Expert Systems with Applications: X 5 (2020) 100022

Contents lists available at ScienceDirect

Expert Systems with Applications: X

journal homepage: www.elsevier.com/locate/eswax

Models and an exact method for the Unrelated Parallel Machine scheduling problem with setups and resources

Luis Fanjul-Peyro

Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática, Ciudad Politécnica de la Innovación, Edifico 8G, Acc. B, Universitat Politècnica de València, Camino de Vera s/n, València *46021, Spain*

ARTICLE INFO

Article history:
Received 11 May 2019
Revised 2 October 2019
Accepted 8 January 2020
Available online 10 January 2020

Keywords:
Scheduling
Parallel machines
Sequence dependent setup times
Additional resources
Makespan

ABSTRACT

This paper deals with the Unrelated Parallel Machine scheduling problem with Setups and Resources (UPMSR) with the objective of minimizing makespan. Processing times and setups depend on machine and job. The necessary resources could be: specific resources for processing, needed for processing a job on a machine; specific resources for setups, needed to do the previous setup before a job is processed on a machine; shared resources, understanding these as unspecific resources that could also be needed in both processing or setup. The number of scarce resources depends on machine and job. As an industrial example, in a plastic processing plant molds are the specific resource for processing machines, cleaning equipment is the specific resource for setups and workers are the unspecific shared resource to operate processing machines and setup cleaning equipment. A mixed integer linear program is presented to model this problem. Also a three phase algorithm based on mathematical exact method is introduced. Model and algorithm are tested in a comprehensive and extensive computational campaign. Tests show good results for different combinations of useE of resources and in most cases come to less than 2.7% of gap against lower bound for instances of 400 jobs.

1. Introduction

In industry, scheduling and a limited number of resources is present in most of its activities. All of this forces to optimize objectives. Literature deals with a great number of different problems. One of the most studied problems is called parallel machine scheduling problem. In this problem, in parallel, a set of jobs have to be processed on a set of machines. In our case, each job is processed on only one machine without preemption and each machine can process only one job at a time. We call the general case Unrelated in which the processing time depends on the machine where the job is assigned. An example of the Unrelated Parallel Machine scheduling problem (UPM) is a ceramics plant where different orders have to be assigned to the different kilns they have. One of the most common objectives for this kind of problem is to minimize maximum completion time, known as makespan or C_{max}. Denoted as $R||C_{max}$ (Graham,Lawler,Lenstra,& Rinnooy Kan,1979), this problem is NP-Hard even for the case of two identical parallel machines (Lenstra, Rinnooy Kan, & Brucker, 1977). The most common additional consideration to add to UPM problems is setup time which consists in preparing a machine before a job is processed on this machine (adjustments, cleaning, reconfigurations, etc.). Scheduling with setups literature is quite extensive. A recent review (Allahverdi, 2015) shows hundreds of papers in the last 10 years that deal with UPM setups. A general case is when setup time depends on the machine and job sequence, namely Unrelated Parallel Machine scheduling problem with sequence dependence Setup times (UPMS), which is considered in this paper. Notice UPM is only an assignment problem (the order of jobs assigned on each machine is not important) but when setups are present (UPMS) it is an assignment and sequencing problem. This makes the problem harder to solve. A recent paper (Fanjul-Peyro, Ruiz, & Perea, 2019) deals with the UPMS problem studying instances up to 1000 jobs.

On the other hand, the literature does not usually deal with the issue of resources necessary to do the jobs. In many industrial problems, not taking resources into account is not solving the real problem. In this paper it is assumed that a discrete amount of scarce renewable resources are needed. This amount depends on the job and on the machine. The resources are assumed as *renewable*, meaning that after their use they are again available and *discrete*, because there are an integer number of resources, this is similar to papers such as Błazewicz, Lenstra, and Rinnooy Kan

E-mail address: lfpeyro@hotmail.com

https://doi.org/10.1016/j.eswax.2020.100022

(1983) or Słowinski (1980) They are also considered *unspecified* since there is no pre-fixed job-machine assignment, and *dynamic* because resources are not fixed for the whole time horizon. Unrelated Parallel Machine scheduling problem with Resources is called UPMR. Recent papers such as Fanjul-Peyro, Perea, and Ruiz (2017) and Fleszar and Hindi (2018) deal with this UPMR problem with instances up to 1000 jobs.

In this paper on the other hand, we are going to consider setups and different kinds of resources: (1) specific resources for processing, needed to process a job on a machine, (2) specific resources for setups, needed to do the previous setup before a job is processed on a machine, (3) shared resources, understanding these as unspecific resources that could also be needed in both processing or setup. As an example, consider a plastic processing plant where cleaning equipment is the specific setup resource, molds are the specific processing resource and workers are the unspecific shared resource. Workers are needed to operate the cleaning equipment (setups) and line machines (processing). Another example is when you have specialized workers as in a ceramics factory where there are setup engineers (setups), technical line workers (processing) and unspecialized helpers that have auxiliary tasks to help in both, processing or setups. Notice that when resources are present, not only assignment and sequencing is needed because it is also necessary to know the time when each job begins since idle time can be present due to resource shortages (referred to in this paper as timing). When considering, for the Unrelated Parallel Machine scheduling problem, the Setup and Resources, it could be called UPMSR. This problem has been seldom studied and not for all the constraint combinations present in this paper.

In this paper, a mathematical integer linear program is introduced to solve all combinations of use of resources (in processing, setups and/or shared), obtaining solutions for up to 50 jobs and 8 machines. An exact algorithm is also presented which adapts the methodology used in the most recent papers mentioned in this section for UPMR and UPMS problems. The algorithm allows us to obtain solutions for up to 400 jobs and 8 machines in tests that in most cases give a gap between solutions and lower bounds (LB) of less than 2.7% for the largest instances.

The rest of the paper is structured as follows: Section 2 reviews the literature related to the problem. In Section 3 problem definition and examples are present. Section 4 introduces the complete mathematical model for UPMSR. In Section 5 a three phase exact algorithm is explained (phase of assignment, sequencing and timing). In Section 6 all computational experiments are shown and finally in Section 7 the conclusion and future research are presented. An appendix shows more models and tests.

2. Literature review

In the literature UPM is widely studied (see Pinedo, 2016 for some references). However UPMS is comparatively less studied. Different mixed integer linear programs (MILP) have been proposed for UPMS. Based on the model presented in Guinet (1991), an improvement of this model can be found in Vallada and Ruiz (2011) Later, a more efficient MILP was presented in Avalos-Rosales, Angel-Bello, and Alvarez (2015) showing efficient MILP. In this last paper mentioned, instances up to 60 jobs and 8 machines are solved. Using this MILP as base for a master problem in an iterative algorithm, Tran, Araujo, and Beck (2016) obtained solutions for instances of UPMS up to 120 jobs. In a recent paper, Fanjul-Peyro, Ruiz, and Perea (2019) got better results than these two previously mentioned papers introducing improvements in model and an exact algorithm. MILP gives solutions for up to 400 jobs and the new exact algorithm could reach up to 1000 jobs and 8 machines with relative deviation from lower bounds (LB) below 0.8%.

On the other hand, a more specific or simpler version of UPMR has been studied in recent decades. In Błazewicz, Kubiak, Röck, and Szwarcfiter (1987) this problem for identical machine and minimization of flow time was studied. A static version of this problem where allocation of resources on machines are fixed for the whole time horizon was proposed, for instance, in Daniels, Hua, and Webster (1999) with identical machines. In that paper, a MILP and heuristics were presented. A dynamic version is also studied in papers as Grigoriev, Sviridenko, and Uetz (2007). Where processing times depend on the number of allocated resources. A simplified version of UPMR was studied in Edis and Oguz (2011) where resources do not depend on the machine where the job is assigned. Other models and variants were studied in Edis and Ozkarahan (2012) or Edis, Oguz and Ozkarahan (2013). Finally, Fanjul-Peyro, Perea, and Ruiz (2017) presented a model based on strip-packing problems to solve a more complex dynamic and unspecific version of UPMR, the kind of resources used in the present paper. For a survey of two-dimensional packing problems see Lodi, Martello, and Monaci (2002). Recently, in Fleszar and Hindi (2018) a two stage and complete model based on constraint programing (CP) has been presented, reaching, in the two stage case, up to 1000 jobs with a gap with LB less than 1.9%.

Recently, some paper deals with a specific case of parallel machines scheduling problem with setup and resources. Akyol Özer and Saraç (2019) deal with identical parallel machine with the objective of minimizing total weighted completion times where resources are only necessary for processing. Bektur and Saraç (2019) studied the case of a common server as a resource for setups which implies that setups can be made only on one machine at the same time by the common server and setup times do not depend on the machine. As we can see, these two are particular versions of the problem we study in this paper, which can be seen as a more general case.

In the current paper we focus on MILP and exact methods, therefore no heuristic nor metaheuristic algorithms are presented leaving these for future research as in recent articles with resources (but without setups) such as Zheng and Wang (2016) and Villa, Vallada, and Fanjul-Peyro (2018). But nevertheless, to the best of the author's knowledge, no paper deals with the complete generalizations used here, that include setups and resources together.

3. Problem definition and example

In order to present a complete example we need to define the input data for UPMSR. Superscripts are used in this paper to represent the type of resources and they are not index nor power functions.

