CEJOR (2020) 28:5–23
https://doi.org/10.1007/s10100-018-0553-8

ORIGINAL PAPER

Mixed integer linear programming models for Flow Shop Scheduling with a demand plan of job types

Joaquín Bautista-Valhondo[1] · Rocío Alfaro-Pozo[2]

Published online: 29 May 2018

Abstract This paper presents two mixed integer linear programming (MILP) models that extend two basic Flow Shop Scheduling problems: Fm/prmu/C_{max} and Fm/block/C_{max}. This extension incorporates the concept of an overall demand plan for types of jobs or products. After using an example to illustrate the new problems under study, we evaluated the new models and analyzed their behaviors when applied to instances found in the literature and industrial instances of a case study from Nissan's plant in Barcelona. CPLEX solver was used as a solution tool and obtained acceptable results, allowing us to conclude that MILP can be used as a method for solving Flow Shop Scheduling problems with an overall demand plan.

Keywords Flow Shop Scheduling problem · Mixed model sequencing problem · Mixed model assembly lines · Overall demand plan · Mixed integer linear programming

1 Preliminaries

The *Flow Shop Scheduling problem* (FSP) is a sequencing problem that has received considerable attention from professionals and researchers in recent decades due in part to the wide range of production environments it can model (Pinedo 2016).

✉ Joaquín Bautista-Valhondo
joaquin.bautista@upc.edu

Rocío Alfaro-Pozo
ralfaro@eae.es

1 IOC-ETSEIB, Universitat Politècnica de Catalunya, 08028 Barcelona, Spain

2 EAE Business School, 08015 Barcelona, Spain

In an FSP, a set of jobs or products J (D elements) needs to be processed in a group of machines K (m elements) arranged in series. All jobs must proceed through all machines in the same order, starting with machine 1 and ending with machine m.

Job $j \in J$ $(j = 1, .., D)$ requires a processing time $p_{j,k} \geq 0$ in machine $k \in K$ $(k = 1, .., m)$. The overall goal of FSP is to determine a release sequence in which to process the jobs that will optimize one or more efficiency criteria.

A version of FSP, the Permutation Flow Shop Problem (PFSP), considers the storage space between two consecutive machines to be unlimited and therefore assumes that when operation (j, k) of job $j \in J$ in machine $k > 1$ $(k \in K)$ is completed, the machine is able to process the next job in the sequence from the moment the job is released by the previous machine $k - 1$. Using the notation proposed by Graham et al. (1979), this problem is known as $Fm/prmu/\gamma$, where parameter γ symbolizes the selected efficiency measure. The following efficiency metrics are among the most common: (1) Makespan or the time required to complete all operations (j, k) in the workshop, C_{max}, and (2) the average time required to complete such operations, C_{med}.

There are production systems, also composed of machines arranged in series, in which it is not advisable to separate products or jobs within a process due to the product size (e.g., chassis, buses), the nature of the jobs (e.g., chemical reactors), or a lack of space. Under such circumstances, when the operation on a product of machine $k < m$ is completed, the operation will release the product to the next machine $k + 1$ in order to process the next product in the sequence, but if machine $k + 1$ is busy, machine k will be blocked even though it has completed its operation. Using again the notation of Graham et al. (1979), this problem is called $Fm/block/\gamma$, where γ again symbolizes the efficiency measure considered.

The nature of these problems is highly combinatorial and the minimization of the makespan is NP-hard in the strong sense (Hoogeveen et al. 1996; Hall and Sriskandarajah 1996; Yu et al. 2004). For this reason, heuristic procedures have traditionally been used to solve these problems in both permutation versions (Nawaz et al. 1983; Osman and Potts 1989; Taillard 1990; Reeves 1995; Aggoune 2004; Ying and Liao 2004; Fernandez-Viagas et al. 2017) and in versions with interruptions between machines (Logendran and Sriskandarajah 1993; Caraffa et al. 2001; Ronconi 2004, 2005; Ribas et al. 2011; Grabowski and Pempera 2007; Han et al. 2012; Pan and Wang 2012; Lin and Ying 2013; Nouri and Ladhari 2017, Ozolins 2017; Tasgetiren et al. 2017).

After explaining the two problems of interest, we will consider the unrealistic conditions that, in our view, affect the set of jobs J performed in some industrial sectors.

- Traditionally in the various versions of FSP, the elements of the set of jobs J are special, with unusual qualities or more specifically are unique jobs or products.
- If we consider some industrial sectors, such as the automotive sector, it is difficult to find realistic problems whose purpose is to determine efficient sequences of 270 or more different products. In fact, it is absurd to think of a daily sequence of 270 engines or 300 car bodies or 500 chassis or 800 buses, all of which are entirely different.
- Due to the above reasons and for practical purposes, it is reasonable to believe that a typology can be naturally established on the set of jobs J (products, parts, etc.),

and it is therefore possible to discuss types of engines or car bodies or chassis or motors.

- In conclusion, in some industrial sectors, it makes sense to discuss types of jobs or types of products or types of parts.

After these considerations, the remainder of this paper is structured as follows: In Sect. 2, we formalize the natural extension of the FSP when jobs are replicated, which for us means that there exists a demand plan for job types. In Sect. 3, we propose using mixed integer linear programming to model these problems. In Sect. 4, we illustrate the problems under study with an example. In Sect. 5, we perform a computational experiment to analyze the behavior of the generated models. Finally, Sect. 6 is dedicated to conclusions and proposals for future research.

2 Problems $Fm/\beta/\gamma/d_i$: $Fm/\beta/\gamma$ with product types

Formalization:

$Fm/\beta/\gamma/d_i$ is a family of sequencing problems that establishes a bijective application between the elements of a set $\mathcal{T}$ of ordinals (T elements), which correspond to positions in a sequence of releases to manufacturing: $\pi\ (T) = (\pi_1, ., \pi_T)$, and the elements of a set J of jobs or products (D elements, with $D = T$).

