mathcal{F}$ become of length $\underline{p}$, while the lengths of the middle operations $\mathcal{M}$ increase by $\underline{p}$; see Fig. 2 where the middle operations $\mathcal{M}$ are represented as shaded boxes. For the resulting schedule, the processing times and the starting times are calculated as

$$p_{ij}(S_*) = p_{ij}(S_*^d) + \underline{p}, \tag{10}$$

$$t_{ij}(S_*) = (i+j-2)\,\underline{p} + \begin{cases} 0, & \text{if } O_{ij} \in \mathcal{I}, \\ t_{ij}(S_*^d), & \text{if } O_{ij} \in \mathcal{M}, \\ C_*(M_i), & \text{if } O_{ij} \in \mathcal{F}, \end{cases} \tag{11}$$

where $C_*(M_i)$ is the completion time of the last operation on machine M_i in schedule S_*^d, $1 \le i \le m$.

Theorem 8 *An optimal schedule for problem $F|plbl(\underline{p})|C_{\max}$ can be specified by formulae (10), (11) for the processing times and starting times of all nm operations, each computable in $O(1)$ time provided the $O(n)$ preprocessing is done. The optimal makespan is*

$$\begin{aligned} C_*^{\text{FlowShop}} = \max\Bigg\{ &\frac{1}{m} p(\mathcal{J}) + (m-1)\underline{p}, \\ &\max_{j\in\mathcal{J}} \{p_j\} + (n-1)\underline{p} \Bigg\}, \end{aligned} \tag{12}$$

where $p(\mathcal{J})$ is the total processing time of all jobs.

Notice that the makespan formula (12) follows from (9):

$$C_*^{\text{FlowShop}} = C_*^d + (m+n-1)\underline{p}, \tag{13}$$

where C_*^d is the optimal makespan of the diminished instance calculated as

$$C_*^d = \max\left\{ \frac{1}{m} p^d(\mathcal{J}), \max_{j\in\mathcal{J}} \left\{p_j^d\right\} \right\}.$$

Here $p_j^d = p_j - m\underline{p}$ is the processing time of job j in the diminished instance I^d and $p^d(\mathcal{J})$ is the total processing time of all jobs in the diminished instance.

5.2 Problem $F|plbl(\underline{p})|L_{\max}$

Now, we consider problem $F|plbl(\underline{p})|L_{\max}$. By Theorem 3, it is sufficient to consider permutation schedules. The following lemma justifies that we can fix the job sequence in accordance with the earliest due date order (EDD).

Lemma 2 *For problem $F|plbl(\underline{p})|L_{\max}$, there exists an optimal permutation schedule with the jobs sequenced in the EDD order.*

The lemma can be proved using pairwise interchange arguments by swapping adjacent jobs violating the EDD order and verifying that the $L_{\max}$-value does not increase. Note that the swapping of adjacent jobs is always feasible, as established in Lemma 1.

Given an instance I of $F|plbl(\underline{p})|L_{\max}$, renumber the jobs so that $d_1 \le d_2 \le \cdots \le d_n$, and apply the job disaggregation methodology of Sect. 3.2.2. Define the two instances:

Instance I^e :	Instance I^d :
$p_j^e = m\underline{p}$,	$p_j^d = p_j - m\underline{p}$,
$d_j^e = (j+m-1)\,\underline{p}$,	$d_j^d = d_j - (j+m-1)\,\underline{p}$.

Notice that the original instance I satisfies

$$p_j = p_j^e + p_j^d, \quad d_j = d_j^e + d_j^d.$$

In the class of permutation schedules with the fixed job sequence $(1, 2, \ldots, n)$, let S^e, S^d and S be the schedules for instances I^e, I^d and I, respectively. Note that S^e is the same as the top left schedule in Fig. 2 with $L_{\max}(S^e) = 0$. By Theorem 4,

$$\begin{aligned} L_j(S) = C_{mj} - d_j &= C_{mj}^d + (j+m-1)\underline{p} - d_j \\ &= C_{mj}^d - d_j^d = L_j(S^d), \end{aligned} \tag{14}$$

so that $L_{\max}(S) = L_{\max}(S^d)$.

An optimal F-type schedule S^d with the EDD job sequence $(1, 2, \ldots, n)$ can be constructed by the algorithm from Baptiste (2000); see Sect. 4.2. Combining S^d (with the smallest possible value of $L_{\max}$) and S^e (with the same job sequence) delivers an optimal schedule S for the original instance I. The most time-consuming step is the algorithm from Baptiste (2000), which takes $O(n \log n + mn)$ time, dominating the time needed to renumber the jobs in the EDD order and the time for combining the two schedules.

Theorem 9 *Problem $F|plbl(\underline{p})|L_{\max}$ is solvable in $O(n \log n + mn)$ time.*

Note that as opposed to Sect. 5.1, we cannot eliminate the term mn from the complexity estimate by introducing compact encoding. Indeed, even the easier problem $F|plbl|L_{\max}$ with unrestricted pliability, which needs to be solved as a subproblem, already requires $O(n \log n + mn)$ time; see Sect. 4.2.

5.3 Problem $O|plbl(\underline{p})|C_{\max}$

The open shop problem $O|plbl(\underline{p})|C_{\max}$ appears to be much harder to handle than the corresponding flow shop version. While for the model studied in the previous section, F-type permutation schedules have a well-defined structure, the O-type schedules for the current model provide a greater level of flexibility. Another difficulty comes from the optimality check: in contrast to problem $O|plbl|C_{\max}$, there exist instances for $O|plbl(\underline{p})|C_{\max}$ where the lower bound C_*, defined by (8) for the related parallel machine problem, cannot be reached. For example, consider instances I_1 and I_2 as follows.

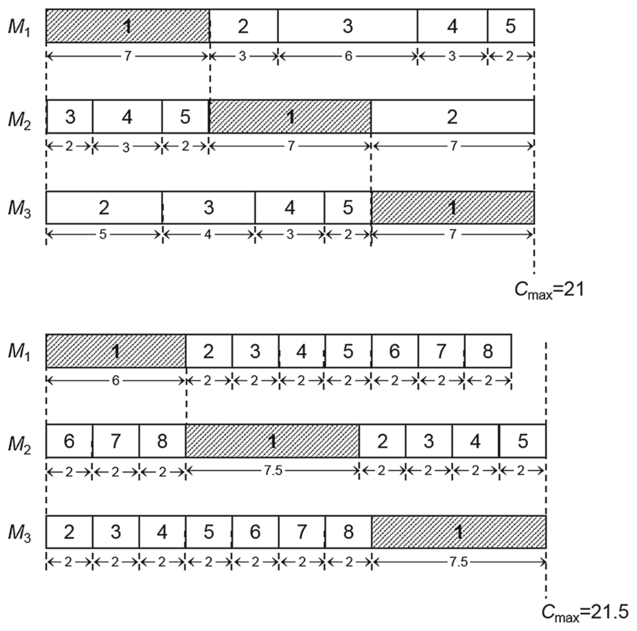

Fig. 5 Optimal schedules for instances I_1 (top) and I_2 (bottom)

Instance I_1: $m = 3$, $\underline{p} = 2$

j	1	2	3	4	5
p_j	21	15	12	9	6

Instance I_2: $m = 3$, $\underline{p} = 2$

j	1	2	3	4	5	6	7	8
p_j	21	6	6	6	6	6	6	6

The average machine workload $\frac{1}{3}p(\mathcal{J})$ and the processing time of the longest job p_1 have the same value of 21, resulting in the lower bound value $C_* = 21$. For instance I_1 that lower bound can be achieved, while for instance I_2 the lower bound is unachievable: it is impossible to process job 1 without some waiting time in-between its operations, while observing the restriction $\underline{p} = 2$ and ensuring that no machine is idle before it finishes processing all jobs. This is due to the fact that all operations of all other jobs are of even length 2, but job 1 has odd processing time and cannot be split into three even length operations; see Fig. 5 for an illustration.

In the following, we consider the cases from Table 4 where problem $O|plbl(\underline{p})|C_{\max}$ can be solved efficiently. We use the notations p_1, p_2 and p_3 for the processing times of the three longest jobs, assuming $p_1 \ge p_2 \ge p_3$.

In Case 1 we adjust the approach from Sect. 5.1 developed for problem $F|plbl(\underline{p})|C_{\max}$. Using the concept of the

Table 4 Solvable cases of problem $O|plbl(\underline{p})|C_{\max}$

Case	Conditions	Makespan
1	$p_1 + n\underline{p} \le \frac{p(\mathcal{J})}{m} + m\underline{p}$	$\frac{p(\mathcal{J})}{m}$
2	$m = 2$	$\max\left\{p_1, \frac{p(\mathcal{J})}{m}\right\}$
3	$m = 3, p_1 = p_2$	$\max\left\{p_1, \frac{p(\mathcal{J})}{m}\right\}$
4	$m = 3,\ p_1 > p_2$ and $p(\mathcal{J}) \le 2p_1 + p_2$	p_1
5	$m = 3,\ p_1 > p_2$ and $p(\mathcal{J}) \ge 3p_1 + p_3$	$\frac{p(\mathcal{J})}{m}$

diminished instance with $p_j^d = p_j - m\underline{p}$, the inequality that defines Case 1 can be rewritten as

$$\left(p_1^d + m\underline{p}\right) + n\underline{p} \le \frac{p^d(\mathcal{J}) + n \cdot m\underline{p}}{m} + m\underline{p}$$

or equivalently

$$\max_{j \in \mathcal{J}} \left\{p_j^d\right\} \le \frac{1}{m} p^d(\mathcal{J}). \tag{15}$$

It implies that there exists an optimal flow shop schedule for problem $F|plbl(\underline{p})|C_{\max}$ with

$$C_*^{\text{FlowShop}} = \frac{1}{m} p(\mathcal{J}) + (m-1)\underline{p};$$

see Theorem 8. An optimal open shop schedule can be obtained from the optimal flow shop schedule by moving to the front the last $k-1$ operations of length $\underline{p}$ on every machine M_k, $2 \le k \le m$. These moves do not cause any conflicts since $n \ge m$, and the flow shop makespan value C_*^{FlowShop} decreases by $(m-1)\underline{p}$:

$$C_*^{\text{OpenShop}} = C_*^{\text{FlowShop}} - (m-1)\underline{p} = \frac{1}{m} p(\mathcal{J}).$$

By the same theorem, an optimal schedule can be fully defined in $O(nm)$ time.

Case 2 follows immediately from the more general result for problem $O2|plbl(\underline{p}_{ij}, \overline{p}_{ij})|C_{\max}$, which we present in Sect. 6.2.

Optimal schedules for Cases 1-2 have a common permutation-like structure, with job sequences on machines $M_1, M_2, \ldots, M_m$ selected from the set of sequences

$$\begin{array}{llllll}
(1, & 2,\ 3,\ \ldots,\ n-2, & n-1, & n), \\
(n, & 1,\ 2,\ \ldots,\ n-3, & n-2, & n-1), \\
(n-1,\ n, & 1,\ \ldots,\ n-4, & n-3, & n-2), \\
 & \vdots & & \\
(3, & 4,\ 5,\ \ldots, & n, & 1, & 2), \\
(2, & 3,\ 4,\ \ldots,\ n-1, & n, & 1).
\end{array}$$

Conditions that define Cases 3–5 also result in optimal schedules with permutation-like structure. They were derived by Koulamas and Kyparisis (2015) for the open shop problem with $m = 3$ machines and with each job consisting of equal-size operations. Due to our assumption $p_j \ge m\underline{p}$ (necessary for feasibility), splitting each job into m equal-size operations of length $\frac{p_j}{m}$ leads to a feasible schedule for the model with pliable jobs. Moreover, the makespan of each optimal schedule for Cases 3-5 achieves the lower bound C_*. Therefore, the resulting schedules are optimal for $O|plbl(\underline{p})|C_{\max}$.