- A set of n jobs to be processed, $N = \{1,\dots n\}$, indexed by j, k.
- A set of m machines that can process the jobs, denoted as $M = \{1,\dots, m\}$, indexed by i.
- p_{ij} denotes the time needed for machine i to process job j.
- s_{ijk} denotes the time needed to do the setup on machine i between job j and k.
- $r^{\mathcal{P}}_{max}$ are the maximum amount of Processing resources. Some of these are needed specifically for Processing jobs, and cannot be used for machine setups. We denote these resources as P-resources.
- $r^{\mathcal{S}}_{max}$ are the maximum amount of Setups resources. Some of these are needed specifically for machine Setups, and cannot be used for job processing. We denote these resources as S-resources.
- $r^{\mathcal{H}}_{max}$ are the maximum amount of unspecific resources that are sHared by both, needs of job processing and needs of machine setups. We denote these resources as H-resources.

p_{ij}

	J1	J2	J3	J4
M1	3	1	3	2
M2	3	2	2	1

s_{ijk}

M1	K1	K2	K3	K4
J1	0	2	7	7
J2	7	0	7	7
J3	7	7	0	3
J4	7	7	2	0

M2	K1	K2	K3	K4
J1	0	2	7	7
J2	2	0	7	7
J3	7	7	0	3
J4	7	7	7	0

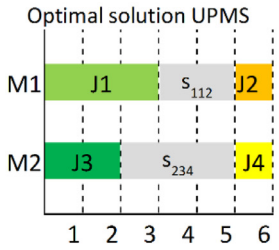

Fig. 1. UPMS instance and optimal solution.

- $r_{ij}^{\mathcal{P}}$are the resources of P-resources needed to process job j on machine i.
- $r_{ijk}^{\mathcal{S}}$ are the resources of S-resources needed to do setup between job j and job k on machine i.
- $r_{ij}^{\mathcal{H}p}$ are the resources of H-resources needed to process job j on machine i. Processing job j on machine i could require $r_{ij}^{\mathcal{P}}$ of P-resources and $r_{ij}^{\mathcal{H}p}$ of H-resources.
- $r_{ijk}^{\mathcal{H}p}$ are the resources of H-resources needed to do setup between job j and job k on machine i. Doing the setup on machine i between jobs j and k could require $r_{ijk}^{\mathcal{S}}$ of S-resources and $r_{ijk}^{\mathcal{H}p}$ of H-resources.

The values for $p_{ij}, s_{ijk}, r_{max}^{\mathcal{P}}, r_{max}^{\mathcal{S}}, r_{max}^{\mathcal{H}}, r_{ij}^{\mathcal{P}}, r_{ij}^{\mathcal{H}p}, r_{ijk}^{\mathcal{S}}, r_{ijk}^{\mathcal{H}s}$ are non-negative integers, satisfying: $r_{ij}^{\mathcal{P}} \leq r_{max}^{\mathcal{P}}, r_{ij}^{\mathcal{H}p} \leq r_{max}^{\mathcal{H}}, r_{ijk}^{\mathcal{S}} \leq r_{max}^{\mathcal{S}}, r_{ijk}^{\mathcal{H}s} \leq r_{max}^{\mathcal{H}}$

A complete instance of 4 jobs and 2 machines are presented. In the first place we show an optimal solution for UPMS and afterwards different types of resources are introduced. Then we found the optimal solution with the type of resources introduced. As is previously mentioned, all resources are renewable and needed during all processing and setup times.

Firstly in Fig. 1 we show the values of processing time p_{ij} and setup time s_{ijk} that we use in all these examples and the optimal solution for this UPMS problem is also obtained.

Notice that no relevant setups are fixed to high values (=7) and all diagonal is zero. It is usual to use diagonal to record the initial setup of a job when it exists, i.e. the setup necessary to do on a machine i before the first job j scheduled on this machine is processed.

In the rest of graphs in this section, the number just above jobs/setups in machine 1 and just below jobs/setups on machine 2 represents the resources needed.

We introduce in Fig. 2 the resources specific to process each job (P-resources). In this example, the value of $r_{max}^{\mathcal{P}}$ is fixed to 3. In Fig. 2 case (a), for time t=1 and t=2 the sum of resources is 4 and exceeds the maximum available resources that are 3 as is shown in the graph below graph (a) with the amount of resources used for each time. In case (b) the same solution (same assignment and sequence) is presented but without excess of use of resources, but with a maximum completion time of 8. However, with a new reassignment and a new sequence, optimal solution with resources is shown in (c), with C_{max} equal to 7.

Then, In the Fig. 2 and following figures, graph (a) represents the solution of UPMS with the resources recorder. These graphs a) have an excess of use of resources. Graph (b) shows the same solution (same assignment and sequence) being sure not to exceed the resources limit. Finally, graph (c) represents an optimal solution with these resources. These optimal solutions, in these examples, imply reassigning all jobs in spite of the new assignment having higher values of processing time for all jobs except J1.

For specific setup resources (S-resources), Fig. 3 shows the values of resources necessary to do the setups. The maximum available resources are fixed to 6 in this case ($r_{max}^{\mathcal{S}} = 6$). Note that the value of this maximum and the distribution of values of $r_{ijk}^{\mathcal{S}}$ are different than the previous case since the resources are completely different. The relevant values are highlighted in italics and the rest of values are fixed to 5.

In Fig. 3 case (a), for time t=4 and t=5 the sum of resources is 8 and, as is shown in the graph below it, this exceeds the maximum available resources that are 6. Graph (b) represents the same solution but without exceeding the maximum value and finally graph (c) represents the optimal solution with this type of resources. Notice that the optimal solution is not exactly the same as in the previous example.

For unspecific resources (H-resources) that can be used by both processing and setups, Fig. 4 shows the values of resources necessary to do the setups and processing. The maximum available resources for both are fixed to 9 in this case ($r_{max}^{\mathcal{H}} = 9$). The available resources are shared and this means that resources of this type can be used in both. Note that the value of the maximum and the distribution of values are different than previous cases since the resources are completely different. The used values are in bold and the rest of values are fixed to 8.

In Fig. 4 case (a), for time t=3 the sum of resources is 12 since J1 uses 6 shared resources (H-resources) for processing and for setup in machine 2, after doing J3 and before doing J4, it needs another 6 shared resources. Setups at t=4 and t=5 need a total of 12 H-resources. Finally, for t=6, processing jobs J2 and J4 need a sum of 12 H-resources. As is shown in the graph below it, in all these times the sum exceeds the maximum available resources that are 9. Graph (b) represents the same solution but without exceeding the maximum value and finally graph (c) with a $C_{max} = 9$ represents the optimal solution with this type of resource with a value of maximum completion time of 8.

Finally, the most complex type is when all these types of resources are present, i.e. specific for processing (P-resources), specific for setups (S-resources) and unspecific to be shared by both processing and setups (H-resources). In Fig. 5 we use the previous values used for all types of resources. Above machine 1 and below machine 2 appears a set of three values that represent in the first place the needs of specific resources for processing (P-resources). In second place and in italics the specific resources needed for setups (S-resources) and in third place and in bold the need of unspecific resources shared by both (H-resources). So, the three numbers above M1 and below M2 represent: P-resources/*S-resources*/**H-resources.**

In Fig. 5 case (a) we notice that for t=1 and t=2 the sum of the first values (P-resources) is 4 and this exceeds the maximum for this type of resource that is 3 ($r_{max}^{\mathcal{P}} = 3$). In second place in italics both values are zero since these represent the needs of specific resources for setup (S-resources) with $r_{max}^{\mathcal{S}} = 6$ and in these times machines are only processing jobs. In third place, the shared resources (H-resources) add up to 9 and does not exceed the maximum for this type of resource that is 9 ($r_{max}^{\mathcal{H}} = 9$). In t=3 processing of J1 in machine 1 is simultaneous with the setup on machine 2 after doing J3 and before starting J4 (s_{234}). In this time

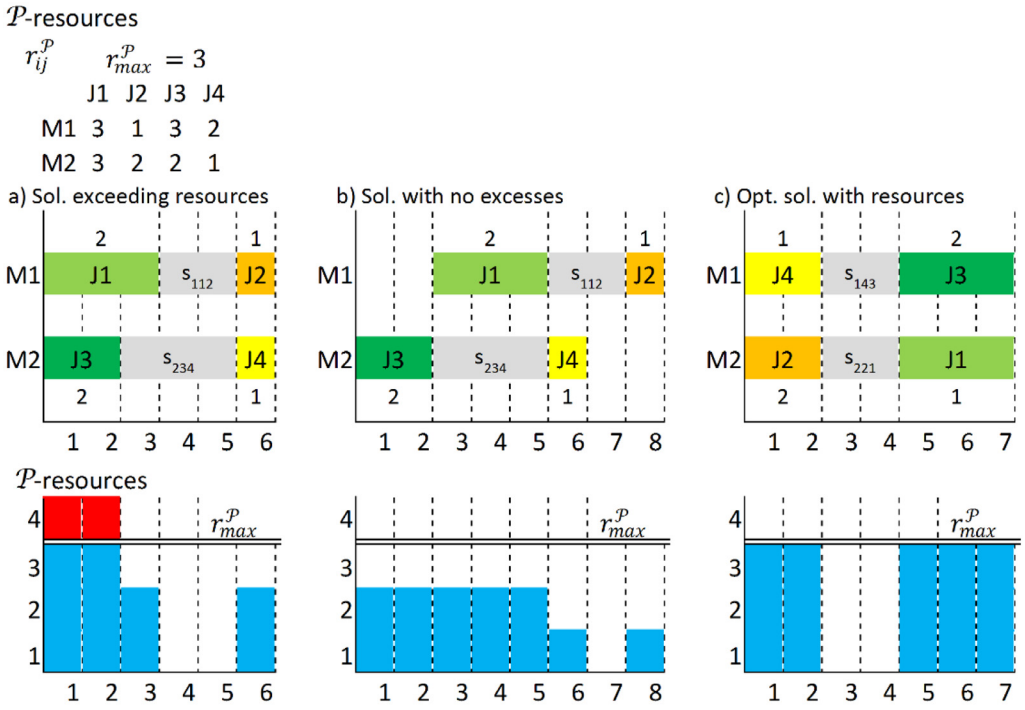

Fig. 2. Problem with specific processing resources.