The jobs or products in group J are classified into exclusive types or classes, J_i, that satisfy: $J = \bigcup_{i \in I} J_i$ and $J_i \cap J_{i'} = \emptyset, \forall \left\{i, i'\right\} \in I$; where I is the set of job types ($i = 1, .., n$). Parameter β can take the permutation (prmu) or blocking (block) values, parameter γ represents the possible efficiency metrics (e.g., C_{max}, C_{med}), vector $\mathbf{d}$ represents the demand plan for the considered job types, and d_i symbolizes the number of jobs of type $i \in I$ within J; $d_i = |J_i|\ \forall i \in I$, satisfying: $\sum_{\forall i} d_i = D = T$.

The units of J travel in order through a set K of m machines arranged in series. We assume that the production of a job of type $i \in I$ requires a processing time $p_{i,k}$, measured under normal operation conditions, in machine $k \in K$ ($k = 1, \ldots, m$), and that these times are heterogeneous.

The differences between classes J_i (e.g., 4×4 s, vans, trucks) indicate the heterogeneity of the processing times $p_{i,k}$, which results in natural decouplings between the processors (operators and robots) assigned to machines. In some cases, operators must wait for a product to be released from the previous machine before beginning work, and in others, when storage between machines is not possible, the operator will have to wait while "blocked" from the completion of the operation in progress in the next machine, even if his operation on the product in progress is completed. Based on the description above, in this paper we are not going to contemplate the possibility of interrupting operations and will leave this option for future work.

The purpose of problems $Fm/\beta/\gamma/d_i$ is to obtain a sequence of replicated jobs or products (d_i), in a line with m machines, with the possibility of blocking or not according to β, and with the objective of optimizing the efficiency metric represented by the γ value. To formalize this purpose, two mathematical models adapted to mixed integer linear programming (MILP) are presented here.

3 MILP models for problems $Fm/\beta/\gamma/d_i$

Parameters

J	Set of jobs or products (*Jobs*): $j = 1, ., D$
$\mathcal{T}$	Set of positions in the production sequence of products: $t = 1, ., T$
I	Set of types of jobs or products (*Job Types*): $i = 1, ., n$
J_i	Set of jobs or products of type $i \in I$
$\mathbf{d}$; d_i	Demand plan of a job types vector and demand of the jobs or products of type i $(i = 1, ., n)$, with $d_i = \|J_i\| \, \forall i \in I$ and satisfying: $\sum_{\forall i} d_i = D = T \equiv \|\mathcal{T}\|$
K	Group of machines: $k = 1, ., m$
$p_{i,k}$	Processing time of a job or product of type $i \in I$ in machine $k \in K$
C_{max}^0	Upper limit of the makespan or minimum completion time of all the jobs in all machines. This can be calculated based on a reference sequence $\pi^0(T)$ obtained by a heuristic algorithm

Variables

$x_{i,t}$	Binary variable whose value is 1 if a job or product of type $i \in I$ is released to production in the t-th position $(t = 1, ., T)$, and 0 otherwise
$\pi(\cdot)$	Partial sequence, $\pi(t) = (\pi_1, ., \pi_t)$, and full sequence, $\pi(T) = (\pi_1, ., \pi_T)$, of production of jobs or products $j \in J$
$\rho_{k,t}$	Processing time of the t-th job in production sequence $\pi(T)$ in machine $k \in K$
$C_{k,t}$	Time of completion of the t-th job π_t in production sequence $\pi(T)$ in machine $k \in K$. If blocking is considered, $C_{k,t}$ symbolizes the release time of job or product $\pi_t \in \pi(T)$ in the machine $k \in K$
C_i	Time of completion of the last job or product of type $i \in I$ in the last machine $(k = m)$; that is:$C_i = \max_t \{C_{m,t} : x_{i,t} = 1\}$. By convention, we will say that C_i is the time of completion of the batch of parts of type $i \in I$, which is equivalent to the time when all jobs in group J_i have been completed
C_{max}	Makespan: Time of completion of the last job or product π_T of the production sequence $\pi(T)$ in the last machine $(k = m)$; that is: $C_{max} = C_{m,T}$
C_{med}	Average time of completion of batches $(\forall i \in I)$: $C_{med} = \frac{1}{n} \sum_{\forall i} C_i$

MILP model for the problem $Fm/prmu/C_{max}/d_i$

$$\text{MILP-1} \cdot Fm/prmu/C_{max}/d_i : \quad \min C_{max} \equiv \min C_{m,T} \tag{1}$$

Subject to:

$$\sum_{i=1}^{n} x_{i,t} = 1 \quad \forall t = 1, \ldots, T \tag{2}$$

$$\sum_{t=1}^{T} x_{i,t} = d_i \quad \forall i = 1, \ldots, n \tag{3}$$

$$\rho_{k,t} = \sum_{i=1}^{n} p_{i,k} x_{i,t} \quad \forall k = 1, \ldots, m \quad \forall t = 1, \ldots, T \tag{4}$$

$$C_{k,t} \geq C_{k,t-1} + \rho_{k,t} \quad \forall k = 1, \ldots, m \quad \forall t = 1, \ldots, T \tag{5}$$

$$C_{k,t} \geq C_{k-1,t} + \rho_{k,t} \quad \forall k = 1, \ldots, m \quad \forall t = 1, \ldots, T \tag{6}$$

$$x_{i,t} \in \{0, 1\} \quad \forall i = 1, \ldots, n \quad \forall t = 1, \ldots, T \tag{7}$$

$$C_{k,0} = 0 \quad \forall k = 1, \ldots, m \tag{8}$$

$$C_{0,t} = 0 \quad \forall t = 1, \ldots, T \tag{9}$$

In model (MILP $\cdot$ $Fm/prmu/C_{max}/d_i$), objective function (1) represents the minimization of the makespan; equalities (2) help to ensure a position in the sequence of every job or product; equalities (3) are used to ensure the demand plan (vector d) is met; equalities (4) link the processing time of each type of product and machine ($p_{i,k}$) with the corresponding processing time of the t-th job of the sequence ($\rho_{k,t}$); restrictions (5) and (6) serve to limit the minimum completion times of the jobs ($C_{k,t}$), according to the production sequence π (T), in the machines of group K; conditions (7) force the decision variables ($x_{i,t}$) to be binary; and finally (8) and (9) set the start of completion times.