The longest processing time order, assumed in Koulamas and Kyparisis (2015) for the whole set of jobs $\mathcal{J}$, is not needed once the three longest jobs $\{1, 2, 3\}$ are identified, so that the optimal schedules in Cases 3-5 can be found in $O(n)$ time.

It is likely that the permutation-like property holds for the general case of problem $O|plbl(\underline{p})|C_{\max}$. An optimal job splitting may violate the equal-size property, with possibly unequal splitting of a job into m operations, as illustrated by the top schedule of Fig. 5. However, a proportionate open shop schedule, where jobs are split into equal-size operations, can be a good starting point for identifying the boundary jobs, processed at the beginning and at the end on each machine. The optimal operation lengths can then be found via linear programming. Unfortunately, we were unable to prove the correctness of the outlined approach and leave it for future research.

6 Restricted pliability with individual lower and upper bounds: Makespan objective

In this section, we consider flow shop and open shop problems with $m = 2$ machines, the makespan objective, and individual lower and upper bounds on operation processing times. To simplify the notation, we denote the two machines by A and B, and the lower and upper bounds of the operations by $\underline{a}_j, \overline{a}_j$ (for machine A) and by $\underline{b}_j, \overline{b}_j$ (for machine B). The objective is to find an order of the jobs for each machine and the lengths a_j and b_j for A- and B-operations for every job j, $1 \le j \le n$, so that

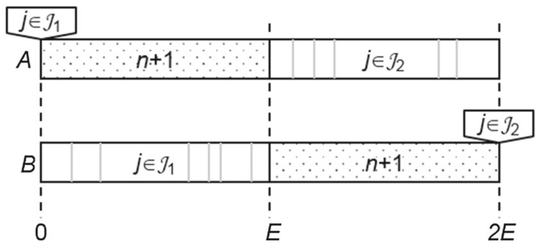

Fig. 6 An optimal solution to the instance of the flow shop problem

$$a_j + b_j = p_j,$$
$$\underline{a}_j \le a_j \le \overline{a}_j,$$
$$\underline{b}_j \le b_j \le \overline{b}_j,$$

and the makespan is minimized.

6.1 Problem $F2|plbl(\underline{p}_{ij}, \overline{p}_{ij})|C_{\max}$

We demonstrate that problem $F2|plbl(\underline{p}_{ij}, \overline{p}_{ij})|C_{\max}$ is NP-hard and its special case with a fixed job order can be solved via linear programming. Interestingly, the counterpart of the problem with flexible operations is NP-hard in both cases; see Gupta et al. (2004) and Lin et al. (2016).

Theorem 10 *Problem $F2|plbl(\underline{p}_{ij}, \overline{p}_{ij})|C_{\max}$ is NP-hard.*

Proof Consider an instance of PARTITION with integers $e_1, \ldots, e_n$ and $\sum_{j=1}^{n} e_j = 2E$. The objective is to decide whether a set $\mathcal{J}_1 \subset \{1, 2, \ldots, n\}$ exists with $\sum_{i\in\mathcal{J}_1} e_i = E$. We construct an instance of the flow shop problem with $n+1$ jobs:

j	1	2	$\cdots$	n	$n+1$
p_j	e_1	e_2	$\cdots$	e_n	$2E$
$[\underline{a}_j, \overline{a}_j]$	$[0, e_1]$	$[0, e_2]$	$\cdots$	$[0, e_n]$	$[E, E]$
$[\underline{b}_j, \overline{b}_j]$	$[0, e_1]$	$[0, e_2]$	$\cdots$	$[0, e_n]$	$[E, E]$

Notice that job $n+1$ has a fixed splitting, with two operations of length E, and for any permutation schedule it partitions the remaining jobs into two subsets, jobs $\mathcal{J}_1$ preceding $n+1$ and jobs $\mathcal{J}_2$ which follow it.

It is easy to verify that PARTITION has a solution if and only if a flow shop schedule of makespan $C_{\max} = 2E$ exists; see Fig. 6 for an illustration, where A-operations of jobs $\mathcal{J}_1$ and B-operations of jobs $\mathcal{J}_2$ have 0 length. □

The problem becomes solvable via linear programming if a job sequence is fixed, even in the case of more than two machines and for more general objective functions. For this, we need to extend the LP formulation (4) by Prot et al. (2013), adding box inequalities $\underline{p}_{ij} \le x_{ij} \le \overline{p}_{ij}$ for all variables x_{ij}.

6.2 Problem $O2|plbl(\underline{p}_{ij}, \overline{p}_{ij})|C_{\max}$

Solving problem $O2|plbl(\underline{p}_{ij}, \overline{p}_{ij})|C_{\max}$ involves two decisions: finding the job splitting $p_j = a_j + b_j$ for all jobs $j \in \mathcal{J}$, and sequencing the operations with fixed lengths on two machines to minimize the makespan. The second task can be done in $O(n)$ time using the well-known algorithm by Gonzalez and Sahni (1976), which constructs an optimal schedule with the makespan

$$C_{\max} = \max\left\{\sum_{j\in\mathcal{J}} a_j,\ \sum_{j\in\mathcal{J}} b_j,\ p_q\right\}. \tag{16}$$

Here, the first two terms correspond to the loads of machines A and B, while the last term is the processing time of a longest job q,

$$p_q = \max\left\{p_j | j \in \mathcal{J}\right\}. \tag{17}$$

In what follows we formulate three LP problems $LP(A)$, $LP(B)$, and $LP(q)$, one for each term in (16), aimed at finding an optimal job splitting. Notice that some of the problems may be infeasible. An optimal solution is selected among the solutions to these three problems as the one with the smallest makespan value.

Consider first problem $LP(A)$ formulated for the class of schedules with $C_{\max} = \sum_{j\in\mathcal{J}} a_j$, assuming that the first term in (16) determines the maximum. If Δ denotes the difference between the third and the first term in (16), then this class of schedules is characterized by $\sum_{j\in\mathcal{J}} a_j = p_q + \Delta \ge \sum_{j\in\mathcal{J}} b_j$, where $\Delta \ge 0$, and minimizing the makespan is equivalent to minimizing Δ:

$$\begin{aligned}
LP(A):\ \min\ & \Delta \\
\text{s.t.}\ & \textstyle\sum_{j\in\mathcal{J}} a_j = p_q + \Delta, \\
& \textstyle\sum_{j\in\mathcal{J}} b_j \le p_q + \Delta, \\
& a_j + b_j = p_j, \qquad j \in \mathcal{J}, \\
& \underline{a}_j \le a_j \le \overline{a}_j, \qquad j \in \mathcal{J}, \\
& \underline{b}_j \le b_j \le \overline{b}_j, \qquad j \in \mathcal{J}, \\
& \Delta \ge 0.
\end{aligned}$$

From the first and the third constraints we derive the following expressions for Δ and b_j:

$$\Delta = \sum_{j\in\mathcal{J}} a_j - p_q,$$
$$b_j = p_j - a_j, \qquad j \in \mathcal{J}, \tag{18}$$

and rewrite $LP(A)$ as follows:

$$\begin{aligned} LP'(A): \min\ & \sum_{j\in\mathcal{J}} a_j \\ \text{s.t.}\ & \sum_{j\in\mathcal{J}} a_j \ge \max\left\{\tfrac{1}{2}p(\mathcal{J}),\ p_q\right\}, \\ & \ell_j \le a_j \le u_j, \quad j\in\mathcal{J}, \end{aligned}$$

where

$$p(\mathcal{J}) = \sum_{j\in\mathcal{J}} p_j,$$

$$\ell_j = \max\left\{\underline{a}_j,\ p_j - \overline{b}_j\right\},\ u_j = \min\left\{\overline{a}_j,\ p_j - \underline{b}_j\right\}. \tag{19}$$

The resulting problem is the knapsack problem with continuous variables a_j, $j\in\mathcal{J}$, solvable in $O(n)$ time (Balas and Zemel 1980).

Problem $LP(B)$ is formulated similarly for the class of schedules with $C_{\max} = \sum_{j\in\mathcal{J}} b_j$; it is also solvable in $O(n)$ time.

Consider now problem $LP(q)$ formulated for the class of schedules with $C_{\max} = p_q$. Since the makespan value is constant, there is no objective function to minimize, and we only need to find a feasible solution with respect to the following constraints:

$$\begin{aligned} LP(q):\ & \sum_{j\in\mathcal{J}\setminus\{q\}} a_j \le b_q, \\ & \sum_{j\in\mathcal{J}\setminus\{q\}} b_j \le a_q, \\ & a_j + b_j = p_j, \quad j\in\mathcal{J}, \\ & \underline{a}_j \le a_j \le \overline{a}_j, \quad j\in\mathcal{J}, \\ & \underline{b}_j \le b_j \le \overline{b}_j, \quad j\in\mathcal{J}. \end{aligned}$$

Using expression $b_j = p_j - a_j$ for $j\in\mathcal{J}$ we obtain:

$$\begin{aligned} LP'(q):\ & \sum_{j\in\mathcal{J}} a_j \le p_q, \\ & \sum_{j\in\mathcal{J}} a_j \ge p(\mathcal{J}) - p_q, \\ & \ell_j \le a_j \le u_j, \qquad j\in\mathcal{J}, \end{aligned}$$

where ℓ_j and u_j are given by (19). The latter problem can be solved in $O(n)$ time by performing the following steps.

1. Compute $a_* = \sum_{j\in\mathcal{J}} \ell_j$, the smallest value of $\sum_{j\in\mathcal{J}} a_j$.
2. If a_* satisfies both main conditions, i.e., $p(\mathcal{J}) - p_q \le a_* \le p_q$, then stop: a feasible solution is found.
3. If $a_* > p_q$, then stop: problem $LP'(q)$ is infeasible.
4. If $a_* < p(\mathcal{J}) - p_q$, then solve the LP problem

$$\begin{aligned} \max\ & \sum_{j\in\mathcal{J}} a_j \\ \text{s.t.}\ & \sum_{j\in\mathcal{J}} a_j \le p_q, \\ & \ell_j \le a_j \le u_j,\ j\in\mathcal{J}, \end{aligned} \tag{20}$$

and verify whether for the found solution the required condition $\sum_{j\in\mathcal{J}} a_j \ge p(\mathcal{J}) - p_q$ is satisfied. Problem (20) is again the knapsack problem with continuous variables a_j, $j\in\mathcal{J}$, solvable in $O(n)$ time.

To summarize, each of the problems $LP(A)$, $LP(B)$ and $LP(q)$ can be solved in $O(n)$ time, and this is the overall time complexity of the described approach for solving $O2|plbl(\underline{p}_{ij}, \overline{p}_{ij})|C_{\max}$.

Theorem 11 *Problem $O2|plbl(\underline{p}_{ij}, \overline{p}_{ij})|C_{\max}$ is solvable in $O(n)$ time.*

We conclude this section by reviewing the results for another related problem, namely the parallel machine problem with *restricted preemption* and the makespan objective studied by Ecker and Hirschberg (1993), Baranski (2011), and Pienkosz and Prus (2015). In that problem, job preemption may happen several times, but each job part has to be at least $\underline{p}$ time units long, for some lower bound $\underline{p}$. Notice that unlike the pliability problem $O|plbl(\underline{p})|C_{\max}$, it is not required that every job is split exactly into m pieces, one per machine.