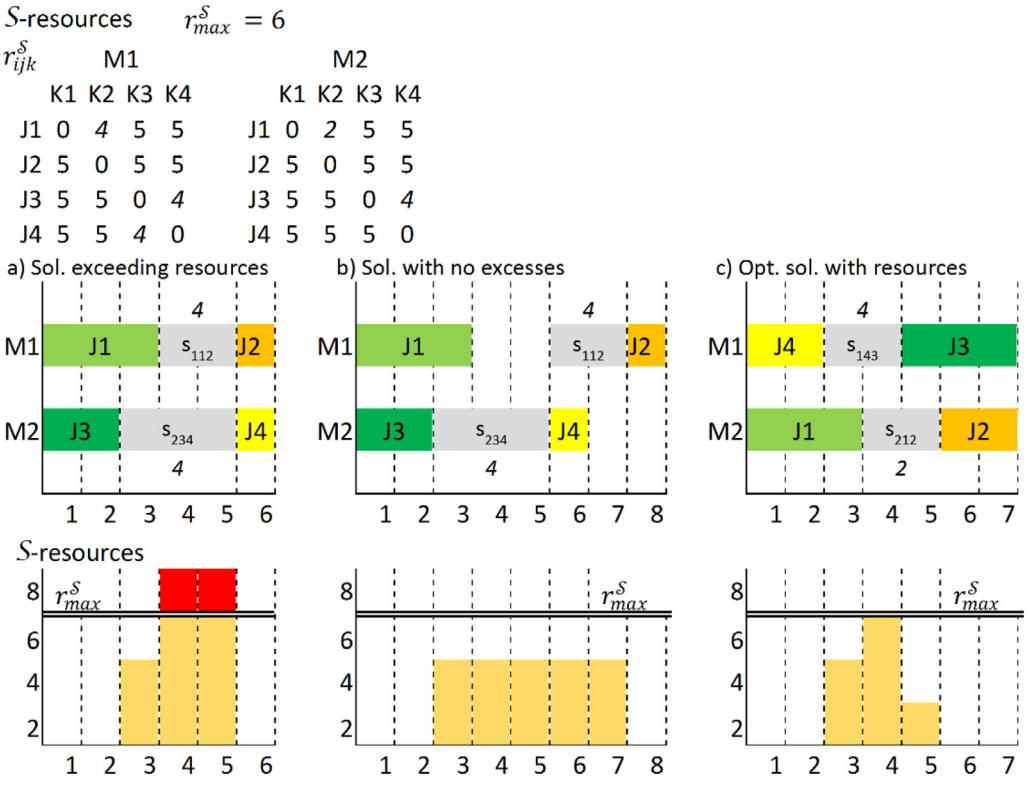

Fig. 3. Problem with specific setup resources.

t=3 the first term adds up to 2 (processing), the second term adds up to 4 (setups) and the third term adds up to 12 (shared). This last term exceeds the maximum for this type of resource (=9). For t=4 and t=5 the sums of the three terms are respectively 0, 8, 12 with the limits of 3, 6, 9. Finally for t=6 the sums are 2, 0, 12. In case (b) the same solution is represented taking into account not to exceed the different limits of types of resources with a $C_{max} = 11$ and finally case (c) shows the optimal solution with all types of resources with a $C_{max} = 9$.

Notice that all optimal solutions found for the different types of resources are different in these examples, showing that each problem could have different results.

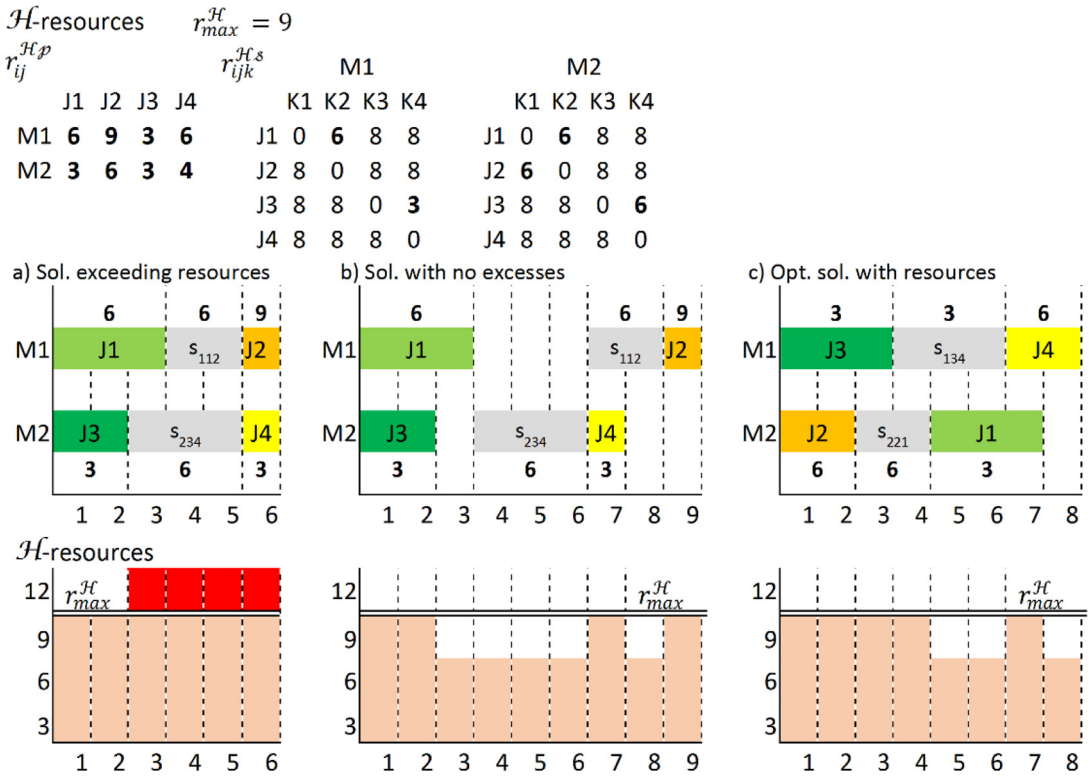

Fig. 4. Problem with unspecific resources shared by processing and setup.

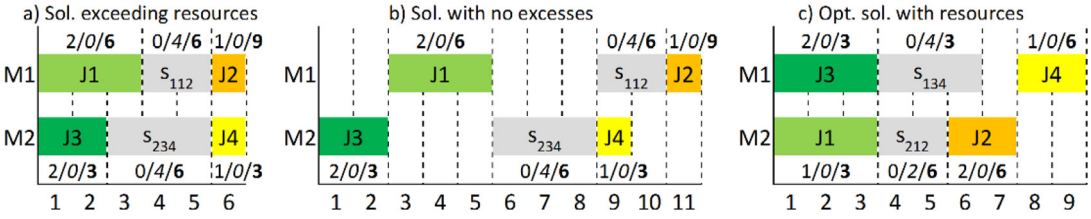

Fig. 5. Problem with specific processing resources, specific setup resources & unspecific resources used by both.

4. Model for UPM with setups and resources

In this section we formally describe the model used to solve the Unrelated Parallel Machine scheduling problem with Setups and Resources (UPMSR). As aforementioned, the UPMSR problem contains elements of the problem with only setups (and no resources) -UPMS- and elements of the problem with only resources (and no setups) -UPMR-. In particular, the MILP for the UPMS problem was widely studied and in a recent paper (Fanjul-Peyro, Ruiz, & Perea, 2019), the MILP used gives solutions up to 400 jobs and an exact method based on that MILP give solutions up to 1000 jobs and 8 machines with less than 0.8% of gap against Lower Bound. So, an adaptation of this MILP has been used in this paper. In the case of the UPMR problem, a packing-based model of Fanjul-Peyro, Perea, and Ruiz (2017) has been used recently. In this paper we improve and adapt this model.

In the UPMR problem, there are two dimensions, the completion times of processing jobs and the resources needed for the processing of the jobs. In Appendix A an example of this can be reviewed. Completion time of each job j is denoted by C_j which sets the point or coordinate on the time axis when job j is finished being processed (and consequently starts at $C_j - p_{ij}$). In Fig. 2 case (a), completion times of jobs are $C_1 = 3, C_2 = 6, C_3 = 2, C_4 = 6$. Similarly, for resources, $R^{\mathcal{P}}_j$ represents the point or coordinate on the P-resources axis where job j has its finish or upper limit of use of resources (and consequently its start or lower limit is $R^{\mathcal{P}}_j - r^{\mathcal{P}}_{ij}$). In Fig. 2 case (a), the coordinates of resources are $R^{\mathcal{P}}_1 = 2$, $R^{\mathcal{P}}_2 = 1$, $R^{\mathcal{P}}_3 = 4$, $R^{\mathcal{P}}_4 = 2$. However, for the complete UPMSR problem we need to take into account not only two factors but six: the processing times, the setup times, the P-resources, the S-resources, the H-resources used for job processing and the H-resources used for machine setups.

For formal definition of variables in this paper we use small letters to refer to data and capital letters to refer to variables. The variables used are the following:

- $X_{ijk} = 1$ if job k is processed immediately after job j on machine i, zero otherwise.
- $Y_{ik} = 1$ if job k is processed on machine i, zero otherwise.
- $C_j \geq 0$ is the completion time of job j (coordinate where processing of job j finishes in the time dimension).

- $B_j \geq 0$ is the completion time of the setup before the processing of job j on the corresponding machine (coordinate where setup of job j finishes in the time dimension).
- $R_j^{\mathcal{P}} \in [0,\ r_{max}^{\mathcal{P}}]$ is the coordinate of processing of job j in the P-resources dimension (the specific resources employed in the processing of job j).
- $R_j^{\mathcal{S}} \in [0,\ r_{max}^{\mathcal{S}}]$ is the coordinate of setup of job j in the S-resources dimension (the specific resources employed in the setup before processing of job j).
- $R_j^{\mathcal{H}p} \in [0,\ r_{max}^{\mathcal{H}}]$ is the coordinate of processing of job j in the H-resources dimension (the unspecific shared resources employed in the processing of job j).
- $R_j^{\mathcal{H}s} \in [0,\ r_{max}^{\mathcal{H}}]$ is the coordinate of setup of job j in the H-resources dimension (the unspecific shared resources employed in the setup before processing of job j).
- $D_{jk}^{\mathcal{P}}, E_{jk}^{\mathcal{P}}, F_{jk}^{\mathcal{S}}, G_{jk}^{\mathcal{S}}, U_{jk}^{\mathcal{H}p}, V_{jk}^{\mathcal{H}p}, U_{jk}^{\mathcal{H}s}, V_{jk}^{\mathcal{H}s}$ are binary auxiliary variables with value = 1 when the variable value studied in constraints (time or resources) of job k is higher than the variable value studied (time or resources) of job j. The letters in superscript are used to indicate the dimension where they are used. A is a sufficiently large constant.