MILP model for the problem $Fm/block/C_{max}/d_i$

$$\text{MILP-2} \cdot Fm/block/C_{max}/d_i : \quad \min C_{max} \equiv \min C_{m,T} \tag{10}$$

Subject to:

$$\sum_{i=1}^{n} x_{i,t} = 1 \quad \forall t = 1, \ldots, T \tag{11}$$

$$\sum_{t=1}^{T} x_{i,t} = d_i \quad \forall i = 1, \ldots, n \tag{12}$$

$$\rho_{k,t} = \sum_{i=1}^{n} p_{i,k} x_{i,t} \quad \forall k = 1, \ldots, m \quad \forall t = 1, \ldots, T \tag{13}$$

$$C_{k,t} \geq C_{k,t-1} + \rho_{k,t} \quad \forall k = 1, \ldots, m \quad \forall t = 1, \ldots, T \tag{14}$$

$$C_{k,t} \geq C_{k-1,t} + \rho_{k,t} \quad \forall k = 1, \ldots, m \quad \forall t = 1, \ldots, T \tag{15}$$

$$C_{k,t} \geq C_{k+1,t-1} \quad \forall k = 1, \ldots, m \quad \forall t = 1, \ldots, T \tag{16}$$

$$x_{i,t} \in \{0, 1\} \quad \forall i = 1, \ldots, n \quad \forall t = 1, \ldots, T \tag{17}$$

$$C_{k,0} = 0 \quad \forall k = 1, \ldots, m \tag{18}$$

$$C_{0,t} = 0 \quad \forall t = 1, \ldots, T \tag{19}$$

$$C_{m+1,t} = 0 \quad \forall t = 1, \ldots, T \tag{20}$$

In the MILP model $Fm/block/C_{max}/d_i$, it is obvious that both the objective function (10) and the constraint blocks (11–15) and (17–19) consecutively match formulas (1–9) of the MILP model $Fm/prmu/C_{max}/d_i$. The changes that are added by considering possible blocking between machines are:

- $C_{k,t}$ here represents the release time (compared to the time of completion) of the t-th job π_t of the production sequence π (T) in machine $k \in K$.
- Restrictions (16) help limit the minimum release time $C_{k,t}$ through the release time of the previous job (π_{t-1}) in the next machine $(k+1)$.
- For convenience, equalities (20) are the release start times in virtual machine: $k = m + 1$.

4 Illustrative example

An assembly line with 21 workstations produces 9 types of engines (M1–M9) grouped into three families (4×4, VAN and Trucks). Figure 1 shows an M1 type engine that belongs to the 4×4 family. Table 1 shows the processing times, measured in seconds under normal operation conditions, for each engine type ($i = 1, \ldots, 9$) in each workstation ($k = 1, \ldots, 21$); these times are heterogeneous and range between 89 and 185 s (Bautista and Cano 2011). Considering a product-oriented online Flow Shop production environment, the goal is to set production sequences of 9 and 18 total engines, composed of 1 and 2 engines of each type, respectively, with the purpose of measuring the economic impact of the elimination of spaces between consecutive workstations.

Fig. 1 Nissan Pathfinder Engine. Characteristics: (i) 747 parts and 330 references, (ii) 378 elemental assembly tasks grouped in 140 production line tasks

Table 1 Processing time under normal operation $(p_{i,k})$ in seconds of the 9 types of engines $(i \in I)$ in the 21 workstations $(k \in K)$ of the set of Nissan-9Ing.I instances (Bautista and Cano 2011)

$k \backslash i$	M1	M2	M3	M4	M5	M6	M7	M8	M9
1	104	100	97	92	100	94	103	109	101
2	103	103	105	107	101	108	106	102	110
3	165	156	164	161	148	156	154	164	155
4	166	175	172	167	168	167	168	156	173
5	111	114	114	115	117	117	115	111	111
6	126	121	122	124	127	130	120	121	134
7	97	96	96	93	96	89	94	101	92
8	100	97	95	106	94	102	103	102	100
9	179	174	173	178	178	171	177	171	174
10	178	172	172	177	178	177	175	173	175
11	161	152	168	167	167	166	172	157	177
12	96	106	105	97	101	100	96	104	96
13	99	101	102	101	99	101	96	102	99
14	147	155	142	154	146	143	154	153	155
15	163	152	156	152	153	152	154	156	156
16	163	185	183	178	169	173	172	182	171
17	173	179	178	169	173	178	174	175	175
18	176	167	181	180	172	173	173	168	184
19	162	150	152	152	160	151	155	148	167
20	164	161	157	159	162	160	162	158	157
21	177	161	154	168	172	170	167	149	169

The engine production sequences must meet the minimum Makespan goal, taking into account two online production configurations in a Flow Shop environment:

- Line L1 with unlimited storage space between pairs of consecutive workstations (problem $Fm/prmu/C_{max}$)
- Line L2 without storage space between pairs of consecutive workstations, with the possibility of blocking between stations (problem $Fm/block/C_{max}$)

The 4 optimal sequences of this example were obtained with implementations of the MILP-1 and MILP -2 models using IBM ILOG CPLEX code (Optimization Studio v.12.2, win-x86-64)

Table 2 shows the results for this example.

Table 2 shows that the elimination of storage space between the 21 workstations (L1 vs. L2) causes slow-downs in production of 10 and 27 s when engines 9 and 18 are produced, respectively.

Considering that the cost of production loss (Bautista et al. 2018) is 137.14 euros per production minute, the elimination of spaces in the assembly line results in an additional cost of 22.86 euros with 9 engines, and 61.71 euros with 18 engines.