Polynomial-time algorithms for the parallel machine problem with restricted preemption where no job part may be shorter than $\underline{p}$ are known only for special cases with two machines; see Table 5. The complexity of the general case with $m = 2$ is left as open in the literature. Interestingly, our algorithm presented in this section finds an optimal schedule not only for the pliability problem $O2|plbl(\underline{p}_{ij}, \overline{p}_{ij})|C_{\max}$, but it also solves the problem with restricted preemption for $m = 2$ machines, assuming $\underline{p}_{ij} = \underline{p}$ for all operations and $p_j \ge 2\underline{p}$.

Table 5 Solvable cases of the parallel machine problem with $m = 2$ and restricted preemption

Case	Conditions	Makespan
1	$p_1 \ge \frac{1}{2}p(\mathcal{J})$	p_1
2	$p_1 \ge 4\underline{p}$	$\max\{p_1, \frac{1}{2}p(\mathcal{J})\}$
3	$p_1 \ge 3\underline{p}$, $p_2 \ge 2\underline{p}$	$\max\{p_1, \frac{1}{2}p(\mathcal{J})\}$

7 Unrestricted and restricted pliability: Minsum objectives

In this section, we consider pliability problems with minsum objectives, focusing on problems with unrestricted pliability and restricted pliability with a common lower bound $\underline{p}$. The restricted problems of type (iii) are strongly NP-hard since by Proposition 1 they are not easier than the related classical problems $F2||f$ and $O2||f$, which are known to be strongly NP-hard for all traditional minsum objectives f_Σ; see Brucker (2007).

7.1 Problems $F|plbl|\sum C_j$ and $O|plbl|\sum C_j$

As observed in Sect. 3.1, problem $F|plbl|\sum C_j$ can be solved in polynomial time via linear programming, considering a fixed sequence of job completion times corresponding to the shortest processing time (SPT) order; the optimality of that order for $P|pmtn|\sum C_j$ is stated, e.g., in Conway et al. (1967).

A faster algorithm is based on the approach which constructs an optimal F-type schedule for problem $P|pmtn|\sum C_j$, formulated by Bruno and Gonzalez (1976) and Labetoulle et al. (1984) for the more general problem $Q|pmtn|\sum C_j$, where machines have different processing speeds. The algorithm can be described as follows.

Algorithm F-Sum

1. Construct an SPT schedule by assigning a shortest job to the earliest available machine, breaking ties arbitrarily.
2. Consider time intervals $\mathcal{I}_u = \left[C_{u-1}, C_u\right]$, $1 \le u \le n$, defined by the completion times C_u of the jobs; for completeness set $C_0 = 0$. In each interval $\mathcal{I}_u$, reallocate the job parts so that any machine M_i, $1 \le i \le m$, processes the job with the index $u + m - i$ in that interval, if $u + m - i \le n$, and no job otherwise; see the bottom schedule in Fig. 7.

Steps 1 and 2 of the algorithm are illustrated by the two schedules of Fig. 7. Note that Step 1 constructs an optimal schedule for problems $P||\sum C_j$ and $P|pmtn|\sum C_j$, while Step 2 reshuffles operation parts without increasing completion times of individual jobs, producing an F-type solution for $P|pmtn|\sum C_j$. By Corollary 1, the resulting schedule is optimal for $F|plbl|\sum C_j$.

The combined time complexity of the two steps is $O(n \log n + mn)$. Following the ideas of compact encoding of an optimal solution, we can use the following function $J(i, u)$ that specifies for each machine-interval pair (i, u) the job which is processed by machine M_i in interval $\mathcal{I}_u$:

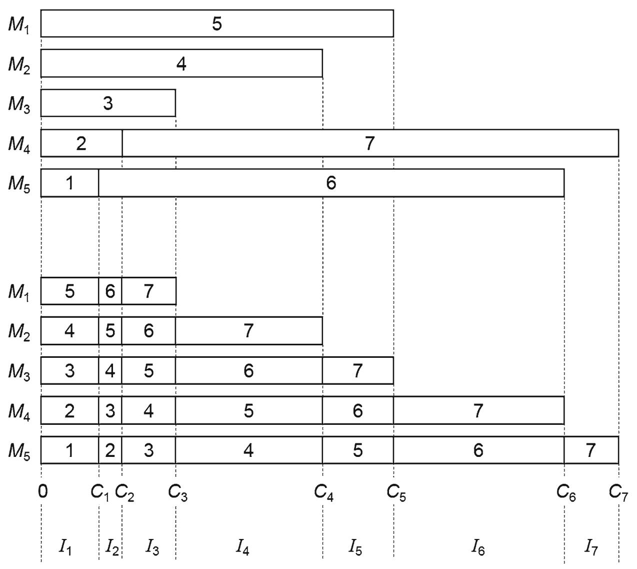

Fig. 7 Two schedules with the same completion times for all jobs: an optimal schedule for problems $P||\sum C_j$ and $P|pmtn|\sum C_j$ (above) and an optimal schedule for problem $F|plbl|\sum C_j$ (below)

$$J(i, u) = \begin{cases} u + m - i, & \text{if } u + m - i \le n \\ 0, & \text{otherwise}, \end{cases} \qquad (21)$$

where the job index 0 means that no job is assigned.

Theorem 12 *An optimal schedule for problem $F|plbl|\sum C_j$ can be constructed in $O(n \log n + mn)$ time. It can be specified by formula (21), computable in $O(1)$ time for each machine-interval pair (i, u), provided the $O(n \log n)$ preprocessing is done and n intervals $\mathcal{I}_u$, $1 \le u \le n$, are found using Step 1 of Algorithm F-Sum.*

Consider now problem $O|plbl|\sum C_j$. The task of constructing an O-type schedule optimal for problem $P|pmtn|\sum C_j$ is a simpler task than constructing an F-type schedule. It is sufficient to adjust an SPT schedule produced by Step 1 of Algorithm F-Sum by adding zero-length operations in the beginning of the schedule, so that every job has an operation on every machine.

Theorem 13 *Problem $O|plbl|\sum C_j$ is solvable in $O(n \log n)$ time.*

7.2 Problem $F|plbl(\underline{p})|\sum C_j$

In order to solve problem $F|plbl(\underline{p})|\sum C_j$, we use the disaggregation methodology from Sect. 3.2.2, which results in two simplified instances: one instance of problem $F|plbl|\sum C_j$ with unrestricted pliability, which can be solved by the approach from the previous section, and one instance with equal processing times. Let the two instances be I^d and I^e, as in Definition 1, and the corresponding schedules be S^d and S^e. We assume that S^d and S^e satisfy conditions (D1)-(D3) required for Theorem 4. Note that if (D3) is not satisfied for S^d, i.e., if there are idle times on some machines, then they can be eliminated, without increasing job completion times, by left-shifting operations or by moving processing load to downstream machines.

By Theorem 4, the schedule S obtained as a merger of S^d and S^e satisfies:

$$\sum_{j=1}^{n} C_j = \sum_{j=1}^{n} C_j^d + \sum_{j=1}^{n} C_j^e.$$

As the objective value $\sum_{j=1}^{n} C_j^e$ of schedule S^e for instance I^e is the same for any permutation schedule S^e, $\sum_{j=1}^{n} C_j$ achieves its minimum value if and only if $\sum_{j=1}^{n} C_j^d$ is minimum. Thus, an optimal schedule for problem $F|plbl(\underline{p})|\sum C_j$ can be found as a merger of schedule S^e defined in Sect. 3.2.2 and schedule S^d constructed by Algorithm F-Sum for instance I^d as in Sect. 7.1.

The merger involves $n+m-1$ intervals of schedule S^e, which we denote by $\mathcal{I}'_v$, $1 \le v \le (m-1)+n$, and n intervals of schedule S^d, defined in the previous section as $\mathcal{I}_u$, $1 \le u \le n$. Notice that the last n intervals of schedule S^e have exactly the same job allocation as the n intervals of schedule S^d. Thus, as a result of the merger, the combined schedule gets the first $(m-1)$ intervals of length $\underline{p}$ taken from S^e, and the next n intervals taken from S^e and S^d; resulting intervals are of length $|\mathcal{I}_u| + \underline{p}$, $1 \le u \le n$. We modify the function $J(i,u)$, introduced in the previous section for a compact encoding of schedule S^d. For the current problem $F|plbl(\underline{p})|\sum C_j$, function $J'(i,v)$ specifies for each machine-interval pair (i,v) the job which is processed by machine M_i in the v-th interval:

$$J'(i,v) = \begin{cases} v-i+1, & \text{if } 1 \le v \le (m-1)+n \\ & \text{and } \ 1 \le v-i+1 \le n, \\ 0, & \text{otherwise,} \end{cases} \tag{22}$$

where the job index 0 means that no job is assigned. Here the expression $v-i+1$ corresponds to the main expression $u+m-i$ from (21) after substituting $u = v-(m-1)$, the link between the u-th interval of S^d and the v-th interval of S^e.

Theorem 14 *An optimal schedule for problem $F|plbl(\underline{p})|\sum C_j$ can be constructed in $O(n \log n + mn)$ time by defining the lengths of $(m-1)+n$ intervals and mn operations allocated to them. It can be specified by formula (22), computable in $O(1)$ time for each machine-interval pair (i,v), provided the $O(n \log n)$ preprocessing is done and n intervals $\mathcal{I}_u$, $1 \le u \le n$, are found using Step 1 of Algorithm F-Sum.*

7.3 Problem $O|plbl(\underline{p})|\sum C_j$

We find an optimal schedule for problem $O|plbl(\underline{p})|\sum C_j$ by constructing a common optimal schedule for problems $P||\sum C_j$ and $P|pmtn|\sum C_j$ and reorganizing its structure in order to achieve a solution of O-type.

Without loss of generality, we assume that n is a multiple of m, i.e., $n = Qm$ for some integer Q. Otherwise, we add as many jobs of maximum length as needed to satisfy this condition (at most $m-1$ jobs are sufficient). Then we apply the approach described below, which would place the longest jobs at the end of the schedule, and remove the added jobs from the resulting schedule.

Construct an SPT schedule by assigning a shortest job to the earliest available machine; see the top schedule in Fig. 8. In what follows, we assume that the jobs are renumbered in SPT order, and the job numbering is from 0 up to $n-1$ (in order to improve readability of the formulae for an optimal schedule).

Consider Q sections of the schedule, each of length $m\underline{p}$: the first section is given by the interval $\left[0, m\underline{p}\right]$ and the remaining sections by intervals $\left[C_{qm-1}, C_{qm-1} + m\underline{p}\right]$, $1 \le q \le Q-1$. In each section q, there are m jobs $\mathcal{J}_q = \{j = qm + r | 0 \le r \le m-1\}$ allocated to m machines, one job per machine. To justify this, notice that all jobs from $\mathcal{J}_0$ start at time 0 and have processing times no less than $m\underline{p}$. The property holds for every subsequent section q since

$$\begin{aligned} C_{qm} - C_{qm-1} &= \sum_{\eta=0}^{q} p_{\eta m} - \sum_{\eta=1}^{q} p_{\eta m-1} \\ &= p_0 + \sum_{\eta=1}^{q} \left(p_{\eta m} - p_{\eta m-1}\right) \ge p_0 \ge m\underline{p}, \end{aligned}$$

and $C_j \ge C_{qm}$ for any $j \in \mathcal{J}_q$. Note that each job appears in exactly one of the Q sections.