We describe the complete model of UPMSR analyzing it in parts. In the first part we introduce the structure of UPMS, in second place we show the completion time limits for setups and processing and finally the constraints for resources are presented.

The model for UPMSR consists of:

- UPMS problem structure

$$\min C_{max} \tag{1}$$

$$\sum_{j\in N_0, k\in N, k\neq j} s_{ijk}X_{ijk} + \sum_{k\in N} p_{ik}Y_{ik} \ \leq C_{max},\ \ i \in M \tag{2}$$

$$\sum_{k\in N} X_{i0k} \ \leq 1,\ \ i \in M \tag{3}$$

$$\sum_{i\in M} Y_{ik} \ = 1,\ \ k \in N \tag{4}$$

$$Y_{ik} = \sum_{j\in N_0, j\neq k} X_{ikj},\ \ i \in M,\ \ k \in N \tag{5}$$

$$Y_{ik} = \sum_{j\in N_0, j\neq k} X_{ijk},\ \ i \in M,\ \ k \in N \tag{6}$$

$$C_k - C_j + A\left(1 - X_{ijk}\right) \geq s_{ijk} + p_{ik},\ \ i \in M, j \in N_0,\ \ k \in N,\ \ j \neq k \tag{7}$$

$$C_k \leq C_{max},\ \ k \in N \tag{8}$$

$$X_{ijk} \in \{0,\ 1\},\ \ \ Y_{ij} \geq 0,\ \ C_k \geq 0,\ \ C_0 \ = 0 \tag{9}$$

In this model the Eq. (1) shows the objective. Constraints (2) define the makespan where all setup times and processing times are added for each machine. In constraints (3) state that no more than one job is scheduled after the dummy job at the beginning of each machine. Constraints (4) force each job to be assigned to only one machine. Constraints (5) and (6) ensure that only one job be done immediately before and only one job immediately after (respectively) each job if jobs are assigned to the same machine, including dummy jobs. Constraints (7) break subtours and give the right scheduling order, because if k is processed after j on machine i, it forces finishing the processing of job k not earlier than $s_{ijk} + p_{ik}$. Constraints (8) establish the relationship between C_{max} and completion times of each job. Finally, (9) set the limits of variables and set the completion time of the dummy jobs to zero. Notice that Y_{ik} can be relaxed to positive variable instead of binary since (5) and (6) are a sum of binary variables and (4) make its sum be equal to 1. The variations introduced in Fanjul-Peyro, Ruiz, and Perea (2019) that improve the UPMS model, adapted from TSP problem and based on scheduling order of jobs, cannot apply in UPMSR since idle times can be present and therefore we need to know finishing times and not only the schedule.

- Completion times for setup and processing

$$B_k \leq C_k - \sum_{i\in M} Y_{ik}p_{ik},\quad \forall k \in N, \tag{10}$$

$$B_k - C_j + A\left(1 - \sum_{i\in M} X_{ijk}\right) \geq \sum_{i\in M} s_{ijk}X_{ijk},\quad \forall j \in N_0, k \in N, j \neq k \tag{11}$$

$$C_k \geq \sum_{i\in M, j\in N_0} s_{ijk}X_{ijk} + \sum_{i\in M} p_{ik}Y_{ik},\ \ k \in N \tag{12}$$

Constraints (10) ensure that setup finishing time of job k is earlier than processing starting time of job k. Constraints (11) force that, if job j is the previous job done before job k on machine i, the finishing time of setup k must be later than the finishing time of processing of job j plus setup time of job k. Constraints (12) set the lower limit of completion time of job k (C_k) to be greater or equal than the sum of its setup plus its processing time. Upper limits for C_k have been provided previously by (8) which set this limit below or equal to C_{max}.

- Limits of resources

$$R_{max}^{\mathcal{P}} \geq R_k^{\mathcal{P}} \geq \sum_{i\in M} Y_{ik}r_{ik}^{\mathcal{P}},\ \ k \in N \tag{13}$$

$$R_{max}^{\mathcal{S}} \geq R_k^{\mathcal{S}} \geq \sum_{i\in M, l\in N} X_{ilk}r_{ilk}^{\mathcal{S}},\ \ k \in N \tag{14}$$

$$R_{max}^{\mathcal{H}} \geq R_k^{\mathcal{H}p} \geq \sum_{i\in M} Y_{ik}r_{ik}^{\mathcal{H}p}, R_{max}^{\mathcal{H}} \geq R_k^{\mathcal{H}s} \geq \sum_{i\in M, l\in N_0} X_{ilk}r_{ilk}^{\mathcal{H}s}, k \in N \tag{15}$$

Constraints (13) and (14) set the coordinate limits of specific resources used in processing ($R_k^{\mathcal{P}}$) and used specifically in setups ($R_k^{\mathcal{S}}$). Constraints (15) are the same for unspecific shared resources. Notice that the maximum shared resources available ($R_{max}^{\mathcal{H}}$) is the same for shared resources used in processing or shared resources used in setups.

A new model of UPMR, simpler and more efficient than previously published, is shown in Appendix A. We adapted it to UPMSR problem.

- Specific resources in processing (P-resources)

$$C_k - C_j + A(1 - D_{jk}^{\mathcal{P}}) \geq \sum_{i\in M} Y_{ik}p_{ik},\quad \forall j,\ \ k \tag{16}$$

$$R_k^{\mathcal{P}} - R_j^{\mathcal{P}} + A\left(1 - E_{jk}^{\mathcal{P}}\right) \geq \sum_{i\in M} Y_{ik}r_{ik}^{\mathcal{P}},\quad \forall j,\ \ k \tag{17}$$

$$D_{jk}^{\mathcal{P}} + D_{kj}^{\mathcal{P}} + E_{jk}^{\mathcal{P}} + E_{kj}^{\mathcal{P}} \geq 1,\quad \forall j, k > j \tag{18}$$

Constraints (16) show that, if C_k is later than C_j, the difference between these completion times is greater than or equal to the processing time of job k. As an example, binary auxiliary variable $D_{jk}^{\mathcal{P}}$ has value 1 when this occurs and zero otherwise. In (17), if $R_k^{\mathcal{P}}$ is greater than $R_j^{\mathcal{P}}$, the difference between coordinates of these resource points is greater than or equal to the resources used in the job k processing. Finally, (18) force at least one of the binary variables to have the value of 1. Since the constraints are defined for all j and k, this implies that, at least: or j precedes k without overlapping its processing on the time axis, or k precedes j without overlapping its processing on the time axis, or the j resource coordinate is higher than k without overlapping its processing resources on the P-resources axis, or the k resource coordinate is higher than j without overlapping its processing resources on the P-resources axis.

- Specific resources in setups (S-resources)

$$B_k - B_j + A\left(1 - F_{jk}^{\mathcal{S}}\right) \geq \sum_{i\in M, l\in N_0} X_{ilk}s_{ilk},\quad \forall j,\ \ k \tag{19}$$

$$R_k^s - R_j^s + A\left(1 - G_{jk}^{\mathcal{S}}\right) \geq \sum_{i \in M, l \in N_0} X_{ilk} r_{ilk}^{\mathcal{S}}, \quad \forall j,\ k \tag{20}$$

$$F_{jk}^{\mathcal{S}} + F_{kj}^{\mathcal{S}} + G_{jk}^{\mathcal{S}} + G_{kj}^{\mathcal{S}} \geq 1, \quad \forall j, k > j \tag{21}$$

Constraints (19) show that, if B_k is later than B_j, the difference between these setup finishing times is greater than or equal to the setup time of job k. As an example, binary auxiliary variable $F_{jk}^{\mathcal{S}}$ has value 1 when this occurs and zero otherwise. In (20), if $R_k^{\mathcal{S}}$ is greater than $R_j^{\mathcal{S}}$, the difference between coordinates of these resource points is greater than or equal to the resources used in the job k setup. Finally, (21) force at least one of the binary variables to have the value of 1. Since the constraints are defined for all j and k, this implies that, at least: or j precedes k without overlapping its setups on the time axis, or k precedes j without overlapping its setups on the time axis, or the j resource coordinate is higher than k without overlapping its setup resources on the S-resources axis, or the k resource coordinate is higher than j without overlapping its setup resources on the S-resources axis.

- Unspecific resources shared by both processing and setup

When considering shared resources, it is necessary not to overlap unspecific shared processing resources among themselves nor overlap unspecific shared setup resources among themselves. The unspecific shared processing resources should also not overlap with the unspecific shared setup resources. For shared resources in the processing/processing case, in the previous constraints to avoid this overlap (16), (17), and (18), it is necessary to change superscript P for superscript Hp. Similarly, in the setup/setup case, in the previous constraints to avoid overlap (19), (20) and (21), it is necessary to change superscript S for superscript Hs. Finally, in the processing/setup case, order is important and its inverse, setup/processing, should be considered together as is shown in the following constraints:

$$C_k - B_j + A(1 - U_{jk}^{\mathcal{H}p}) \geq \sum_{i \in M} Y_{ik} p_{ik}, \quad \forall j,\ k \tag{22}$$

$$B_j - C_k + A\left(1 - U_{kj}^{\mathcal{H}s}\right) \geq \sum_{i \in M, l \in N_0} X_{ilj} s_{ilj}, \quad \forall j, k \tag{23}$$

$$R_k^{\mathcal{H}p} - R_j^{\mathcal{H}s} + A\left(1 - V_{jk}^{\mathcal{H}p}\right) \geq \sum_{i \in M} Y_{ik} r_{ik}^{\mathcal{H}p}, \quad \forall j,\ k \tag{24}$$

$$R_j^{\mathcal{H}s} - R_k^{\mathcal{H}p} + A\left(1 - V_{kj}^{\mathcal{H}s}\right) \geq \sum_{i \in M, l \in N_0} X_{ilj} r_{ilj}^{\mathcal{H}s}, \quad \forall j, k \tag{25}$$

$$U_{jk}^{\mathcal{H}p} + U_{kj}^{\mathcal{H}s} + V_{jk}^{\mathcal{H}p} + V_{kj}^{\mathcal{H}s} \geq 1, \quad \forall j, k \tag{26}$$

In constraints (22) we show the case when the processing finishing time of job k is later than the setup finishing time of job j. Therefore, the difference between finishing times must be equal to or greater than the processing time of job k. In this case, as an example, the binary auxiliary variable $U_{jk}^{\mathcal{H}p}$ has value 1 when setup of job j is done before processing job k and zero otherwise. In (23) we find the inverse case, where setup finishing time of job j is later than processing finishing time of job k. So, the difference between finishing times must be equal to or greater than the setup time of job j. Eqs. (24) and (25) are similar constraints to the two previous constraints but in this case using shared resources. Finally, (26) force at least one of the binary variables to have the value of 1.