The assembly line was designed for a fixed cycle time of 175 s, and consequently, the actual production times available for the manufacture of 9 and 18 engines are 1.42

Table 2 Line (L1, L2) according to parameter β $(prmu, block)$, engine demand plan ($d_i = 1, 2 \forall i$), optimal production sequence π (T), C_{max} value (sec) and CPU time (sec) for the illustrative example with 9 engine types

Line	β	$d_i \forall i$	Sequenece π (T)	C_{max}	CPU (s)
L1	prmu	1	[5 3 9 1 4 7 6 2 8]	4372	0.218
L2	block	1	[5 2 6 1 4 7 9 3 8]	4382	1.079
L1	prmu	2	[5 3 6 9 6 3 1 2 4 1 2 9 5 4 7 7 8 8]	5944	7.361
L2	block	2	[5 2 8 9 9 3 2 4 7 1 7 5 1 6 4 6 3 8]	5971	1648.594

and 1.85 h, respectively. Note that both times are greater than the corresponding C_{max} values in Table 2.

5 Computational experimentation

The computational experimentation proposed is focused on analyzing the behavior of Mixed Integer Linear Programming (MILP) to solve sequencing problems in Flow Shop production environments with extensive demand. We propose two experiments. In Experiment-1 we used a selection of Taillard's instances (Taillard 1993). For Experiment-2 we have selected 7 categorical instances of the set of industrial instances Nissan-9Eng.I (Bautista and Cano 2011).

5.1 Experiment-1

For the Experiment-1, we used Set-1 and Set-4 Taillard's instances (E) from the literature that are related to the sequencing problems studied. We also adapted those instances to industrial cases with general demand for types of jobs or products (E$'$).

In brief, the data included in this experiment are:

- Set-1 (Taillard 1993): instances $\varepsilon = 1, ., 10$ ($\varepsilon \in$ E), number of job types $|I| \equiv n = 20$, number of machines $|K| \equiv m = 5$, and total demand of jobs $T \equiv D = 20$.
- Set-4 (Taillard 1993): instances $\varepsilon = 31, ., 40$ ($\varepsilon \in$ E), number of job types $|I| \equiv n = 50$, number of machines $|K| \equiv m = 5$, and total demand of jobs $T \equiv D = 50$.
- Set-1d (Set-1 adapted to problem $Fm/\beta/\gamma/d_i$): instances $\varepsilon = 1, ., 10$ ($\varepsilon \in$ E$'$), number of job types $|I| \equiv n = 20$, number of machines $|K| \equiv m = 5$, Demand plan of job types $d_i = 5$ ($\forall i = 1, .., 20$), and total demand of jobs $T \equiv D = 100$, with identical processing times $p_{i,k}$ ($\forall i \in I, \forall k \in K$), instance by instance, to those of Set-1.
- Set-4d (Set-4 adapted to problem $Fm/\beta/\gamma/d_i$): instances $\varepsilon = 31, ., 40$ ($\varepsilon \in$ E$'$), number of job types $|I| \equiv n = 50$, number of machines $|K| \equiv m = 5$, Demand plan of job types $d_i = 5$ ($\forall i = 1, ., 20$), and total demand of jobs $T \equiv D = 250$, with identical processing times $p_{i,k}$ ($\forall i \in I, \forall k \in K$), instance by instance, to Set-4.

The compiled codes for the procedures involved were executed on a DELL Inspiron-13 (Intel(R) Core(TM) i7-7500U @ 2.70 GHz CPU 2.90 GHz, 16 GB of RAM, x64 Windows 10 Pro). The characteristics of the 2 procedures are:

- MILP-1: Model $Fm/prmu/C_{max}/d_i$: (i) Objective function for minimizing the C_{max} value of the production sequence; (ii) implementation for IBM ILOG CPLEX solver (Optimization Studio v.12.2, win-x86-64); (iii) maximum CPU time of 7200 s. (40 instances) allowed for solving each instance.
- MILP-2: Model $Fm/block/C_{max}/d_i$: (i) Objective function for minimizing the C_{max} value of the production sequence; (ii) implementation for IBM ILOG CPLEX solver (Optimization Studio v.12.2, win-x86-64); (iii) maximum CPU time of 7200 s. (40 instances) allowed for solving each instance.

Tables 3, 4, 5 and 6 show the results of the experiment obtained by CPLEX for the two models implemented $(Fm/prmu/C_{max}/d_i,\ Fm/block/C_{max}/d_i)$. Table 3 corresponds to the instances of Set-1 (20 jobs), Table 4 corresponds to those of Set-4 (50 jobs), and Tables 5 and 6 correspond to those of Set-1d (100 jobs) and of Set-4d (250 jobs), respectively.

In the tables, the column headers represent the following characteristics:

$\varepsilon \in \mathrm{E}$	Identification number of the instances for Set-1 and Set-4
$\varepsilon \in \mathrm{E}'$	Identification number of the instances for Set-1d and Set-4d
C_{max}	Best makespan value obtained for procedure MILP-1 or MILP-2
C^*_{max}	Optimal value of makespan
C^0_{max}	Best known value of makespan
LB_p	C_{max} lower limit for problem $Fm/prmu/C_{max}/d_i$ obtained for MILP-1
LB_b	C_{max} lower limit for problem $Fm/block/C_{max}/d_i$ obtained for MILP-1/2
Gap_{C^*}	Relative gap between C_{max} and C^*_{max}
Gap_{C^0}	Relative gap between C_{max} and C^0_{max}
Gap_{LB_p}	Relative gap between C_{max} and LB_p
Gap_{LB_b}	Relative gap between C_{max} and LB_b

The relative gap values between C_{max} and the other characteristics related to makespan $(C^*_{max}, C^0_{max}, LB\_p, LB\_b)$ are calculated using (21).

$$Gap_X(\varepsilon) = \frac{C_{max}(\varepsilon) - X(\varepsilon)}{C_{max}(\varepsilon)} \quad X \in \left\{LB_p, LB_b, C^*_{max}, C^0_{max}\right\}, \quad \forall\varepsilon \in \mathrm{E}, \forall\varepsilon \in \mathrm{E}' \tag{21}$$

where the LB_p values for the problem $Fm/prmu/C_{max}/d_i$ are obtained directly with procedure MILP-1, the LB_b values correspond to the maximum value between LB_p and the lower limit from procedure MILP-2 for problem $Fm/block/C_{max}/d_i$. Meanwhile, the values of C^*_{max} are confirmed as optimal through procedure MILP-1, and the C^0_{max} value originates from the literature for problem $Fm/block/C_{max}$ (Bautista et al. 2012).