In order to produce a feasible O-type schedule with m operations per job, each of length no less than $\underline{p}$, we adopt a two-stage approach: first redistribute the jobs within every section q, $0 \le q \le Q-1$; next redistribute the jobs in-between the sections. Stage 1 ensures that every machine gets one operation of every job of length $\underline{p}$; Stage 2 places a nonzero operation of a job from in-between two sections next to the $\underline{p}$-operation of the same job and combines them into one operation, see Fig. 8 for an illustration.

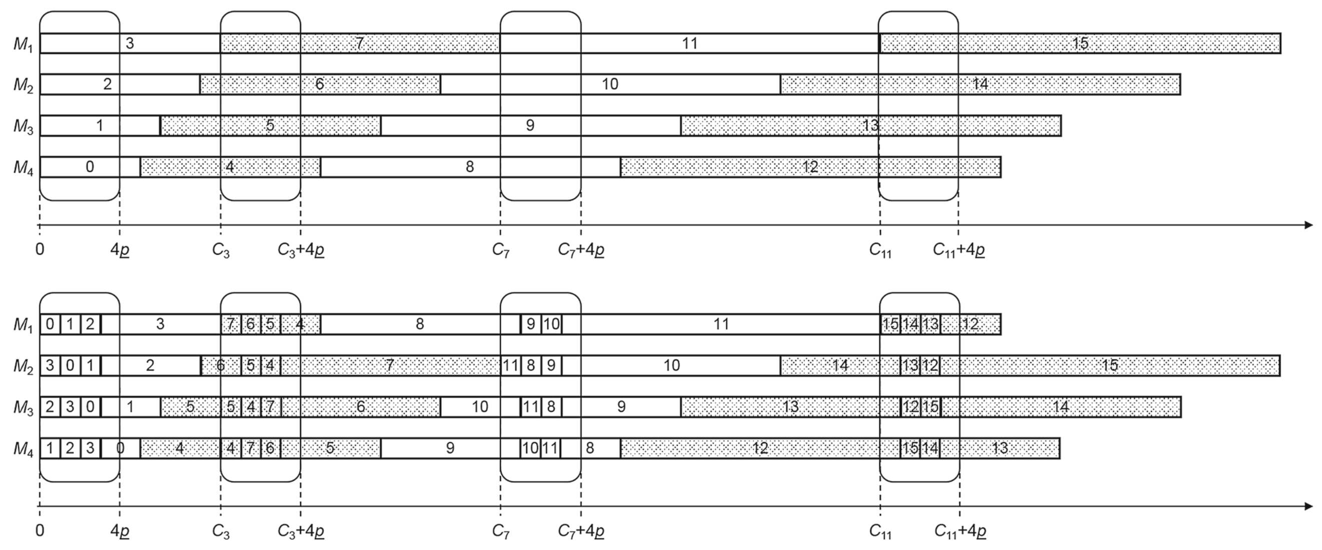

Fig. 8 Two schedules with the same completion times for all jobs: an optimal schedule for problems $P||\sum C_j$ and $P|pmtn|\sum C_j$ (above) and an optimal schedule for problem $O|plbl(\underline{p})|\sum C_j$ (below)

Formally, in Stage 1 we split every section q into m subintervals of length $\underline{p}$ and reshuffle the jobs in each subinterval in a wrap-around manner. The reshuffling is done slightly differently, depending on whether q is even or odd:

Job $j = qm + r$ (even q)	Machine index for the k-th operation of job j			
	$k = 1$	$k = 2$	$\cdots$	$k = m$
$qm + 0$	1	2	$\cdots$	m
$qm + 1$	m	1	$\cdots$	$m - 1$
$qm + 2$	$m - 1$	m	$\cdots$	$m - 2$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$qm + (m - 1)$	2	3	$\cdots$	1

Job $j = qm + r$ (odd q)	Machine index for the k-th operation of job j			
	$k = 1$	$k = 2$	$\cdots$	$k = m$
$qm + 0$	m	$m - 1$	$\cdots$	1
$qm + 1$	$m - 1$	$m - 2$	$\cdots$	m
$qm + 2$	$m - 2$	$m - 3$	$\cdots$	$m - 1$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$qm + (m - 1)$	1	m	$\cdots$	2

After Stage 1, the following property holds for any pair of jobs $j \in \mathcal{J}_q$ and $j + m \in \mathcal{J}_{q+1}$, which were adjacent in the initial schedule: the last $\underline{p}$-operation of j in section q and the first $\underline{p}$-operation of $j + m$ in the next section are assigned to the same machine (compare the last column of the top table with the first column of the bottom one). Due to this, at Stage 2 we can rearrange the job parts in-between the sections while keeping j and $j + m$ adjacent: the last part of job j is placed on the same machine as the last $\underline{p}$-operation of that job in section q, and the two operations are merged; the first part of $j + m$ is placed on the same machine as the first $\underline{p}$-operation of that job in the section $q + 1$, and the two operations are also merged; see the bottom schedule in Fig. 8. Thus, we create a schedule with exactly m operations per job, the $m - 2$ middle operations are of minimum length $\underline{p}$ while the first and last operation of each job may be larger.

The operation lengths are thus defined as follows:

(a) all intermediate operations of any job j, corresponding to position $2 \le k \le m - 1$, have the common length $\underline{p}$;
(b) all first operations of section $q = 0$ have the common length $\underline{p}$;
(c) for any job $j = qm + r$ processed in position $k = 1$ in section q, $1 \le q \le Q - 1$, its first operation is merged with the part of that job positioned just before section q; the combined length is $C_{qm-1} - C_{j-m} + \underline{p}$, where C_{qm-1} is the starting point of section q and C_{j-m} is the completion time of the job $j - m$ which precedes job j in the initial schedule;
(d) for any job $j = qm + r$ processed in position $k = m$ in section q, $0 \le q \le Q - 1$, its last operation is merged with the part of that job positioned just after section q; their combined length is $C_j - \left(C_{qm-1} + m\underline{p}\right) + \underline{p}$, where C_j is the completion time of the last part of job j in the initial schedule and $\left(C_{qm-1} + m\underline{p}\right)$ is the end point of section q (assuming $C_{-1} = 0$ for completeness).

For a compact encoding, the machine order can be represented by the function $M(j, k)$ which defines the machine index for the k-th operation of job $j = qm + r$ that appears in the schedule (this is not necessarily the operation on machine M_k):

$$M(qm+r,k) = \\ = \begin{cases} k-r, & \text{if } q \text{ is even and } k-r \geq 1, \\ m+(k-r), & \text{if } q \text{ is even and } k-r<1, \\ (m+1)-(k+r), & \text{if } q \text{ is odd and } k+r<m+1, \\ (2m+1)-(k+r), & \text{if } q \text{ is odd and } k+r \geq m+1. \end{cases} \tag{23}$$

Note that here $1 \leq k \leq m$ and $0 \leq r \leq m-1$.

The described approach finds an optimal open shop schedule, since the job completion times in it are the same as in an optimal schedule for problem $P|pmtn|\sum C_j$. If $n \neq Qm$ and auxiliary jobs of maximum length have been added initially, we can assume without loss of generality that they are the last Γ jobs to finish processing, for some $\Gamma < m$. Their removal from the final schedule keeps completion times of the remaining jobs equal to their competition times in an optimal schedule to $P|pmtn|\sum C_j$.

Theorem 15 *An optimal schedule for problem $O|plbl(\underline{p})|\sum C_j$ can be constructed in $O(n \log n + mn)$ time by constructing an SPT schedule for $P||\sum C_j$, splitting it into $Q = \lceil n/m \rceil$ sections and rearranging nm operations. It can be specified by formula (23), which defines the machine number for the k-th operation of job j, and rules (a)-(d) for operation lengths, each computable in $O(1)$ time, provided the $O(n \log n)$ preprocessing related to SPT scheduling is done.*

7.4 Other minsum criteria

In this section, we give a brief overview of other traditional minsum criteria $f \in \{\sum w_j C_j, \sum U_j, \sum w_j U_j, \sum T_j, \sum w_j T_j\}$.

7.4.1 Weighted sum of completion times

By Theorem 2, problems $\alpha 2|plbl|\sum w_j C_j$, are NP-hard and problems $\alpha|plbl|\sum w_j C_j$ are strongly NP-hard for $\alpha \in \{F, O\}$.

Problem $Om|plbl|\sum w_j C_j$ can be solved in $O(mn(\sum p_j)^{m-1})$ time by adopting the algorithm due to Lawler et al. (1993) developed for $Pm||\sum w_j C_j$ and adding zero-length operations in the beginning of the schedule. Note that the same schedule is optimal for $Pm||\sum w_j C_j$ and $Pm|pmtn|\sum w_j C_j$.

For problem $Fm|plbl|\sum w_j C_j$, by Theorem 1, an optimal solution can be found in the class of F-type schedules for $Pm|pmtn|\sum w_j C_j$. If an optimal job permutation is known, then the LP formulation (4) from Prot et al. (2013) produces such a solution. Thus, the following two-step approach solves the problem.

1. Solve problem $Pm|pmtn|\sum w_j C_j$ by the $O(mn(\sum p_j)^{m-1})$ time algorithm by Lawler et al. (1993). Renumber the jobs in the order of their completion times.
2. Solve LP (4) and treat it as a solution to $Fm|plbl|\sum w_j C_j$.

Unfortunately, our methodology is not applicable to the corresponding pliability problems with a common lower bound. In the flow shop case, the disaggregation methodology of Sect. 3.2.2 cannot be adopted as it requires a common permutation for optimal schedules S^d and S^e for instances I^d and I^e. A common optimal permutation may not exist for arbitrary job weights w_j and processing times p_j.

In the case of the open shop, the approach from Sect. 7.3 cannot be generalized since an optimal schedule for problem $Pm|pmtn|\sum w_j C_j$ does not necessarily have time intervals of length $m\underline{p}$ where exactly m jobs are scheduled.

7.4.2 The number of late jobs

By Theorem 2, problems $O|plbl|\sum U_j$ and $F|plbl|\sum U_j$ are NP-hard. It is an open question whether these problems are solvable in pseudopolynomial time. Notice that this question is also open for $P|pmtn|\sum U_j$. In what follows we consider the versions of the above problems with a fixed number of machines.

Problem $Om|plbl|\sum U_j$ can be solved in $O(n^{3(m-1)})$ time by adopting the algorithm due to Lawler et al. (1993) developed for $Pm|pmtn|\sum U_j$ and adding zero-length operations in the beginning of the schedule.

For problem $Fm|plbl|\sum U_j$, by Theorem 1, an optimal solution can be found in the class of F-type schedules for $Pm|pmtn|\sum U_j$. For the latter problem, there exists an optimal schedule in which all on-time jobs are processed before the late jobs. Thus, we can use an optimal solution to $Pm|pmtn|\sum U_j$ in order to define the largest set $\mathcal{J}_1 \subseteq \mathcal{J}$ of on-time jobs. For scheduling them in the flow shop manner, introduce an auxiliary problem $Fm|plbl|L_{\max}$ defined on the set of jobs $\mathcal{J}_1$. It can be solved in $O(mn + n \log n)$ time, as discussed in Sect. 4.2. Adding the jobs $\mathcal{J} \setminus \mathcal{J}_1$ at the end of the schedule provides a solution to the original problem $Fm|plbl|\sum U_j$. Combining $O(n^{3(m-1)})$ and $O(mn + n \log n)$, we conclude that the overall time complexity is $O(n^{3(m-1)})$, assuming $m \geq 2$ and $n \geq m$.

Next we consider the pliability problem $Fm|plbl(\underline{p})|\sum U_j$. Note that the open shop problem $Om|plbl(\underline{p})|\sum U_j$ is left open since it causes difficulties similar to the easier problem $Om|plbl(\underline{p})|C_{\max}$ with the makespan objective. Our approach is based on the following two properties, which hold even for the NP-hard problem $F|plbl(\underline{p})|\sum U_j$ with an arbitrary number of machines.