The model presented for UPMSR in Eqs. (1)–(26) can be reduced to other problems previously studied when setups or resources are not present, such as UPM, UPMS or UPMR. On the other hand, when setups and different types of resources are present, seven combinations of them can generate seven different problems:

- UPMSR-P: UPMSR with only specific processing resources.
- UPMSR-S: UPMSR with only specific setup resources.
- UPMSR-H: UPMSR with only unspecific shared resources.
- UPMSR-P+S: UPMSR with specific processing resources and specific setup resources.
- UPMSR-P+H: UPMSR with specific processing resources and unspecific shared resources.
- UPMSR-S+H: UPMSR with specific setup resources and unspecific shared resources.
- UPMSR-P+S+H: UPMSR with specific processing resources, specific setup resources and unspecific shared resources.

The MILP model for UPMSR is modeled by constraints (1)–(26). However, from (13) to the end, depending on the resources present in each case, not all constraints could be necessary. A deeper study where we formalize all these different problems is included in Appendix A.

Although it is not studied in this paper, extending the problem to more than one resource of each type is as simple as adding a new set of constraints of the corresponding type, e.g. if two different specific processing resources are included, it would be enough to add a new set of constraints (16), (17), and (18) with their corresponding values. Other models based on time are studied but obtain worse results as is shown is Appendix A.

5. Three phase exact algorithm (TPhA)

To solve UPMSR problem we need to find: the assignment for each job on a machine, the sequencing of the set of jobs on each machine and the timing of each processing and setup of each job. So, we could divide the problem into three parts: assignment, sequencing and timing. As Tran, Araujo, and Beck (2016) and Fanjul-Peyro Ruiz, and Perea (2019) showed, the UPMS problem could be solved dividing into two phases: the previous assignment and the subsequent sequencing. When resources are introduced in problem, a timing of setups and processing should also be determined in a posterior phase. An exact algorithm is developed to solve UPMSR in three phases.

5.1. Pseudocode algorithm

For a better understanding of the algorithm, we present a pseudocode that represents it. In the following sections details of each phase are developed. In line 1 of the pseudocode we set the finish variable (*STOP*) to false, the best value found (*BestVal*) -or best C_{max} - to infinite, the Lower Bound (*LB*) to zero and the remaining time (*TimeLeft*) to the initial available time (*TotalTime*). In Line 2 a loop is made as long as the finish variable (*STOP*) does not change or there is some time left. Line 3 is a label to indicate the assignment phase, developed in line 4 with *Master MILP* using Eqs. (1)–(6) as Section 5.2 explains. Line 5 indicates the *Master* solution Sol^M with its values of C_{max} and Y_{ik} obtained (C_{max}^M, Y^M). Line 6 checks if *Master* C_{max} (C_{max}^M) is lower than the best solution found previously; if not, the sequencing and timing phase should be done to determine a complete solution. Line 7 checks if the solution of *Master* is an optimal solution for *Master* MILP. If it is optimal, we need to do the timing and sequencing phase together in order to obtain a complete optimal solution for *Master* assignment and introduce a cut to avoid this local optimum as explained in Section 5.4. When the *Master* solution is not optimal, it is not necessary to do both phases together and the algorithm gets better results by doing the sequencing and timing phase separately. In this case, as the label in Line 8 expresses, the sequencing phase is done with its MILP (Line 10) obtaining the sequencing solution Sol^S with its particular values for $C_{max},\ Y_{ik}, X_{ijk}$ (C_{max}^S, Y^M, X^S), explained in Section 5.3.

Algorithm 1: Pseudocode of Three Phase Algorithm for UPMSR

```
1  Set STOP = False, BestVal = +∞, LB = 0, TimeLeft = TotalTime
2  while STOP = False and Timeleft > 0 do
3      #Phase 1: assignment
4      Find next Master solution: MILP (1) to (6) with Y_ik binary and X_ijk integer
                  (first iteration until GAP ≤ 2% or 0.9 TotalTime)
5      Let Sol^M(C^M_max, Y^M) be the Master solution found
6      if C^M_max < BestVal then
7          if Sol^M ≠ optimal then
8              #Phase 2: sequencing
9              Update TimeLeft
10             Solve Sequencing(Y^M): MILP (1) to (9) with Y_ik = Y^M
11             Let Sol^S(C^S_max, Y^M, X^S) be the Sequencing solution found
12             #Phase 3a: timing
13             Update TimeLeft
14             Solve Temporality3a(Y^M, X^S): CP (27) to (37) with Y_ik = Y^M and X_ijk = X^S
15             Let Sol^Ta(C^Ta_max) be the Temporality3a solution found
16             BestVal = min {BestVal, C^Ta_max}
17         else
18             #Phase 3b: timing + sequencing
19             Update TimeLeft
20             Solve Temporality3b(Y^M): MILP (1) to (26) with Y_ik = Y^M
21             Let Sol^Tb(C^Tb_max) be the Temporality3b solution found
22             BestVal = min {BestVal, C^Tb_max}
23             Add to Master → CUT(C^Tb_max, Y^M): (38) with C^q_max = C^Tb_max and Y_ik = Y^M
24             LB = max {LB, C^M_max}
25         end
26     else
27         if Sol^M = optimal then STOP = True, global optimal solution found
28     end
29     Update TimeLeft
30 end
```

After sequencing, the timing phase begins (label in Line 12). In this case we use a Constraint Program (CP) model explained in Section 5.4. The CP model is more efficient than the MILP model when only the resources should be set, as tests prove shown in Appendix A. We call this phase *Temporality3a*, and its corresponding solution and C_{max} are named Sol^{Ta} and C^{Ta}_{max} (Line 15). In the case that the *Master* solution is optimal, as aforementioned, we need to do the sequencing and timing phase together, as the label in Line 18 indicates. For this case, the complete MILP is used. We denoted this phase *Temporality3b* and its corresponding solution and C_{max} are named Sol^{Tb} and C^{Tb}_{max} (Line 21). These results can be introduced in the Cut Eq. (38) as shown in Line 23 (e.g. the value of C^{Tb}_{max} is used as the C^{q}_{max} value in the Cut Equation – see Section 5.4). Now this cut, can be introduced in the *Master* model to avoid a local optimum. Finally, if *Master* C_{max} (C^{M}_{max}) is not lower than the best solution found previously (Line 6), it can be equal or higher (Line 26). In the case of higher values of *Master* C_{max}, the algorithm continues searching for a new *Master* solution. In the case that *Master* C_{max} is equal to the best solution found and if the *Master* solution is optimal (Line 27), this implies that the master optimal makespan matches with the best feasible solution stored. So, the global optimal solution for the complete problem has been found.

5.2. Phase 1: assignment

In the assignment phase we are going to set the variables Y_{ik} which determine the machine i where job k is assigned. In order to do this, we use the constraints (1)–(6) and relax the variables X_{ijk} as nonnegative continues variables while making Y_{ik} binary. This reduced MILP is denoted as *Master*. In the first iteration of Master we let a gap of 2% as in paper Tran, Araujo and Beck (2016) or 90% of time limit as in paper Fanjul-Peyro, Ruiz, and Perea (2019) in order to have time to find a feasible first solution in successive phases. Later, after finishing the rest of phases, the algorithm returns to phase 1 to obtain a new Master solution. However, master solution is only an assignment solution since cycles could be present until Eqs. (7)–(9) are introduced in the next phase. Notice that an optimal solution of Master is always a lower bound for a complete problem. This is not true for the rest of solutions obtained in other phases as we show later.

5.3. Phase 2: sequencing

In the second phase we set the sequencing. With the variables Y_{ik} fixed with values obtained in phase one, we now use the constraints (1)–(9). We obtain as result the value of variables X_{ijk} which give us a sequencing of assigned jobs on their machine. The solution is not a feasible solution since no resources have been introduced yet. Moreover, the solution found in this phase is not a valid lower bound for a complete problem. In a new example of 5 jobs and 2 machines we are going to suppose that the processing and setup times are: $p_{11} = 2$, $p_{12} = 1$, $p_{23} = 2$, $p_{24} = 1$, $p_{15} = 1$, $p_{25} = 2$, $s_{112} = 1$, $s_{234} = 1$, $s_{125} = 3$, $s_{245} = 1$, $s_{121} = 3$, and others values are so large that they will not be considered, e.g. =10 (and initial setups are zero). In Fig. 6 case (a) we can see an optimal solution from Master -only assignment -; in case (b), when setups are introduced in this previous assignment, gives a value of 8 for C_{max}. It cannot be considered a valid lower bound since, as is shown in case (c), optimal solution has a value of 7. Notice that job 5 should be assigned to machine 1 (processing time of 1) in the case of Master (no setups present), case (a). But when setups are taken into account along with processing times – case (c), job 5 has to be assigned to machine 2 even if its processing time in this machine is 2 because setups for this job change from machine 1 to machine 2.