An analysis of Tables 3, 4, 5 and 6 reveals the following:

- Procedure MILP-1 obtains and ensures optimal solutions in all instances with 20 jobs (Set-1), 50 jobs (Set-4) and 100 jobs (Set-1d).

Table 3 Results for Set-1 instances using procedures MILP-1 and MILP-2

Procedure	MILP-1: $Fm/prmu/C_{max}/d_i$ (=1), $D = 20$				MILP-2: $Fm/block/C_{max}/d_i$ (=1), $D = 20$					
$\varepsilon \in$ E	C^*_{max}	C_{max}	Gap_{C^*}	$CPU\ (s)$	LB_b	C^0_{max}	C_{max}	Gap_{C^0}	Gap_{LB_b}	$CPU\ (s)$
1	1278	1278	0.00	16.49	1325	1374	1379	0.004	0.039	7176
2	1359	1359	0.00	13.58	1360	1408	1417	0.006	0.040	7063
3	1081	1081	0.00	96.95	1206	1280	1301	0.016	0.073	7074
4	1293	1293	0.00	20.60	1369	1448	1448	0.000	0.054	7191
5	1235	1235	0.00	64.71	1303	1341	1348	0.005	0.034	7070
6	1195	1195	0.00	16.86	1271	1363	1369	0.004	0.071	7079
7	1234	1234	0.00	15.83	1280	1381	1390	0.006	0.079	7088
8	1206	1206	0.00	15.89	1310	1379	1379	0.000	0.050	7039
9	1230	1230	0.00	15.06	1292	1373	1380	0.005	0.064	7058
10	1108	1108	0.00	14.31	1166	1283	1283	0.000	0.091	7096
Average	–	–	0.00	29.03				0.005	0.060	7093

Table 4 Results for Set-4 instances using procedures MILP-1 and MILP-2

Procedure	MILP-1: $Fm/prmu/C_{max}/d_i$ (=1), $D = 50$				MILP-2: $Fm/block/C_{max}/d_i$ (=1), $D = 50$					
$\varepsilon \in$ E	C^*_{max}	C_{max}	Gap_{C^*}	CPU (s)	LB_b	C^0_{max}	C_{max}	Gap_{C^0}	Gap_{LB_b}	CPU (s)
31	2724	2724	0.00	15.11	2724	3002	3091	0.029	0.119	7175
32	2834	2834	0.00	916.97	2838	3201	3282	0.025	0.135	7094
33	2621	2621	0.00	524.52	2621	3011	3155	0.046	0.169	7075
34	2751	2751	0.00	1979.61	2751	3128	3225	0.030	0.147	6986
35	2863	2863	0.00	171.44	2867	3166	3277	0.034	0.125	7183
36	2829	2829	0.00	109.35	2829	3169	3288	0.036	0.140	7107
37	2725	2725	0.00	530.91	2725	3013	3147	0.043	0.134	2932
38	2683	2683	0.00	440.43	2684	3073	3174	0.032	0.154	7164
39	2552	2552	0.00	1173.59	2552	2908	3017	0.036	0.154	7157
40	2782	2782	0.00	3.03	2782	3120	3234	0.035	0.140	7120
Average	–	–	0.00	586.50				0.035	0.142	6699

Table 5 Results for Set-1d instances using procedures MILP-1 and MILP-2

Procedure	MILP-1: $Fm/prmu/C_{max}/d_i$ (=5), $D = 100$				MILP-2: $Fm/block/C_{max}/d_i$ (=5), $D = 100$			
$\varepsilon \in \mathrm{E}'$	LB_p	C_{max}	Gap_{LB_p}	CPU (s)	LB_b	C_{max}	Gap_{LB_b}	CPU (s)
1	5748	5748	0.00	195.43	5748	6625	0.132	7197
2	6183	6183	0.00	65.79	6183	6624	0.067	7112
3	5067	5067	0.00	0.83	5067	6352	0.202	6910
4	5976	5976	0.00	1.05	5976	7031	0.150	7036
5	5637	5637	0.00	45.88	5637	6449	0.126	7147
6	5671	5671	0.00	29.77	5678	6646	0.146	7168
7	5834	5834	0.00	3.17	5834	6677	0.126	7103
8	5560	5560	0.00	14.08	5560	6801	0.182	7193
9	5758	5758	0.00	9.64	5758	6727	0.144	6928
10	5118	5118	0.00	4.99	5118	6260	0.182	7198
Average			0.00	37.06			0.146	7099

Table 6 Results for Set-4d instances using procedures MILP-1 and MILP-2

Procedure	MILP-1: $Fm/prmu/C_{max}/d_i$ (=5), $D = 250$				MILP-2: $Fm/block/C_{max}/d_i$ (=5), $D = 250$			
$\varepsilon \in \mathrm{E}'$	LB_p	C_{max}	Gap_{LB_p}	CPU (s)	LB_b	C_{max}	Gap_{LB_b}	CPU (s)
31	13,408	13,408	0.0000	10.22	13,408	16,379	0.181	2208
32	13,776	13,793	0.0012	328.95	13,776	17,570	0.216	2107
33	12,809	12,809	0.0000	5567.52	12,809	15,982	0.199	3686
34	13,436	13,436	0.0000	734.00	13,446	16,906	0.205	6236
35	13,975	13,975	0.0000	6599.62	13,995	17,583	0.204	4832
36	13,783	13,783	0.0000	943.01	13,783	17,176	0.198	7092
37	13,189	13,189	0.0000	1645.21	13,189	16,624	0.207	6909
38	13,267	13,267	0.0000	11.14	13,284	16,674	0.203	2594
39	12,407	12,424	0.0014	583.24	12,409	15,864	0.218	6524
40	13,648	13,648	0.0000	11.85	13,648	16,757	0.186	2256
Average			0.0003	1643.47			0.202	4444