Property 1 *For problem $F|plbl(\underline{p})|\sum U_j$, there exists an optimal permutation schedule with on-time jobs scheduled in non-decreasing order of due dates, followed by all late jobs.*

Property 2 *Let I be an instance of problem $F|plbl(\underline{p})|\sum U_j$ and let I^d be the diminished instance as in Definition 1. Then, given a subset $\mathcal{J}_1 \subseteq \mathcal{J}$, there exists a schedule S for instance I in which all jobs of set $\mathcal{J}_1$ are on time, if and only if there exists a schedule S^d for instance I^d in which all jobs of the set $\mathcal{J}_1$ are on time.*

Note that the first property can be proved by pairwise interchange arguments using adjacent swaps, as in Lemma 1. The second property can be proved by considering the $L_{\max}$-equivalents of the two problems in question.

Based on Properties 1, 2, we formulate the following 2-step approach.

1. Construct the diminished instance I^d for problem $Fm|plbl(\underline{p})|\sum U_j$ and find the largest set $\mathcal{J}_1 \subseteq \mathcal{J}$ of on-time jobs using the $O(n^{3(m-1)})$-time approach described in the beginning of this section.
2. Solve problem $Fm|plbl(\underline{p})|L_{\max}$ defined on the set of jobs $\mathcal{J}_1$, using the $O(mn + n\log n)$-time approach from Sect. 5.2. Adding the jobs $\mathcal{J} \setminus \mathcal{J}_1$ at the end of the schedule provides a solution to the original problem $Fm|plbl(\underline{p})|\sum U_j$.

The combined time complexity of the above two steps is $O(n^{3(m-1)})$, assuming $m \geq 2$ and $n \geq m$.

7.4.3 Weighted number of late jobs, total tardiness and weighted total tardiness

By Theorem 2, problems $\alpha 2|plbl|\sum T_j$ and $\alpha 2|plbl|\sum w_j U_j$ are NP-hard in the ordinary sense and problems $\alpha 2|plbl|\sum w_j T_j$ are strongly NP-hard for $\alpha \in \{F, O\}$.

Problem $Om|plbl|\sum w_j U_j$ can be solved in $O\left(n^{3m-5}\left(\sum w_i\right)^2\right)$ time for $m \geq 3$ and in $O\left(n^2\left(\sum w_i\right)\right)$ time for $m = 2$ by adopting the algorithms due to Lawler and Martel (1989) and Lawler et al. (1993) developed for $Pm|pmtn|\sum w_j U_j$ and adding zero-length operations in the beginning of the schedule.

For problems $Fm|plbl|\sum w_j U_j$ and $Fm|plbl(\underline{p})|\sum w_j U_j$ we can use the same idea as in the previous section: first solve the related parallel machine problem to find out an optimal set $\mathcal{J}_1$ of on-time jobs (after creating the diminished instance in the case of problem $Fm|plbl(\underline{p})|\sum w_j U_j$), then solve problem $Fm|plbl|L_{\max}$ or $Fm|plbl(\underline{p})|L_{\max}$, respectively, for the job set $\mathcal{J}_1$ to obtain a schedule in which all jobs of set $\mathcal{J}_1$ are on time. Finally, add the late jobs at the end of the schedule. Thus, problems $Fm|plbl|\sum w_j U_j$ and $Fm|plbl(\underline{p})|\sum w_j U_j$ are pseudopolynomially solvable with the same time complexity as problem $Pm|pmtn|\sum w_j U_j$.

It is an open question whether problems $O2|plbl|\sum T_j$ and $F2|plbl|\sum T_j$ are solvable in pseudopolynomial time. Notice that this question is also open for $P2|pmtn|\sum T_j$.

8 Pliability problems with $n < m$

If the number of jobs is smaller than the number of machines, $n < m$, an optimal schedule for a pliability problem with unrestricted pliability or restricted pliability with a common lower bound exhibits a more regular structure than in the case $n \geq m$. Note that for the flow shop problem Theorem 3 still holds, and we can limit our consideration to permutation schedules. If we limit our consideration to a specific permutation π, we add that restriction in the second field of the problem notation.

Theorem 16 *For problem $O|n < m,\ plbl(\underline{p})|f$, there exists an optimal schedule with*

$$C_j = p_j, \quad j \in \mathcal{J}. \tag{24}$$

For problem $F|n < m,\ plbl(\underline{p})|f$ with a fixed job permutation $\pi = (1, 2, \ldots, n)$, there exists an optimal schedule with

$$C_j = \max_{1 \leq u \leq j}\{p_u\} + (j-1)\underline{p}, \quad j \in \mathcal{J}. \tag{25}$$

For both schedules, the characteristics of each operation (machine index, operation length and its starting time) can be specified by formulae computable in $O(1)$ time. In the case of the flow shop, the formulae for starting times require $O(n)$ preprocessing related to finding C_j-values for all $j \in \mathcal{J}$ using (25).

Proof For problem $O|n < m, plbl(\underline{p})|f$, consider a schedule given by the functions $M(j,k)$, $p(j,k)$ and $S(j,k)$, which define for the k-th operation of job j the corresponding machine index, operation length and its starting time:

$$M(j,k) = \begin{cases} k+m+1-j, & \text{if } k+1 \leq j, \\ k+1-j, & \text{otherwise,} \end{cases}$$

$$p(j,k) = \begin{cases} \underline{p}, & \text{if } k \leq m-1, \\ p_j - (m-1)\underline{p}, & \text{otherwise,} \end{cases}$$

$$S(j,k) = (k-1)\underline{p}.$$

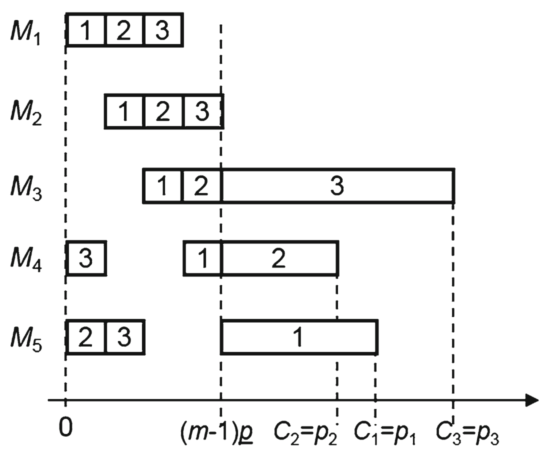

Fig. 9 An optimal schedule for $O|n<m, plbl(\underline{p})|f$

The schedule is illustrated in Fig. 9. The interval $\left[0, (m-1)\underline{p}\right]$ is used for operations of length $\underline{p}$ sequenced in a wrap-around manner; the last m operations on m downstream machines handle the remaining processing for all jobs, one job per machine. The completion times C_j satisfy (24) and they cannot be reduced; thus, the resulting schedule is optimal.

For problem $F|n<m, plbl(\underline{p})|f$ and a fixed job permutation $\pi=(1,2,\ldots,n)$, consider a permutation schedule constructed in the following way. Split each job j into $m-j$ initial operations of length $\underline{p}$, followed by operation $m-j+1$ of length $p_j-(m-1)\underline{p}$, followed by $j-1$ tail operations, again of length $\underline{p}$. For each job j the initial operations are scheduled in a staircase manner, starting at time $(j-1)\underline{p}$. Operation $m-j+1$ is started at time $(m-1)\underline{p}$ for each job j. Finally, schedule the tail operations of length $\underline{p}$, again in a staircase manner as early as possible, without violating any (permutation) flow shop constraints. The schedule is illustrated in Fig. 10.

The job completion times in the constructed schedule satisfy:

$$C_1 = p_1,$$
$$C_j = \max\left\{C_{j-1}+\underline{p},\, p_j+(j-1)\underline{p}\right\},\quad j=2,\ldots,n,$$

Here, $C_j = C_{j-1}+\underline{p}$ corresponds to the case when the last operation of j, which is of minimum length, starts on the last machine immediately after job $j-1$ is completed, and $C_j = p_j+(j-1)\underline{p}$ corresponds to the case when job j is processed contiguously after the minimum quantities of the preceding $j-1$ jobs are processed on M_1. Since in either case C_j matches the lowest achievable value, the resulting schedule is optimal. Notice that the above formulae imply (25).

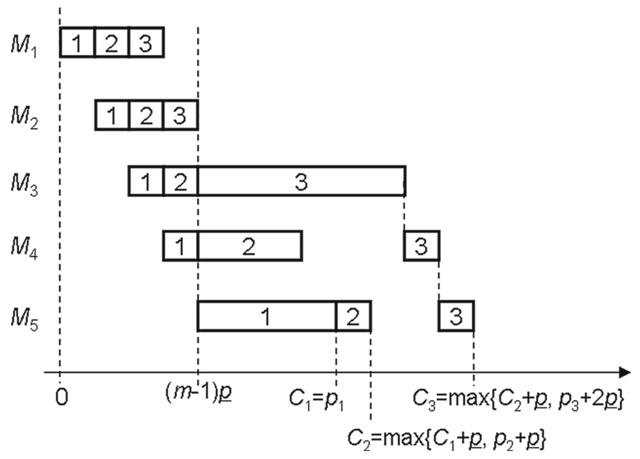
M_1
1 2 3
M_2
1 2 3
M_3
1 2
3
M_4
1 2
3
M_5
1 2
3
0
$(m\text{-}1)\underline{p}$
$C_1=p_1$
$C_3=\max\{C_2+\underline{p},\, p_3+2\underline{p}\}$
$C_2=\max\{C_1+\underline{p},\, p_2+\underline{p}\}$

Fig. 10 An optimal schedule for $F|n<m, plbl(\underline{p})|f$ and a fixed job permutation $\pi=(1,2,\ldots,n)$

For a compact encoding, we specify an optimal schedule via functions $p(j,i)$ and $S(j,i)$, which compute the processing times and starting times for operations O_{ij}:

$$p(j,i)=\begin{cases}\underline{p}, & \text{if } i+j\neq m+1,\\ p_j-(m-1)\underline{p}, & \text{otherwise,}\end{cases}$$
$$S(j,i)=\begin{cases}(i-1)\underline{p}, & \text{if } i+j\leq m+1,\\ C_j-(m+1-i)\underline{p}, & \text{otherwise.}\end{cases}$$

Here, C_j is given by (25). Note that in the flow shop, the i-th operation of a job j is operation O_{ij} processed by machine M_i. □

We conclude that for problem $O|n<m, plbl(\underline{p})|f$, a single schedule is optimal for any non-decreasing objective f, while solving problem $F|n<m, plbl(\underline{p})|f$ reduces to finding an optimal job permutation minimizing $f(C_1,\ldots,C_n)$ with C_j given by (25). Notice that formula (25) is similar to the formula for C_j known for the proportionate flow shop $Fm|p_{ij}=\tau_j|f(C_1,\ldots,C_n)$ (Pinedo 2016):

$$C_j=(m-1)\max_{1\leq u\leq j}\{\tau_u\}+\sum_{u=1}^{j}\tau_u,\quad j\in\mathcal{J}. \tag{26}$$

In the latter problem, every job $j\in\mathcal{J}$ consists of m operations of equal length τ_j, so that the total length of job j is $p_j=m\tau_j$. Not surprisingly, the sequencing problems with completion times given by (25) and (26) are similar to their single-machine counterparts:

- for $f=C_{\max}$, any permutation provides the same value; namely $C_{\max}=\max\{p_j|j\in\mathcal{J}\}+(n-1)\underline{p}$ for the pliability problem;

- for $f = \sum C_j$ the jobs can be sequenced in SPT order, so that for the pliability problem $C_j = p_j + (j-1)\underline{p}$ for any $j \in \mathcal{J}$;
- for $f = L_{\max}$, the jobs can be sequenced in EDD order; for the pliability problem this can be proved using adjacent swaps introduced in Lemma 1;
- for $f = \sum U_j$, Moore's algorithm (1968) can be adjusted accordingly, without increasing its running time;
- for $f = \sum T_j$, the pseudopolynomial-time algorithm for $1||\sum T_j$ can be adjusted as well, also without increasing its running time.