5.4. Phase 3: timing

In phase 3 a feasible solution is searched for. Assignment variables Y_{ik} are fixed and sequencing variables X_{ijk} are found. To find a feasible solution we now used a whole model in order to introduce resources. Setting these variables makes the MILP easier to solve but resource constraints are still large. Moreover, when the assignment phase obtains an optimal master solution, the complete feasible solution needs to be an optimal solution for this assignment in order to introduce a cut that allows the algorithm to get out of local optimal solution. It implies that the sequencing phase should be skipped when in the assignment phase (Master) we obtain an optimal solution. In this case, a whole MILP is used with only variables Y_{ik} fixed. As an example of this is Fig. 7, which uses the same values as Fig. 6 but without job 5 and with $R_{max} = 3$. Graph (a) shows an optimal solution from Master and with setups but there is a lack of resources at time 1 and 2 since the sum of resources used in the processing of jobs 1 and 3 exceeds the maximum resources. In (b), the solution is feasible taking into account the resources, with makespan of 6. However, it is not the optimal solution for this assignment as graph (c) shows. A different sequence – but the same assignment – on machine 1 gives an optimal makespan of 5. So, in order to introduce a cut that establishes a lower bound for a particular assignment, when phase 1 gives an optimal master solution, we do the sequencing and timing phases together.

As a result of the previous discussion, we can discriminate two cases in the timing phase: one when master solutions are not optimal and one when they are. In the first case, when the solution from Master is not optimal, the sequencing phase is done before the timing phase is carried out with X_{ijk} variables fixed by the sequencing phase. From now on we will refer to this first case as phase *3a*. In the second case, when the solution from Master is optimal, as Fig. 7 shows, we need to execute a whole MILP with only Y_{ik} variables fixed. We denoted this phase as *3b*. For the timing phase, a Constraint Programing (CP) could be used instead of MILP with solvers. This also is used by Fleszar and Hindi (2018) in the case of UPMR problem. As we later show in tests, CP performs

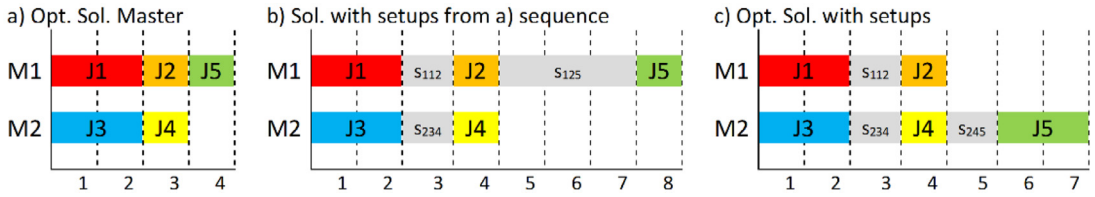

Fig. 6. Opt. Master Sol. with setups introduced, is not a valid LB.

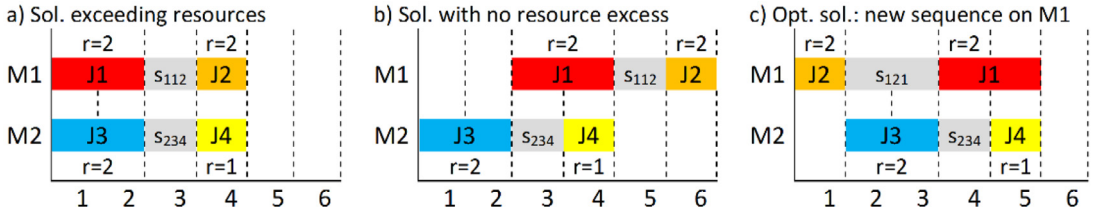

Fig. 7. Master Opt. Sol. implies doing sequencing & timing phases together.

well in the case 3a, when it is not an optimal master solution and it is only necessary to set the resources in the solution. However, in the 3b case (optimal master solution) we need to do sequencing and timing together. The performance of CP decreases and does not reach the results of MILP. The low performance is due in part to the fact that CP finds it difficult to solve UPMS problem, compared with MILP. Nevertheless, CP shows very good behavior when only resources need to be set.

To codify CP we need the following data and variables:

- Y^*_{ik} are the variables set in the assignment phase each with its value p_{ik}.
- X^*_{ijk} are the variables set in the sequencing phase each with its value s_{ijk}.
- $r^{\mathcal{P}*}_{ik}$ are the values of specific processing resources assigned.
- $r^{\mathcal{S}*}_{ijk}$ are the values of specific setup resources sequenced.
- $r^{\mathcal{H}p*}_{ik}$ are the values of assigned shared resources for processing.
- $r^{\mathcal{H}s*}_{ijk}$ are the values of assigned shared resources for setups.
- *pulse* represents a cumulative function of CP.

The CP used in phase 3a (no optimal solution from Master) is:

$$intervalVar\ Y^*_{ik},\ X^*_{ijk} \tag{27}$$

$$rp\_Y^*_{ik} = pulse\left(Y^*_{ik},\ r^{\mathcal{P}*}_{ik}\right) \tag{28}$$

$$rs\_X^*_{ijk} = pulse\left(X^*_{ijk},\ r^{\mathcal{S}*}_{ijk}\right) \tag{29}$$

$$rhp\_Y^*_{ik} = pulse\left(Y^*_{ik}, r^{\mathcal{H}p*}_{ik}\right) \tag{30}$$

$$rhs\_X^*_{ijk} = pulse\left(X^*_{ijk}, r^{\mathcal{H}s*}_{ijk}\right) \tag{31}$$

$$minimize\left(\max\left[endOf\left(Y^*_{ik}\right)\right]\right) \tag{32}$$

$$endBeforeStart\left(X^*_{ijk}, Y^*_{ik}\right) \tag{33}$$

$$endBeforeStart\left(Y^*_{ik}, X^*_{ikj}\right) \tag{34}$$

$$R^p_{\max} \geq \sum rp\_Y^*_{ik} \tag{35}$$

$$R^s_{\max} \geq \sum rs\_X^*_{ijk} \tag{36}$$

$$R^h_{\max} \geq \sum rhp\_Y^*_{ik} + \sum rhs\_X^*_{ijk} \tag{37}$$

In (27) variables are defined. Notice that they are fixed and will come from phase of assignment and sequencing. From (28)–(31) resources used by processing and setup are defined. In (32) the objective is shown as the minimization of the maximum of when each processing job ends. Eq. (33) forces that after each setup its corresponding job should be done. Eq. (34) forces that after each job a new setup should be done except for the last job k in each machine which has no X^*_{ikj}. Notice that the subscript in this case is ikj instead of ijk. Eqs. (35)–(37) are proposed to avoid the use of more resources at the same time than maximum resources available of each type.

On the other hand, in the case that the master solution is optimal, and after solving timing and sequencing together (3b), it is necessary to introduce a cut in Master in order to avoid this local optimal solution. Since idle times could be present, in (38) a new cut is introduced.

$$C_{max} \geq C^q_{max}\left(\sum_{i\in M, k\in N^q_i} Y_{ik} - n + 1\right) \tag{38}$$

In (38) is forced that when in iteration q the value of the feasible solution's makespan is C^q_{max} then this value is a lower bound for C_{max} in the Master problem as long as the assignment is the same as in iteration q (N^q_i). Notice that, if all assignments are the same, the value in brackets is 1 and if only one assignment changes, the value will be zero (less than zero if more than one changes its assignment). A cut used in Tran, Araujo, and (Beck, 2016) for UPMS cannot be used in UPMSR problem since idle times could be present.

6. Computational experiments

We show in this section the result obtained in a computational campaign. Small, medium and large instances are used. The solvers used were Gurobi 7.0 and IBM ILOG CP and CPLEX 12.7. We primarily used Gurobi to solve MILP because of its superior performance but a comparison with the use of CPLEX instead will also be shown. All experiments were carried out on virtual machines with 2 cores and 16 GB of RAM with Windows 10 64 bits as operating system. Virtual machines come from an Openstack virtualization platform running on 12 blades with 12 cores AMD Opteron Abu Dhabi 6344 processors 2.6 GHz and 256 GB of RAM each one.

The experiments were randomly distributed on virtual machines in order to speed up the completion of all experiments without parallel computing. The programming language was C#.NET of Microsoft Visual Studio 2015 IDE.

The main indicator used is the gap between Lower Bound and C^*_{max} obtained by different models or exact methods expressed in percentage as $100(C^*_{max}/LB - 1)$.

In small instances, we set the time limit in 3 hours for the model and the Three Phase Algorithm (TPhA) in order to give them the same time as Tran, Araujo, and Beck (2016). For medium and large instances used with TPhA, 30 min was fixed as the time limit. However, the behavior of the Three Phase Algorithm with time is also studied in Appendix A.

6.1. Instances

For the instances, we have to consider several factors such as number of jobs, number of machines, magnitude and dispersion of the processing times and setup times, amount of resources available and their consumption by jobs. So, to introduce the UPMSR problem, we based the instances used in this paper on those present among the literature for UPMS and UPMR problems.

In this paper the distribution times for processing and setups use the same as Tran, Araujo, and Beck (2016) and Fanjul-Peyro, Ruiz, and Perea (2019), previously introduced in Arnaout, Rabadi, and Musa (2010) with small, medium and large instances. With a processing distribution of *U(50,100)* and setup distribution of *U(50,100)*, small instances were randomly generated with 10, 20, 30, 40, 50 jobs and 4, 6, 8 machines. Initial setup is used in all instances.

For setup times, the triangular inequality must be satisfied i.e. $s_{ijk} \leq s_{ijl} + p_{il} + s_{ilk},\ i \in m;\ j,k,l \in n$ to avoid situations when doing a third job between a pair of jobs uses less time than only doing the original pair of jobs.

For resources, following Fanjul-Peyro, Perea, and Ruiz (2017) or Fleszar and Hindi (2018), a distribution of $U(1, 9)$ is applied for assigned resources. Resources could be specific for Processing (P), specific for Setups (S) or sHared by both, processing and setups (H) and all combinations of them i.e. P+S, P+H, S+H, P+S+H. Total: seven types of resources. The maximum number of resources has been determined so that all instances can be feasible. Since resources required has a value between 1 and 9 the $R_{max}$is:

$$R_{max} = m\frac{r_{max} + r_{min}}{2} = m\frac{9+1}{2} = 5m \tag{39}$$

Five replicates are made for each combination. So, the total number of small instances are $5{\times}3{\times}7{\times}5 = 525$.