- Procedure MILP-1 obtains and ensures optimal solutions in 8 of the 10 instances with 250 jobs (Set-4d). The solutions obtained with MILP-1 for instances #32 and #39 of Set-4d have Gap_{LB_p} values equal to 0.12% and 0.14%, respectively. The average Gap_{LB_p} value for Set-4d is approximately 0.03%.
- Procedure MILP-2 does not ensure optimal solutions in any of the four Sets (1, 4, 1d and 4d).
- In Set-1 (20 jobs), MILP-2 obtains 3 better solutions for 10 instances (instances #4, #8 and #10) and offers a Gap_{C^0} average value of 0.5%. Meanwhile, in the instances with 50 jobs (Set-4), MILP-2 obtains a Gap_{C^0} average value of approximately 3.5% and is not able to match any better known value (C^0_{max}).
- For instances with 100 and 250 jobs (Set-1d and Set-4d) and with blocking between machines, there is no information on the best known makespan value; therefore, to measure the quality of the solutions provided by MILP-2, we used the makespan lower limits offered by the procedure. Under such conditions, the average values of Gap_{LB_b} are equal to 0.146 for Set-1d and to 0.202 for Set-4d.
- The average CPU times used by MILP-1 are approximately 29, 587, 37 and 1643 s for each instance of 20, 50, 100 and 250 jobs, respectively. Note that these times do not increase progressively with the number of jobs to be sequenced ($T \equiv D$), but these times do appear to depend on the number of job types ($|I| \equiv n$) that correspond to each instance.
- The average CPU times used by MILP-2 are approximately 7093, 6699, 7099 and 4444 s for each instance of 20, 50, 100 and 250 jobs, respectively. These times are not related to the number of jobs to be sequenced ($T \equiv D$) or to the number of job types ($|I| \equiv n$).
- In summary, considering the CPU time limitation of 7200 s for each instance, the MILP-1 procedure oriented toward problem $Fm/prmu/C_{max}/d_i$ obtains and ensures 38 optimal solutions for the 40 instances studied, while procedure MILP-2, oriented toward problem $Fm/block/C_{max}/d_i$, obtains 3 better solutions for a total of 20 instances (Set-1 and Set-4).

5.2 Experiment-2

There are currently 23 production plans for the nine engines and one working day at the Nissan Spanish Industrial Operations (Bautista and Cano 2011). Each program corresponds to a set of operation times biased by the demand of each of the nine products. We summarize here the characteristics of each of the 23 production plans. We have grouped them into seven categories according to the type of engine demand. One representative production plan is selected for each category to be used in the computational experimentation developed in this subsection. As said, the total number of engines assembled in a working day is 270 in two shifts:

- Category-1 (plan #1): identical demand for each of the nine products (balanced demand) (30 engines per product type).
- Category-2 (plan #2): identical demand for each of the three engine families: 4x4, VAN, and trucks (90 per product family).

Table 7 Daily demands by product type and plan $(d_{i,\varepsilon})$ for the 7 instances categorical Nissan-9Eng.I ($\varepsilon \in$ E)

Plan#ε	M1	M2	M3	M4	M5	M6	M7	M8	M9	4x4	Van	Truck	Total
1	30	30	30	30	30	30	30	30	30	90	60	120	270
2	30	30	30	45	45	23	23	22	22	90	90	90	270
3	10	10	10	60	60	30	30	30	30	30	120	120	270
6	50	50	50	30	30	15	15	15	15	150	60	60	270
9	70	70	70	15	15	8	8	7	7	210	30	30	270
12	24	23	23	45	45	28	28	27	27	70	90	110	270
18	60	60	60	30	30	8	8	7	7	180	60	30	270

- Category-3 (plan #3): one of the engine families has low demand while the demand of the other two families is high and identical.
- Category-4 (plan #6): one of the engine families has high demand while the demand of the other two families is medium and identical.
- Category-5 (plan #9): one of the engine families has high demand while the demand of the other two families is low and identical.
- Category-6 (plan #12): the demand of the engine families follows an arithmetic progression.
- Category-7 (plan #18): the demand of the engine families follows a geometric progression.

Table 1 shows the processing times, measured in seconds under normal operation conditions, for each engine type ($i = 1, \ldots, 9$) in each workstation ($k = 1, \ldots, 21$). On the other hand, Table 7 shows daily demands by engine type and plan for the 7 instances categorical Nissan-9Eng.I.

The compiled codes for the procedures involved were executed on a DELL Inspiron-13 (Intel(R) Core(TM) i7-7500U @ 2.70 GHz CPU 2.90 GHz, 16 GB of RAM, x64 Windows 10 Pro). The characteristics of procedures are:

- MILP-1: Model $Fm/prmu/C_{max}/d_i$: (i) Objective function for minimizing the $\mathrm{C_{max}}$ value of the production sequence; (ii) implementation for IBM ILOG CPLEX solver (Optimization Studio v.12.2, win-x86-64); (iii) maximum CPU time of 180 s allowed for solving each instance (7 instances).
- MILP-2: Model $Fm/block/C_{max}/d_i$: (i) Objective function for minimizing the $\mathrm{C_{max}}$ value of the production sequence; (ii) implementation for IBM ILOG CPLEX solver (Optimization Studio v.12.2, win-x86-64); (iii) maximum CPU time of 180 s allowed for solving each instance (7 instances).

Table 8 shows the results of the experiment obtained by CPLEX for the two models implemented (MILP-1 $Fm/prmu/C_{max}/d_i$, and MILP-2: $Fm/block/C_{max}/d_i$).