Not all results known for the single-machine problem are transferable though. As in the case of the proportionate flow shop, the problem with $f = \sum w_j C_j$ can be solved in $O(n^2)$ time by a special 'minimum cost insertion' (MCI) algorithm. Without presenting full proof details, which follow from Shakhlevich et al. (1998), we formulate some properties of an optimal schedule and outline the algorithm.

Property 3 *If in an instance of problem $F|n < m, plbl(\underline{p})|\sum w_j C_j$ two jobs h and j satisfy conditions*

$$p_h \le p_j \text{ and } w_h \ge w_j,$$

then there exists an optimal schedule with h scheduled prior to j.

Given a schedule, we define concepts of a new-max job and a segment. A *new-max job* is a job with a processing time which exceeds the processing times of all its predecessors in the schedule. A *segment* starts with a new-max job and it includes all subsequent jobs, with processing times no larger than that of the new-max job.

Property 4 *An optimal schedule for problem $F|n < m, plbl(\underline{p})|\sum w_j C_j$ consists of segments, in which all jobs are sequenced in non-increasing order of their w_j-values.*

The algorithm, which follows the ideas from Shakhlevich et al. (1998), is as follows.

Algorithm MCI

```
1. Renumber the jobs so that $w_1 \ge w_2 \ge \cdots \ge w_n$. Break
   ties in favor of a job with a smaller processing time.
2. Set the initial sequence consisting of job 1 as $\sigma_1 = (1)$.
3. For $j = 2$ to $n$
    (a) Produce sequence $\sigma_j$ consisting of jobs $1, 2, \ldots, j$ as
        the best outcome of inserting $j$ into $\sigma_{j-1}$. If there are
        several insertion possibilities with the same outcome,
        choose the one with the latest insertion position of $j$.
```

Note that the "best outcome" can be obtained by trying all possible insertions of job j and selecting the one which delivers the minimum value of the objective. Due to the job numbering and by Property 3, we do not need to consider insertion positions for j in any part of the schedule that precedes a smaller job. By the job numbering and Properties 4 and 3, we only need to consider insertion positions for j immediately before the next new-max job (if j is not a new-max job) and at the end of the schedule (if j is a new-max job). Therefore, for each insertion, the change in the objective can be calculated in $O(1)$ time. Since there are at most n insertion points in σ_{j-1} and $n-1$ repetitions of Step 3, the overall time complexity is $O(n^2)$.

9 Conclusions

In this paper, we studied general properties of pliability models and performed a thorough complexity classification of flow shop and open shop problems with pliable jobs.

Comparing open shop and flow shop models, we cannot draw a single conclusion: in many cases, the two counterparts have the same time complexity. For the unrestricted model with $f = L_{\max}$, we have a faster algorithm for the open shop problem; for the models with a common lower bound and $f \in \{C_{\max}, L_{\max}\}$, the flow shop problems are polynomially solvable, but the complexity status of the open shop counterparts is left as an open question. For the restricted case and $f = C_{\max}$, the flow shop problem with two machines is NP-hard, while the open shop problem is solvable in $O(n)$ time.

Under the assumption $n < m$, several type (i) and type (ii) problems become tractable. For the open shop problem of type (i) and (ii), a common schedule is optimal for any regular criterion. The same is true for the flow shop problem of type (i). The flow shop problem of type (ii) reduces to finding an optimal job permutation and becomes similar to its single-machine counterpart. Whenever a job permutation is found, the characteristics of an optimal schedule can be found by using $O(1)$ formulae.

In the situation $n \ge m$, the problems of type (i) and (ii) appear to be no harder than their flow shop and open shop counterparts, with or without preemption, and are often solvable by faster algorithms than the traditional problems. Exceptions are problems $O|plbl(\underline{p})|C_{\max}$ and $O|plbl(\underline{p})|L_{\max}$: the complexity status of both problems remains open. On the other hand, problems of type (iii) are no easier than their traditional counterparts. In particular, for $F2|\circ|C_{\max}$, the pliability version is NP-hard, while its classical counterpart is solvable in $O(n \log n)$ time (Johnson 1954).

Having studied basic models with pliability, we propose the following directions for further research. Type (ii) and type (iii) models handle scenarios with restrictions on operation lengths: there can be a common lower bound $\underline{p}$ for all jobs, or the lower bounds $\underline{p}_{ij}$ can be individual for all job-machine pairs. The intermediate cases, lying in-between type (ii) and type (iii) models, are job-dependent lower bounds $\underline{p}_{ij} = \underline{p}_j$ or machine-dependent lower bounds $\underline{p}_{ij} = \underline{p}_i$. Our study already provides some initial results, in particular those presented in Sects. 5.1, 6.1, 6.2: the approach from Sect. 5.1 can be generalized for solving $F2|plbl(\underline{p}_{ij} = \underline{p}_i)|C_{\max}$, the NP-hardness proof from Sect. 6.1 is applicable for problem $F2|plbl(\underline{p}_{ij} = \underline{p}_j)|C_{\max}$, and the algorithm from Sect. 6.2 solves efficiently the most general open shop problem of type (iii). Other versions require further analysis.

Another type of pliability can be defined in terms of the deviation from "ideal" operation lengths p^0_{ij}. In such models actual processing times p_{ij} have to be selected from intervals $\left[p^0_{ij} - \Delta,\ p^0_{ij} + \Delta\right]$ with some given parameter Δ. Whenever an actual processing time exceeds its ideal value, $p_{ij} > p^0_{ij}$, a cost may be incurred associated with additional power or other resources for extra-work, allocated to the machine above the expected "ideal" load. Alternatively, performing a part of an operation on a "wrong" machine may increase the processing time of that part, since a "wrong" machine may operate at a slower rate processing the relocated operation part. The proposed model has similarities to models with controllable processing times where operation lengths can be reduced via the usage of additional resources. It will be interesting to explore links between the proposed "Δ-redistribution" model and the stream of research related to controllable processing times.

Acknowledgements The authors thank the referee for suggestions aimed at improving the paper. This work was supported by the Deutsche Forschungsgemeinschaft, KN 512/7-1. The work of Stefan Waldherr was supported by the TUM Institute for Advanced Study through a Hans Fischer Senior Fellowship. The work of N.V. Shakhlevich was supported by the EPSRC Grant EP/K041274/1.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

Appendix 1: Proof of Lemma 1

Consider problem $F|plbl(\underline{p})|f$ and the permutation schedule S where job u is sequenced immediately before job v in the permutation, as described in the formulation of Lemma 1. To prove the lemma, we first formulate assumptions (A1)-(A2) about schedule S, then derive properties (P1)-(P7) of S, and finally construct the required schedule S', the feasibility of which is justified by stating conditions (C1)-(C6), characterizing the constructed schedule.

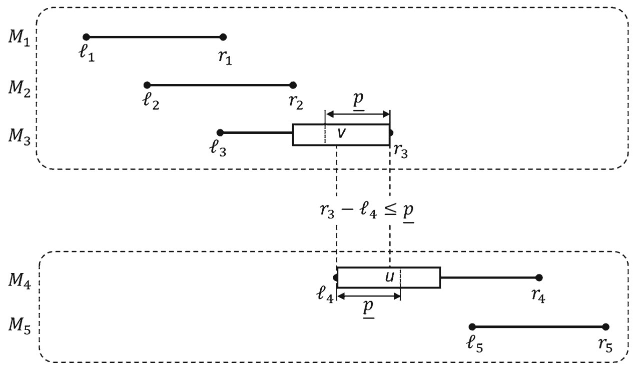

Fig. 11 Splitting into two subinstances if $r_i - \ell_{i+1} \le \underline{p}$ ($i = 3$), with unique allocation of the v-operation to M_i and u-operation to M_{i+1}; allocation of remaining operations is not shown

Schedule S: Assumptions
Given schedule S, define for each machine M_i, $1 \le i \le m$, the time window $[\ell_i, r_i]$ where jobs u and v are processed: ℓ_i is the starting time of the u-operation on machine M_i and r_i is the completion time of the v-operation on that machine. Since all operations have length no less than $\underline{p}$, the time windows follow a staircase pattern: for any pair of machines M_i, M_{i+1}, $1 \le i \le m-1$,

$$\begin{aligned} \ell_i + \underline{p} &\le \ell_{i+1}, \\ r_i + \underline{p} &\le r_{i+1}. \end{aligned} \tag{A1}$$

Furthermore, if for a pair of machines M_i, M_{i+1} the intersection of $[\ell_i, r_i]$ and $\left[\ell_{i+1}, r_{i+1}\right]$ is empty, then an instance of problem $F|plbl(\underline{p})|f$ can be split into two independent subinstances defined by machine sets $\{M_1, \ldots, M_i\}$ and $\{M_{i+1}, \ldots, M_m\}$. A splitting can also be done if the intersection is non-empty, but sufficiently small, with a common subinterval $\left[\ell_{i+1}, r_i\right]$ of length

$$r_i - \ell_{i+1} \le \underline{p},$$

see Fig. 11 for an illustration with $i = 3$. Notice that in the intersection interval machine M_i can only process the compulsory part of the v-operation, while machine M_{i+1} can only process the compulsory part of the u-operation, partially if the interval length is less than $\underline{p}$. For the remaining pairs of machines M_x, M_y with $x \le i$, $y \ge i+1$, where at least one of the inequalities is strict, the time windows $[\ell_x, r_x]$ and $\left[\ell_y, r_y\right]$ do not intersect due to (A1).

In the remainder of the proof, we assume that any splittable instance is replaced by independent subinstances, with each subinstance satisfying

$$r_i - \ell_{i+1} > \underline{p} \tag{A2}$$

for every pair of machines M_i, M_{i+1}, $1 \le i \le m-1$.

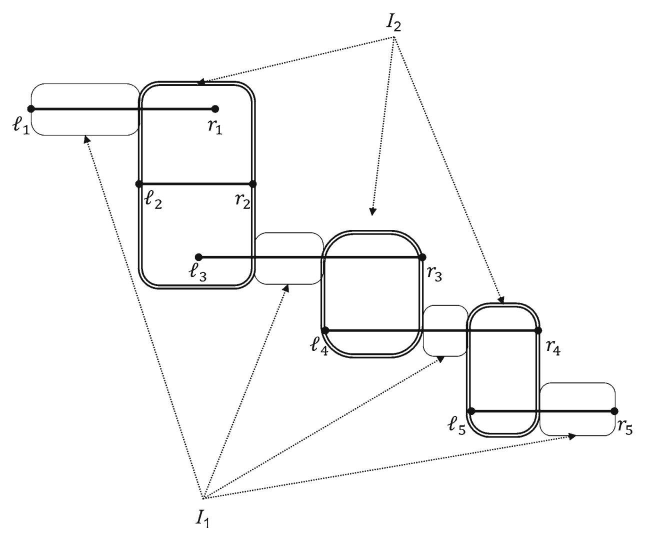

Fig. 12 Two types of intervals I_1 and I_2 for schedule S

Schedule S: Properties
For the combined interval $[\ell_1, r_m]$, define subintervals of types I_1 and I_2 in accordance with the number of available machines: a subinterval is of type I_1, if only one machine is available for processing $\{u, v\}$, and it is of type I_2, if there are at least two machines available. An illustrative example is presented in Fig. 12. Denote the total lengths of I_1- and I_2-intervals by T_1 and T_2, respectively, with $T_1 + T_2 = r_m - \ell_1$.