For medium instances the number of jobs used is 60, 80, 100, 120, and 140 with 4, 6, and 8 machines, for a total of $5{\times}3{\times}7{\times}5 = 525$ medium instances. Finally, we generate a similar number of instances with 200, 250, 300, 350, and 400 jobs, and 4, 6, and 8 machines for testing a total of $5{\times}3{\times}5{\times}7 = 525$ large instances. A total of 1575 instances are studied.

However, in Appendix A, different distribution times for setups and processing are studied as well as calibration tests.

The executable to generate instances, the executable to run the model or the Three Phase Algorithm and all tables are included in on line materials.

6.2. Results and discussion

To know the values range obtained by the results in this paper we will previously show a comparative in Table 1 of values of the easier case of UPMR problem (with no setups). In this table we compare the results obtained by MILP proposed in the UPMR paper Fanjul-Peyro, Perea, & Ruiz, 2017) with the new model for UPMR exposed in (40)–(47) and our final model of UPMSR. We use in Table 1 the same medium instances tested and the same time limit (an hour) that is used in the aforementioned paper.

Table 1
comparative of MILP for UPMR.

	gap	RPD	%Opt	%RPD=0
UPMR-pre model	385.99	16.57	0.22	3.56
UPMR-new model	173.85	7.58	3.11	12.22
UPMSR-model	7.17	6.70	34.22	36.22

Table 2
Gap & % of optimal solution of Model and TPhA with small instances. In type cases with sHared (H) resources higher than 30 jobs are omitted since some no feasible solutions are found. Time limit = 180-min.

		Model		TPhA	
Type	jobs	Gap	%Opt	Gap	Opt%
P	10-50	3.60	32.00	5.66	18.67
S	10-50	3.62	33.33	4.73	25.33
P+S	10-50	5.31	32.00	6.14	20.00
H	10-30	6.09	37.78	5.96	37.78
P+H	10-30	9.12	33.33	6.44	28,89
S+H	10-30	16.47	33.33	9.61	22,22
P+S+H	10-30	18.35	24.44	10.07	11.11
Avg.		*8.94*	*32.32*	*6.94*	*23.43*

In Table 1 "UPMR-pre model" represents the MILP published in Fanjul-Peyro, Perea, and Ruiz (2017). The MILP expressed in Eqs (40)–(47) is denoted as "UPMR-new model". Finally, "UPMSR-model" is how we named the MILP used in this paper to solve instances with setups and resources but applied to this case, where all setups are zero. Gap is the difference between solution and LB found by each MILP. RPD (relative percentage difference) shows the difference between solutions found for each MILP and the best LB found by the three models. As we can see, UPMR-new improves the UPMR-pre results and our UPMSR MILP applied to the UPMR problem obtains the best results. Notice that the UPMR problem is easier than the UPMSR problem studied in the present paper since no setups are present. However, gap of model for the UPMR problem have similar values to values obtained for the UPMSR problem as we can see in the next results. Remember all of these results are only to compare MILP in UPMR and not for other exact methods as in Fleszar and Hindi (2018) which improves these results. However this study allowed us to know which MILP of UPMR was better to adapt to the UPMSR problem.

Now, in order to know the performance of the new MILP model introduced for the UPMSR problem, we use the small instances. For more complex types of resources i.e. when shared resources are present, the model returns no feasible solutions for some instances larger than 30 jobs, even with three hours of execution. Table 2 shows the results for the MILP model and the Three Phase Algorithm (TPhA) for small instances with three hours as the time limit.

In Table 2 the heading jobs represents the interval of number of jobs, 10–50 for resources needed in Processing (P), in Setup (S), and in Processing and Setup (P+S) types. Only jobs from 10 to 30 are shown in the case of the shared resources type (H) being present since the model returns some no feasible solutions for instances higher than 30 in these cases. The different type of problems with shared resources included are those that need this resource type: only sHared (H), processing and shared (P+H), setups and shared (S+H), process, setup and shared (P+S+H). The percentage of gap (difference between solution obtained and LB) and percentage of optimal solution found are represented. As can be appreciated, the average gap of the Three Phase Algorithm (6.94)

Table 3
Gap of TPhA with medium instances. Time limit = 30 min.

Type/jobs	60	80	100	120	140	Avg.
P	5.84	4.85	4.18	3.59	3.48	**4.39**
S	5.31	3.95	4.01	3.10	2.95	**3.86**
P+S	5.83	4.71	4.39	3.81	3.62	**4.47**
H	6.19	4.61	4.39	3.83	3.73	**4.55**
P+H	6.75	5.02	5.03	4.05	4.32	**5.03**
S+H	14.37	13.26	12.75	12.06	11.87	**12.86**
P+S+H	13.69	13.15	13.09	13.57	13.47	**13.39**
Avg.	*8.28*	*7.08*	*6.83*	*6.29*	*6.21*	***6.94***

Table 4
Gap of TPhA with large instances. Time limit = 30 min.

Type/jobs	200	250	300	350	400	Avg.
P	2.73	2.40	2.08	1.96	1.84	**2.20**
S	2.49	2.21	1.79	1.54	1.40	**1.88**
P+S	2.78	2.44	2.27	2.04	1.79	**2.26**
H	3.22	2.87	2.71	2.36	2.25	**2.68**
P+H	3.76	3.30	3.08	2.73	2.62	**3.10**
S+H	12.53	11.36	11.38	11.22	11.03	**11.50**
P+S+H	12.79	11.97	12.16	11.13	11.87	**11.98**
Avg.	*5.76*	*5.22*	*5.07*	*4.71*	*4.69*	***5.09***

improves the average gap of the model (8.94). However, notice that in cases with shared resources, TPhA improves the gap results of the model and the model improves TPhA otherwise. In fact, for these cases without shared resources, when the number of jobs is equal to 50, TPhA also improves the results of the model. Moreover, as is shown in Appendix A, with other distributions, TPhA improves gaps of the model in most cases. The model obtains better percentages of optimal solutions, even in cases when it does not obtain better gaps. The reason for this is in part due to the fact that TPhA obtains its LBs only from master (phase 1) whereas the model can improve its LBs gradually. So, TPhA gives better gaps even when it finds worse LBs by improving the solution even more. Higher values of gaps are found in more difficult cases (S+H and P+S+H) and lower values can be observed when only specific resources are present (P and S cases).

For Medium instances (jobs from 60 to 140), the results for TPhA with 30 minutes as time limit are shown in Table 3.

MILP is not included since in all types for some instances (in many cases, all instances) the model does not return any feasible solutions. As the total average shows, the gap value for medium instances (6.94%) is similar to the gap value for small instances (6.94%). There is a decrease of gaps with the increase of jobs from an average of 8.28% for 60 jobs to 6.21% for 140 jobs. A reason for this comes from the fact that setup increases its amount of data exponentially with the number of jobs. For each machine a matrix of $n \times n$ values is present in setups, so for each job on each machine n possible values can be selected. When number n is large, setup selection usually is the lowest possible value among setup time data. So, it seems that this helps the solver to find good solutions. In fact, gap is increasing with number of jobs up to 50 jobs and begins to decrease from 60 jobs on. No optimal solutions are found, so all instances used up the 30 min time limit.When the average results of each type of resource are analyzed we can observe that average gap values are below 5.1% in the first five cases, and only for the last two harder cases have values below 13.4% with respect to their LBs.

Finally, for large instances from 200 to 400 jobs, Table 4 shows a similar behavior to that of medium instances: an increase of jobs gives a decrease of gap. Now the total average value of gap is 5.09% but with average values lower than 3.1% for all types of resource except the last two studied.

Summarizing, for small instances the MILP proposed can give solutions for these hard problems but with the increase of number of jobs, it begins to fail. TPhA can obtain similar values to the model for small instances and can reach solutions at least up to 400 jobs. Despite the fact that gap seems great, especially for more complex cases, an approximate solution and lower bound can be presented. Gap decreases with the increase of number of jobs, reaching less than 2.7% for 400 jobs and all types of resources except the last two harder problems when resources specific for setups and shared resources are both present.

7. Conclusion and future research

We have presented a scheduling problem with the objective of minimizing makespan. In the problem studied we have taken into account processing times, setup times and scarce resources specific for processing and/or resources specific for setups and/or resources shared for both processing and setups. This brings us closer to realistic problems such as in plastic processing plants or ceramics factories. A MILP has been developed to model all combinations of these factors. MILP is used to obtain results for small instances. The Three Phase Algorithm (TPhA) is introduced to improve the results of the model and make it possible to reach larger instances. TPhA splits the problem in three phases, assignment, sequencing and timing, with a new cut introduced to avoid local optimal. This method allows us to get results up to 400 jobs in 30 min with average gap (results vs lower bound) being around 4.7% for all possible combinations. It falls below 2% in average when we do not consider the two harder cases, when specific resources for setups and shared resources are both present. Therefore, the Three Phase Algorithm exposed seems a good first approximation for this problem.

As future research, heuristics or metaheuristics could be a way to improve these results. Also, some different objectives could be considered such as tardiness or total completion time.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Luis Fanjul-Peyro: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing - original draft, Writing - review & editing, Visualization.

Acknowledgments

The authors are partly supported by the Spanish Ministry of Science, Innovation, and Universities, under the project “OPTEP-Port Terminal Operations Optimization” (No. RTI2018-094940-B-I00) financed with Fondo Europeo de Desarrollo Regional (FEDER) funds.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.eswax.2020.100022.

Appendix A

This appendix includes:

- A new, more compact and efficient model for the UPMR problem and an example of using strip-packing methods for this problem.

- Other possible models for the UPMSR problem but obtaining worse results than the proposed model.
- Adaptation of UPMRS model to different problems
- Tests with other distributions of times for setups and processing.
- Calibrations: use of CPLEX instead of Gurobi for the several MILPs used, results with different time limits, use of CP in phase 3b (sequencing and timing together) and results using model instead of CP in phase 3a (timing with fixed assignment and sequencing).