The analysis of Table 8 reveals the following:

- Procedure MILP-1 obtains and ensures optimal solutions in all instances with 270 jobs (7 instances categorical Nissan-9Eng.I) when we resolve the $Fm/prmu/C_{max}/d_i$ problem.

Table 8 Results for 7 Nissan-9Eng.I instances categorical using procedures MILP-1 and MILP-2 (180 s. CPU)

$Plan\#\varepsilon$	MILP-1		MILP-2			
	C^*_{max}	$CPU\ (s)$	LB_b	C_{max}	Gap_{LB_b}	$CPU\ (s)$
1	50,091	45.84	50,091	51,094	0.020	180.33
2	50,174	15.19	50,174	51,006	0.016	180.25
3	50,301	10.34	50,301	50,757	0.009	180.13
6	50,202	14.26	50,203	51,072	0.017	180.17
9	50,378	10.39	50,378	51,385	0.020	180.15
12	50,192	17.41	50,193	51,071	0.017	180.16
18	50,273	14.28	50,273	51,267	0.019	180.16
Average	50,230.14	18.24	50,230.43	51,093.14	0.017	180.19
Max	50,378	45.84	50,378	51,385	0.020	180.33
Min	50,091	10.34	50,091	50,757	0.009	180.13

- Procedure MILP-2 does not ensure optimal solutions in any of the instances with 270 job (7 instances categorical Nissan-9Eng.I) in the $Fm/block/C_{max}/d_i$ problem.
- The average values of Gap_{LB_b} are equal to 1.69% for Set categorical Nissan-9Eng.I.
- The average CPU times used by MILP-1 are approximately 18.24 s for each instance of 270 jobs (7 instances categorical Nissan-9Eng.I).
- The average CPU times used by MILP-2 are approximately 180.19 s for each instance of 270 jobs (7 instances categorical Nissan-9Eng.I) when we impose a maximum CPU time of 180 s allowed for solving each instance.
- Considering that the cost of production loss (Bautista et al. 2018; Bautista-Valhondo and Alfaro-Pozo 2018) is 137.14 euros per production minute, the elimination of spaces in the assembly line (prmu vs. block) results in an additional cost average of 1972.57 euros/day with 270 engines. However, the original assembly line was designed for a fixed cycle time of 175 s, and consequently, the actual production times available for the manufacture of 270 engines is 50,770 s.

6 Conclusions

In this study, we presented and justified a natural extension of two classic sequencing problems: $Fm/prmu/C_{max}$ and $Fm/block/C_{max}$. This extension is motivated by our concern over adapting academic problems closer to reality in industrial environments related to the automotive sector.

Our extension takes into account the type of jobs or products in such problems, based on the fact that, in many highly standardized industrial sectors such as the automotive sector, it is unlikely to find productive processes where all jobs or all products (chassis, car bodies, engines, seats, etc.) are completely different. For this reason, we incorporated the concept of a demand plan of job types ($d_i \forall i \in I$) into the original problems, which resulted in the problems $Fm/prmu/C_{max}/d_i$ and $Fm/block/C_{max}/d_i$. The concept of a demand plan of job (product) types can be extrapolated to other variants of Flow Shop and Job Shop problems, if the circumstances are appropriate.

We formulated Mixed Integer Linear Programming models, implemented in CPLEX, for the two new problems and analyzed the quality of the procedures through a computational experiment with instances collected and adapted from the literature.

Our computational experience is composed of two experiments. In Experiment-1 we have used a selection of 20 instances (from the classic instances of Taillard), whose dimensions we have adapted to the automotive industry. In Experiment-2 we have selected 7 categorical instances (from the set of Nissan-9Eng.I instances) corresponding to 7 engine production plans in the Nissan factory in Barcelona, and whose dimensions are: 270 products, 9 types of engines and 21 work stations.

Taking into account the results of the first experiment, we conclude that it is not prudent to discard Mixed Integer Linear Programming for solving sequencing problems in Flow Shop production environments, as MILP is a competitive technique with which to solve problem $Fm/prmu/C_{max}/d_i$ for industrial instances of 250 jobs and obtains and confirms the optimal solutions in most instances with an average CPU time of less than 28 min. MILP is less effective with problem $Fm/block/C_{max}/d_i$ with industrial instances, although its results are acceptable when compared to the best solutions in the literature for problem $Fm/block/C_{max}$.

Looking at the results of Experiment-2, that are more realistic for the automotive industry, we conclude that MILP has offered very satisfactory solutions for the two new problems proposed; in effect: (i) MILP gets the 7 optimal solutions for the problem $Fm/prmu/C_{max}/d_i$ in an average CPU time of 18.24 s, and (ii) MILP gets solutions, for the problem $Fm/block/C_{max}/d_i$, with an average value that are 0.17% of the average value of the optimal solutions, when we set a maximum CPU time of 3 min.

Therefore, we conclude that MILP should be incorporated into the set of tools dedicated to solving sequencing problems in realistic production environments. The role of MILP within the set of techniques for solving such problems can be a leading role or as part of the metaheuristic procedures that combine a construction phase of one or more initial solutions, with one or several phases of local improvement of such solutions.

In future work, we intend to apply the knowledge acquired in this experimental study to various case studies related to sequencing problems of mixed models in product-oriented production systems (assembly lines and workshops with regular flow). We also intend to analyze the economic impact of the alternative of using a fixed production cycle time for all machines (processors, workstations) or allowing processors to have cycle times that depend on both products and machines.

Acknowledgements This research was subsidized by the Ministry of Economy and Competitiveness of the Government of Spain through project FHI-SELM2 (TIN2014-57497-P).