In what follows we formulate properties of I_1- and I_2-intervals for schedule S.

(P1) On machine M_1, an interval of type I_1 is followed by an interval of type I_2.
(P2) On every machine M_i, $2 \le i \le m-1$, there is either one interval of type I_2 or there are three intervals of types I_2, I_1, I_2, which appear in this order.
(P3) On machine M_m, an interval of type I_2 is followed by an interval of type I_1.
(P4) $T_1 \ge 2\underline{p}$,
(P5) $r_i - \ell_i \ge 2\underline{p}, \qquad 1 \le i \le m$,
(P6) $\max\{p_u, p_v\} \le T_1 + T_2 - \underline{p}$,
(P7) $p_u + p_v \le T_1 + 2T_2$.

Properties (P1)-(P3) hold due to (A1), (A2).

Property (P4) holds since I_1 contains at least two intervals: one on M_1 and one on M_m (by (P1), (P3)), and the length of each such interval is at least $\underline{p}$ (by (A1)).

Property (P5) holds since during time interval $[\ell_i, r_i]$, machine M_i processes one operation of job u and one operation of job v, each of length no less than $\underline{p}$.

If property (P6) does not hold, then scheduling the longest job does not leave room of length $\underline{p}$ for the compulsory part of the other job on M_m (if job u is the longest) or on M_1 (if job v is the longest). In that case, no feasible permutation schedule exists with job u and job v scheduled in intervals $[\ell_i, r_i]$, $1 \le i \le m$, in any order; a contradiction to the feasibility of S.

Finally, if property (P7) does not hold, then no feasible schedule exists even for the relaxed problem with m identical parallel machines processing two jobs $\{u, v\}$ of lengths p_u and p_v with preemption in intervals $[\ell_i, r_i]$, $1 \le i \le m$; again, a contradiction to the feasibility of S. Here we take into account that only intervals of type I_2 are suitable for processing two jobs simultaneously.

Schedule S′: Construction
Given time windows $[\ell_i, r_i]$, $1 \le i \le m$, satisfying (A1), (A2) and (P1)–(P7), we construct schedule S' by allocating operations of jobs v and u in these time windows, with v preceding u. Our task is to find operation lengths p_{iv} and p_{iu} for these two jobs on all machines M_i, $1 \le i \le m$.

Without loss of generality we assume that

$$\min\{p_u, p_v\} \ge T_2 + \underline{p}; \tag{27}$$

otherwise adjust (temporarily) the processing times of jobs u and v to $\max\left\{p_u, T_2 + \underline{p}\right\}$ and $\max\left\{p_v, T_2 + \underline{p}\right\}$, respectively; the extra amount of processing will be removed from a feasible schedule, after it is constructed. This condition simplifies the construction of the new schedule, and the adjustment does not affect (P1)–(P5) and keeps (P6), (P7) satisfied. Indeed, property (P6) for the adjusted processing times is of the form $\max\left\{p_u, p_v, T_2 + \underline{p}\right\} \le T_1 + T_2 - \underline{p}$. It holds since the original property (P6) is satisfied for p_u, p_v and by (P4). For property (P7), the sum of the adjusted processing times in the left hand side is equal to one of the following values:

$$\begin{array}{ll}
p_u + p_v, & \text{if } \; p_u > T_2 + \underline{p},\ p_v > T_2 + \underline{p}; \\
p_u + \left(T_2 + \underline{p}\right), & \text{if } \; p_u > T_2 + \underline{p},\ p_v \le T_2 + \underline{p}; \\
\left(T_2 + \underline{p}\right) + p_v, & \text{if } \; p_u \le T_2 + \underline{p},\ p_v > T_2 + \underline{p}; \\
2\left(T_2 + \underline{p}\right), & \text{if } \; p_u > T_2 + \underline{p},\ p_v > T_2 + \underline{p}.
\end{array}$$

The first expression is bounded by $T_1 + 2T_2$ if (P7) holds for original p_u, p_v; the second and the third expressions are bounded by $\left(T_1 + T_2 - \underline{p}\right) + \left(T_2 + \underline{p}\right)$ by (P6); the last expression is bounded by $2T_2 + T_1$ by (P4).

The algorithm for scheduling jobs u and v satisfying (27) consists of two stages.

Stage 1 Allocate the processing amount $T_2 + \underline{p}$ of job v, using the following intervals:

– $\left[\ell_1, \ell_1 + \underline{p}\right]$ on M_1,

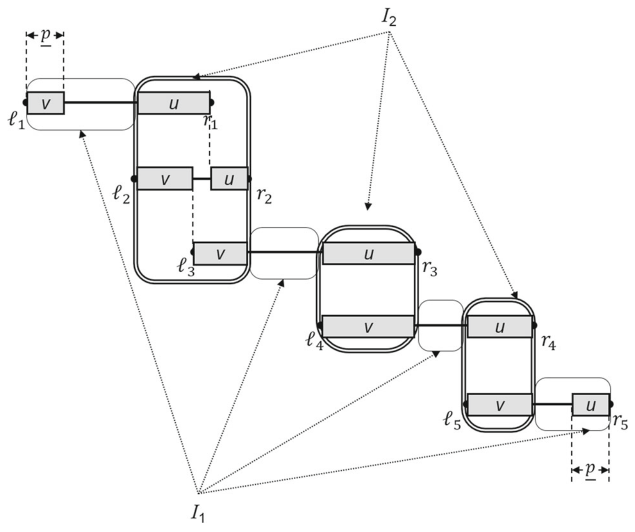

Fig. 13 The outcome of Stage 1

– $\left[\ell_i, \min\{r_{i-1}, \ell_{i+1}\}\right]$ on each M_i, $2 \le i \le m-1$,
– $\left[\ell_m, r_{m-1}\right]$ on M_m.

Allocate the processing amount $T_2 + \underline{p}$ of job u, using the following intervals:

– $\left[r_m - \underline{p}, r_m\right]$ on M_m,
– $\left[\max\{r_{i-1}, \ell_{i+1}\}, r_i\right]$ on each M_i, $2 \le i \le m-1$,
– $[\ell_2, r_1]$ on M_1.

The schedule found as a result of Stage 1 is shown in Fig. 13.

Stage 2 If $p_u = p_v = T_2 + \underline{p}$, then no further action is needed. Otherwise use intervals of type I_1, which are left free after Stage 1, for allocating the remaining quantities of jobs u and v, splitting each I_1-interval into three parts, some of which may be of zero length: one part for job v, another one for an idle interval and the third part for job u. For each machine, the splitting can be performed arbitrarily, but the resulting cumulative length of all operations of jobs u and v has to constitute p_u and p_v. Such a splitting can always be found, justified as condition (C6) below.

Note that if an interval of type I_2 has more than two available machines, then two of them receive jobs u and v for processing, while the remaining available machines are left idle (for example, an idle interval on machine M_2 in Fig. 13).

In order to justify the correctness of the algorithm, we demonstrate that the following conditions hold for the resulting schedule:

(C1) every machine M_i, $1 \le i \le m$, processes job v first and job u next;
(C2) for each job v and u, the minimum processing amount $\underline{p}$ is assigned to each machine M_i, $1 \le i \le m$;
(C3) the time intervals with v-operations on two consecutive machines do not overlap; the same is true for u-operations;
(C4) the time intervals with v- and u-operations on one machine do not overlap;
(C5) at the end of Stage 1 the intervals of combined capacity of $T_2 + 2\underline{p}$ are used to process the amount $2T_2 + 2\underline{p}$ of jobs v and u;
(C6) at the end of Stage 2 processing quantities p_v, p_u are allocated in full.

First we prove conditions (C1)–(C3) for job v; similar arguments are applicable for job u.

Condition (C1) holds since job v starts at time ℓ_i on every machine M_i, according to allocation of Stage 1; further allocation of the same job on M_i performed at Stage 2 is done in the I_1-interval, if one exists; such an interval follows the previously used I_2 interval by property (P2).

Condition (C2) holds for job v since the length of the interval $\left[\ell_i, \min\{r_{i-1}, \ell_{i+1}\}\right]$ is no smaller than $\underline{p}$: $\ell_{i+1} - \ell_i \ge \underline{p}$ by (A1) and $r_{i-1} - \ell_i > \underline{p}$ by (A2).

Condition (C3) holds for job v since after Stage 1 is completed, the allocated intervals $\left[\ell_i, \min\{r_{i-1}, \ell_{i+1}\}\right]$ and $\left[\ell_{i+1}, \min\{r_i, \ell_{i+2}\}\right]$ on machines M_i, M_{i+1} do not overlap for any i, $1 \le i \le m-1$. If at Stage 2 an interval of type I_1 is used for allocating job v on machine M_i, $1 \le i \le m$, no overlapping can happen as any I_1-interval is associated with only one available machine.

We now prove the remaining conditions.

Condition (C4) follows from the splitting strategy of Stage 2.

Condition (C5) holds since at the end of Stage 1 operations of job v fully occupy all intervals of type I_2 together with $\left[\ell_1, \ell_1 + \underline{p}\right]$, and operations of job u fully occupy all intervals of type I_2 together with $\left[r_m - \underline{p}, r_m\right]$.

In order to prove condition (C6), first recall that due to (27) we have $p_u, p_v \ge T_2 + \underline{p}$. The parts of the processing amounts of jobs u and v that still need to be allocated after Stage 1, are $p_u - \left(T_2 + \underline{p}\right)$ and $p_v - \left(T_2 + \underline{p}\right)$, respectively. The remaining capacity of I_1-intervals is $\left(T_1 - 2\underline{p}\right)$. Then condition (C6) follows from property (P7) rewritten as

$$p_u + p_v - 2\left(T_2 + \underline{p}\right) \le T_1 - 2\underline{p}.$$

Finally, in the case that one of the original processing times is lower than $T_2 + \underline{p}$, the extra amount can be arbitrarily removed from intervals of type I_2 as long as the processing times on all machines remain at least $\underline{p}$.

Thus, the described algorithm constructs the required schedule S'. □

Appendix 2: An instance of problem $F|plbl|L_{\max}$ with $\Omega(nm)$ nonzero operations in an optimal solution

The following example shows that the term mn in the complexity estimate $O(n\log n + mn)$ for problem $F|plbl|L_{\max}$ in Theorem 7 cannot be eliminated. Consider an instance with m machines and n jobs under the assumption that

$$m < \frac{1}{3}n. \tag{28}$$

The job set $\mathcal{J}$ consists of three types of jobs, $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$, $\mathcal{H} = \{h_1, h_2, \ldots, h_{n-2m}\}$ and $\mathcal{V} = \{v_1, v_2, \ldots, v_m\}$, with the following characteristics:

$$\begin{array}{lll} p_{u_j} = j, & d_{u_j} = j, & u_j \in \mathcal{U}, \\ p_{h_j} = m, & d_{h_j} = m + j, & h_j \in \mathcal{H}, \\ p_{v_j} = j, & d_{v_j} = n - m + 1, & v_j \in \mathcal{V}. \end{array}$$

Clearly,

$$\overline{d} = n - m + 1$$

is the maximum due date in the instance. We demonstrate that in an optimal solution, there is a unique way for allocating the jobs from $\mathcal{U} \cup \mathcal{H}$ such that every job u_j is split into j unit-length operations and every job h_j is split into m unit-length operations, see Fig. 14 for an illustration. This implies that the total number of nonzero operations is at least

$$Q = \frac{1}{2}m(m+1) + (n-2m)m > m\left(n - \frac{3}{2}m\right) > \frac{1}{2}mn = \Omega(mn), \tag{29}$$

where the last inequality holds by (28). Deriving the estimate Q, we do not count the number of nonzero operations associated with the jobs $\mathcal{V}$ as there may be multiple ways for their allocation.

First notice that for the relaxed problem $P|pmtn|L_{\max}$, the optimal objective value is $L_{\max} = 0$, which can be calculated using the closed form expression from Baptiste (2000). Moreover, the total processing time of all n jobs is equal to the total capacity $\overline{d}m$ of all machines in the interval $\left[0, \overline{d}\right]$. Thus, for any schedule with $L_{\max} = 0$, every machine M_i operates without idle times in $\left[0, \overline{d}\right]$.

Without focusing on the allocation of operations of jobs $\mathcal{V}$ on machine M_m, consider the allocation of the jobs $\mathcal{U} \cup \mathcal{H}$ on that machine. Job u_1 has to be fully processed in time interval $[0, 1]$ completing at time 1 on M_m. If the operation of u_1 on M_m is of length $p_{m,u_1} < 1$, then there is an idle time on M_m, since in an optimal schedule with $L_{\max} = 0$ no other job can have its final operation before time 1. Thus, $p_{m,u_1} = 1$.

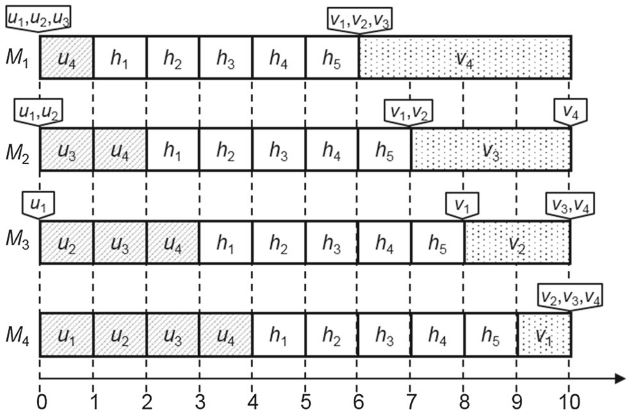

Fig. 14 An optimal solution to an instance of $F|plbl|L_{\max}$ with $\Omega(mn)$ nonzero operations

Job u_2 has to be fully processed in time interval $[0, 2]$ completing at time 2 on M_m. Again, the operation length of u_2 on M_m has to be $p_{m,u_2} = 1$ in order to avoid an idle time on that machine.

Continuing this line of arguments, it is easy to prove by induction that machine M_m processes unit-length operations of jobs $\mathcal{U} \cup \mathcal{H}$ in the order of their numbering, first $\mathcal{U}$ and then $\mathcal{H}$. Such an allocation on M_m induces the deadlines for completing jobs $\mathcal{U} \cup \mathcal{H}$ on machine M_{m-1}: $d_{u_j} - 1$ for the $\mathcal{U}$-jobs and $d_{h_j} - 1$ for the $\mathcal{H}$-jobs. In this way, we obtain a smaller instance with machines $\{M_1, \ldots, M_{m-1}\}$ and adjusted job characteristics: for every job from $\mathcal{U} \cup \mathcal{H}$ the d- and p-values are reduced by 1, while for the $\mathcal{V}$-jobs, their total processing time is reduced by 1. Since this subinstance is similar to the initial one, the above arguments are applicable to prove that machine M_{m-1} processes a zero-length operation of u_1 first, followed by the unit-length operations of jobs $\mathcal{U} \backslash \{u_1\} \cup \mathcal{H}$ in the order of their numbering.

Proceeding similarly, we conclude that there is a unique way of allocating jobs $\mathcal{U} \cup \mathcal{H}$ in an optimal F-type schedule, and it leaves on every machine M_i, $1 \le i \le m$, time windows $\left[\overline{d} - m + i - 1, \overline{d}\right]$ for processing jobs $\mathcal{V}$. One possible allocation of the $\mathcal{V}$-jobs is presented in Fig. 14 , where each job v_i is processed in full on one of the machines, with zero-length operations on the remaining machines. Thus, estimate (29) holds and the constructed instance has $\Omega(mn)$ nonzero operations.

References

Anuar, R., & Bukchin, Y. (2006). Design and operation of dynamic assembly lines using work-sharing. *International Journal of Production Research*, *44*, 4043–4065.

Askin, R. G., & Chen, J. (2006). Dynamic task assignment for throughput maximization with worksharing. *European Journal of Operational Research*, *168*, 853–869.

Balas, E., & Zemel, E. (1980). An algorithm for large zero-one knapsack problems. *Operations Research*, *28*, 1130–1154.

Baptiste, P. (2000). Preemptive scheduling of identical machines. UTC research report 2000/314, Univ. de Tech. de Compiègne, F-60200 Compiègne, France.

Baranski, T. (2011). Task scheduling with restricted preemptions. In *Proceedings of the federated conference on computer science and information systems* (pp. 231–238).

Brauner, N., Crama, Y., Grigoriev, A., & van de Klundert, J. (2005). A framework for the complexity of high-multiplicity scheduling problems. *Journal of Combinatorial Optimization*, *9*, 313–323.

Bruno, J., & Gonzalez, T. (1976). Scheduling independent tasks with release dates and due dates on parallel machines. Technical Report 213, Pennsylvania State University.

Brucker, P. (2007). *Scheduling algorithms* (5th ed.). Heidelberg: Springer.

Bultmann, M., Knust, S., & Waldherr, S. (2018). Flow shop scheduling with flexible processing times. *OR Spectrum*, *40*, 809–829.

Bultmann, M., Knust, S., & Waldherr, S. (2018). Synchronous flow shop scheduling with pliable jobs. *European Journal of Operational Research*, *270*, 943–956.

Burdett, R. L., & Kozan, E. (2001). Sequencing and scheduling in flowshops with task redistribution. *Journal of the Operational Research Society*, *52*, 1379–1389.

Chang, J. H., & Chiu, H. N. (2005). A comprehensive review of lot streaming. *International Journal of Production Research*, *43*, 1515–1536.

Conway, R. W., Maxwell, W. L., & Miller, L. W. (1967). *Theory of scheduling* (p. 1967). Reading, MA: Addison-Welsey.

Crama, Y., & Gultekin, H. (2010). Throughput optimization in two-machine flowshops with flexible operations. *Journal of Scheduling*, *13*, 227–243.

Ecker, K., & Hirschberg, R. (1993). Task scheduling with restricted preemptions. *Lecture Notes in Computer Science*, *694*, 464–475.

Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of flowshop and job shop scheduling. *Mathematics of Operations Research*, *1*, 117–129.

Gonzalez, T., & Sahni, S. (1976). Open shop scheduling to minimize finish time. *Journal of the ACM*, *23*, 665–679.

Gupta, J. N. D., Koulamas, C. P., Kyparisis, G. J., Potts, C. N., & Strusevich, V. A. (2004). Scheduling three-operation jobs in a two-machine flow shop to minimize makespan. *Annals of Operations Research*, *129*, 171–185.

Gultekin, H. (2012). Scheduling in flow shops with flexible operations: Throughput optimization and benefits of flexibility. *International Journal of Production Economics*, *140*, 900–911.

Hefetz, N., & Adiri, I. (1982). A note on the influence of missing operations on scheduling problems. *Naval Research Logistics Quarterly*, *29*, 535–539.

Johnson, S. M. (1954). Optimal two-and-three-stage production schedules with set-up times included. *Naval Research Logistics Quarterly*, *1*, 61–68.

Koulamas, C., & Kyparisis, G. J. (2015). The three-machine proportionate open shop and mixed shop minimum makespan problems. *European Journal of Operational Research*, *243*(1), 70–74.

Kropp, D. H., & Smunt, T. L. (1990). Optimal and heuristic models for lot splitting in a flow shop. *Decision Sciences*, *21*, 691–709.

Labetoulle, J., Lawler, E. L., Lenstra, J. K., & Rinnooy Kan, A. H. G. (1984). Preemptive scheduling of uniform machines subject to release dates. In H. R. Pulleybank (Ed.), *Progress in combinatorial optimization* (pp. 245–261). New York: Academic Press.

Lawler, E. L. (1983). Recent results in the theory of machine scheduling. In A. Bachem, M. Grötschel, & B. Korte (Eds.), *Mathematical programming the state of the art* (pp. 202–234). Berlin: Springer.

Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., & Shmoys, D. B. (1993) Sequencing and scheduling: algorithms and complexity. Handbook in Operations Research and Management Science (Vol. 4, 445–522). Amsterdam.

Lawler, E. L., & Martel, C. U. (1989). Preemptive scheduling of two uniform machines to minimize the number of late jobs. *Operations Research*, *37*, 314–318.

Lin, B. M. T., Hwang, F. J., & Gupta, J. N. D. (2016). Two-machine flowshop scheduling with three-operation jobs subject to a fixed job sequence. *Journal of Scheduling*, *20*, 293–302.

McLain, J. O., Thomas, L. J., & Sox, C. (1992). "On-the-fly" line balancing with very little WIP. *International Journal of Production Economics*, *27*, 283–289.

McNaughton, R. (1959). Scheduling with deadlines and loss functions. *Management Science*, *12*, 1–12.

Moore, J. M. (1968). An n job, one machine sequencing algorithm for minimizing the number of late jobs. *Management Science*, *15*, 102–109.

Ostolaza, J., McLain, J. O., & Sox, C. (1990). The use of dynamic (state-dependent) assembly-line balancing to improve throughput. *Journal of Manufacturing and Operations Management*, *3*, 105–133.

Pienkosz, K., & Prus, A. (2015). Task scheduling with restricted preemptions on two parallel processors. *International Conference on Methods and Models in Automation and Robotics*, *2015*, 58–61.

Pinedo, M. (2016). *Scheduling: Theory, algorithms, and systems* (6th ed.). Berlin: Springer.

Potts, C. N., Shmoys, D. B., & Williamson, D. P. (1991). Permutation vs. non-permutation flow shop schedules. *Operations Research Letters*, *10*, 281–284.

Prot, D., Bellenguez-Morineau, O., & Lahlou, C. (2013). New complexity results for parallel identical machine scheduling. *European Journal of Operational Research*, *231*, 282–287.

Ruiz-Torres, A. J., Ablanedo-Rosas, J. H., & Ho, J. C. (2010). Minimizing the number of tardy jobs in the flow shop problem with operation and resource flexibility. *Computers and Operations Research*, *37*, 291–292.

Ruiz-Torres, A. J., Ho, J. C., & Ablanedo-Rosas, J. H. (2011). Makespan and workstation utilization minimization in a flowshop with operations flexibility. *Omega*, *39*, 273–282.

Sahni, S. (1979). Preemptive scheduling with due dates. *Operations Research*, *27*, 925–934.

Serafini, P. (1996). Scheduling jobs on several machines with the job splitting property. *Operations Research*, *44*, 617–628.

Shakhlevich, N. V., Hoogeveen, H., & Pinedo, M. (1998). Minimizing total weighted completion time in a proportionate flow shop. *Journal of Scheduling*, *1*, 157–168.

Trietsch, D., & Baker, K. R. (1993). Basic techniques for lot streaming. *Operations Research*, *41*, 1065–1076.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.