References

Aggoune R (2004) Minimizing the makespan for the flow shop scheduling problem with availability constraints. Eur J Oper Res 153(3):534–543. https://doi.org/10.1016/S0377-2217(03)00261-3

Bautista J, Cano A (2011) Solving mixed model sequencing problem in assembly lines with serial workstations with work overload minimisation and interruption rules. Eur J Oper Res 210(3):495–513. https://doi.org/10.1016/j.ejor.2010.10.022

Bautista-Valhondo J, Alfaro-Pozo R (2018) An expert system to minimize operational costs in mixed-model sequencing problems with activity factor. Expert Syst Appl 104:185–201. https://doi.org/10.1016/j.eswa.2018.03.031

Bautista J, Cano A, Companys R, Ribas I (2012) Solving the *Fm|block|Cmax* problem using Bounded Dynamic Programming. Eng Appl Artif Intell 25(6):1235–1245. https://doi.org/10.1016/j.engappai.2011.09.001

Bautista J, Alfaro-Pozo R, Batalla-García C (2018) Minimizing lost-work costs in a mixed-model assembly line. In: Viles E, Ormazábal M, Lleó A (eds) Closing the gap between practice and research in industrial engineering. Lecture notes in management and industrial engineering. Springer, Cham. https://doi.org/10.1007/978-3-319-58409-6_24

Caraffa V, Ianes S, Bagchi TP, Sriskandarajah C (2001) Minimizing makespan in a blocking flowshop using genetic algorithms. Int J Prod Econ 70(2):101–115. https://doi.org/10.1016/S0925-5273(99)00104-8

Fernandez-Viagas V, Ruiz R, Framinan JM (2017) A new vision of approximate methods for the permutation flowshop to minimise makespan: state-of-the-art and computational evaluation. Eur J Oper Res 257(3):707–721. https://doi.org/10.1016/j.ejor.2016.09.055

Grabowski J, Pempera J (2007) The permutation flow shop problem with blocking. A tabu search approach. Omega 35(3):302–311. https://doi.org/10.1016/j.omega.2005.07.004

Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG (1979) Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann Discret Math 5:287–326. https://doi.org/10.1016/S0167-5060(08)70356-X

Hall NG, Sriskandarajah C (1996) Survey of machine scheduling problems with blocking and no-wait in process. Oper Res 44(3):510–525. https://doi.org/10.1287/opre.44.3.510

Han YY, Pan QK, Li JQ et al (2012) An improved artificial bee colony algorithm for the blocking flowshop scheduling problema. Int J Adv Manuf Technol 60:1149–1159. https://doi.org/10.1007/s00170-011-3680-0

Hoogeveen JA, Lenstra JK, Veltman B (1996) Preemptive scheduling in a two-stage multiprocessor flow shop is NP-hard. Eur J Oper Res 89(1):172–175. https://doi.org/10.1016/S0377-2217(96)90070-3

Lin SW, Ying KC (2013) Minimizing makespan in a blocking flowshop using a revised artificial immune system algorithm. Omega 41(2):383–389. https://doi.org/10.1016/j.omega.2012.03.006

Logendran R, Sriskandarajah C (1993) Two-machine group scheduling problem with blocking and anticipatory setups. Eur J Oper Res 69(3):467–481. https://doi.org/10.1016/0377-2217(93)90029-M

Nawaz M, Enscore EE, Ham I (1983) A heuristic algorithm for the m-machine, n-job flow-shop sequencing problema. Omega 11(1):91–95. https://doi.org/10.1016/0305-0483(83)90088-9

Nouri N, Ladhari T (2017) Evolutionary multiobjective optimization for the multi-machine flow shop scheduling problem under blocking. Ann Oper Res. https://doi.org/10.1007/s10479-017-2465-8

Osman IH, Potts CN (1989) Simulated annealing for permutation flow-shop scheduling. Omega 17(6):551–557. https://doi.org/10.1016/0305-0483(89)90059-5

Ozolins A (2017) Improved bounded dynamic programming algorithm for solving the blocking flow shop problem. Cent Eur J Oper Res. https://doi.org/10.1007/s10100-017-0488-5

Pan QK, Wang L (2012) Effective heuristics for the blocking flowshop scheduling problem with makespan minimization. Omega 40(2):218–229. https://doi.org/10.1016/j.omega.2011.06.002

Pinedo ML (2016) Scheduling (theory, algorithms, and systems). Springer International Publishing, Berlin. https://doi.org/10.1007/978-3-319-26580-3

Reeves CR (1995) A genetic algorithm for flowshop sequencing. Comput Oper Res 22(1):5–13. https://doi.org/10.1016/0305-0548(93)E0014-K

Ribas I, Companys R, Tort-Martorell X (2011) An iterated greedy algorithm for the flowshop scheduling problem with blocking. Omega 39(3):293–301. https://doi.org/10.1016/j.omega.2010.07.007

Ronconi DP (2004) A note on constructive heuristics for the flowshop problem with blocking. Int J Prod Econ 87(1):39–48. https://doi.org/10.1016/S0925-5273(03)00065-3

Ronconi DP (2005) A branch-and-bound algorithm to minimize the makespan in a flowshop with blocking. Ann Oper Res 138(1):53–65. https://doi.org/10.1007/s10479-005-2444-3

Taillard E (1990) Some efficient heuristic methods for the flow shop sequencing problema. Eur J Oper Res 47(1):65–74. https://doi.org/10.1016/0377-2217(90)90090-X

Taillard E (1993) Benchmarks for basic scheduling problems. Eur J Oper Res 64(2):278–285. https://doi.org/10.1016/0377-2217(93)90182-M

Tasgetiren MF, Kizilay D, Pan QK, Suganthanc PN (2017) Iterated greedy algorithms for the blocking flowshop scheduling problem with makespan criterion. Comput Oper Res 77:111–126. https://doi.org/10.1016/j.cor.2016.07.002

Ying KC, Liao CJ (2004) An ant colony system for permutation flow-shop sequencing. Comput Oper Res 31(5):791–801. https://doi.org/10.1016/S0305-0548(03)00038-8

Yu W, Hoogeveen H, Lenstra JK (2004) Minimizing makespan in a two-machine flow shop with delays and unit-time operations is NP-hard. J Sched 7(5):333–348. https://doi.org/10.1023/B:JOSH.0000036858.59787.c2