New model for UPMR based on strip-packing

In a strip-packing 2D problem a set of rectangles must be fit in a box which has one of its dimensions fixed. The objective is to minimize the other dimension. Each rectangle has a height and a width. An approximation to the UPMR could be to understand that each job has a width, delimited by its processing time (p_{ik}), and a height, set by the number of resources needed for it to be processed (r_{ik}). In strip-packing it is necessary to know the localization of each rectangle, in this case through the coordinates of its top-right corner. In the case of UPMR, where axis x represents time and axis y the number of resources, the coordinates will be given by C_k and R_k respectively. C_k represents the time when job k is finished and R_k the point or height of number of resources. The fixed dimension is height H set by R_{max} and total width W is determined by C_{max}. In Fig. 8 two jobs (*J1, J2*) must be scheduled. *J1* is previously set and *J2* could have two (among other) different positions: later than *J1* without overlap on x (*J2a*) or higher than *J1* without overlap on y (*J2b*). Notice that, in the UPMR problem, jobs can be assigned to any machine since there are no pre-assigned jobs. So, m cases (one for each machine) should be defined for each job.

Based on the UPMR strip-packing model introduced in Fanjul-Peyro, Perea, and Ruiz (2017), a new more compact and efficient MILP is presented, where W_{jk}, H_{jk} are auxiliary binary variables.

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

$$C_{max} \geq C_k \geq \sum_{i \in M} Y_{ik} p_{ik}, \quad R_{max} \geq R_k \geq \sum_{i \in M} Y_{ik} r_{ik}, \quad \forall k \in N \tag{41}$$

$$\sum_{i \in M} Y_{ik} = 1, \quad \forall k \in N \tag{42}$$

$$C_k - C_j + A\left(1 - W_{jk}\right) \geq \sum_{i \in M} p_{ik} Y_{ik}, \quad \forall j \in N, \ k \in N, \ j \neq k \tag{43}$$

$$R_k - R_j + M\left(1 - H_{jk}\right) \geq \sum_{i \in M} r_{ik} Y_{ik}, \quad \forall j \in N, \ k \in N, \ j \neq k \tag{44}$$

$$Y_{ik} + Y_{ij} - W_{jk} - W_{kj} \leq 1, \quad \forall j \in N, \ k \in N, \ j \neq k, k > j \tag{45}$$

$$W_{jk} + W_{kj} + H_{jk} + H_{kj} \geq 1, \quad \forall j \in N, \ k \in N, \ j \neq k, k > j \tag{46}$$

$$C_k \geq 0, \ R_k \geq 0, \ H_{jk}, \ W_{jk}, \ Y_{ik} \in \{0, 1\} \tag{47}$$

In (40) the objective function minimizes makespan. The bounds limits for time and resources are set by (41). Each job can only be assigned to one machine through (42). Constrains (43) and (44) avoid overlap in time and resources axes respectively. (45) takes into account that if job j and job k are on the same machine, it is necessary and sufficient not to overlap on the time axis. In the general case, constraints (46) ensure that there is no overlap neither in the time axis nor in the resources axis. Finally, (47) defines variable types. In the UPMSR model, (45) is not necessary since (5)–(7) prevent it.

Other models for UPMSR based on time

Other MILP models can be considered. In the MILP employed in this paper, a spatial approximation is used when the strip-packing problem is adapted to be used with resources. However, a temporal approximation could be used. Based on Edis, Oguz, and Ozkarahan (2013) and as was done in paper Fanjul-Peyro, Perea, and Ruiz (2017), the temporal model used there was adapted to UPMSR but has problems giving solutions even for instances of 10 jobs (not included here). A more promising temporal approach came when the finishing times of processing and setup were considered. The new input data for UPMSR temporal MILP are:

- $E_{ikt} = 1$ if k is being processed on machine i in time t, zero otherwise.
- $F_{ijkt} = 1$ if k is in setup after job j on machine i in time t, zero otherwise.
- t_{max} is a time that is large enough

For this MILP we use Eqs. (1)–(9) and define B_k as the time when setup of job k is finished.

$$\sum_{t \leq t_{max}} E_{ikt} \geq Y_{ik} p_{ik}, \quad \forall i, k \in N \tag{48}$$

$$C_k \geq \sum_{i \in M} t E_{ikt}, \quad \forall k \in N, \ t \leq t_{max} \tag{49}$$

$$C_k + 1 - \sum_{i \in M} Y_{ik} p_{ik} \leq \sum_{i \in M} t E_{ikt} + A\left(1 - \sum_{i \in M} E_{ikt}\right), \quad \forall k \in N, \ t \leq t_{max} \tag{50}$$

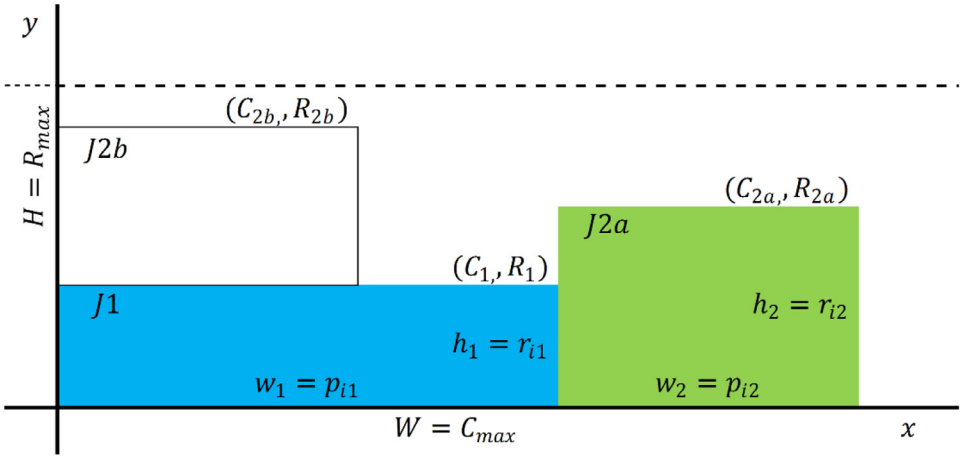

Fig. 8. UPMR from strip-packing

$$\sum_{t \le t_{max}} F_{ijkt} \ge X_{ijk} s_{ijk}, \ \forall i, k \in N, \ j \in N_0 \tag{51}$$

$$B_k \ge \sum_{i \in M, j \in N_0} t F_{ijkt}, \ \forall k \in N, \ t \le t_{max}, \tag{52}$$

$$B_k + 1 - \sum_{i \in M, j \in N_0} X_{ijk} s_{ijk} \le \sum_{i \in M, j \in N_0} t F_{ijkt} + A\left(1 - \sum_{i \in M, j \in N_0} F_{ijkt}\right), \quad \forall k \in N, \ t \le t_{max} \tag{53}$$

Constraints (48) and (49) are the limits of E_{ikt}. Eq. (50) forces when E_{ikt} has value 1 at time t, this time t is greater than or equal to the difference between the completion time of job k and its processing time plus 1. Constraints (51)–(53) represent the same but for setups, where the time to be considered is s_{ijk}.

For resources, this model uses simpler constraints than the MILP model presented in this paper.

- Resources in processing

$$\sum_{i \in M, k \in N} r^{\mathcal{P}}_{ik} E_{ijt} \le R^{\mathcal{P}}_{max}, \ \forall t \le t_{max} \tag{54}$$

Constraints (54) force that, at each time t, the sum of all resources used in processing should be lower than or equal to the maximum resources to process.

- Resources in setups

$$\sum_{i \in M, j \in N, k \in N, j \ne k} r^{\mathcal{S}}_{ijk} F_{ijkt} \le R^{\mathcal{S}}_{max}, \ \forall t \le t_{max} \tag{55}$$

Constraints (55) force that, at each time t, the sum of all resources used in setups should be lower than or equal to the maximum resources for setups.

- Shared resources

$$\sum_{i \in M, k \in N} r^{\mathcal{H}p}_{ik} E_{ijt} + \sum_{i \in M, j \in N, k \in N, j \ne k} r^{\mathcal{H}s}_{ijk} F_{ijkt} \le R^{\mathcal{H}}_{max}, \forall t \le t_{max} \tag{56}$$

Constraints (56) force that, at each time t, the sum of all shared resources used in processing and all shared resources used in setups should be lower than or equal to the maximum shared resources.

This model only allows us to reach results up to 30 jobs in the easier cases (resources only in processing or in setups) but not in all cases. So, that put it quite far from the results of MILP used in this paper which has results up to 50 jobs.

Reduction of UPMSR model to different problems

Different types of problems can be studied with the model proposed in Eqs. (1)–(26). In the next subsection we formalize these different problems.

UPM: Unrelated Parallel Machine

In the case of UPM no resources are present, therefore constraints (13)–(26) are not useful. Since there are not any setups in UPM, all s_{ijk} are zero and all X_{ijk} could be set to zero. In this case, only Eqs. (1), (2), and (4) are taken into account. This reduced model is the classical model used for UPM problems.

UPMS: UPM with Setups

In the case of UPMS no resources are present, therefore constraints (13)–(26) are not useful. Since no idle times are present in UPMS problem, constraints (10)–(12) are not necessary. So only Eqs. (1)–(9) need to be taken into account. This reduced model is like the Avalos-Rosales, Angel-Bello, and Alvarez (2015) model for UPMS problem.

UPMR: UPM with Resources

In the case of UPMR the resources present are only specific for processing (since there are no setups). Therefore constraints (14), (15) and (19)–(26) are not useful. Since there are not any setups in UPMR, all s_{ijk} are zero but if we do not set all X_{ijk} to zero, constraints help us to avoid overlap in the time axis. So only Eqs. (1)–(13) and (16)–(18) are taken into account. This reduced model improves the results of the best model published of UPMR and also improves the results of the new model specific for UPMR, introduced in Appendix A, as we show later in computational results.

UPMSR-P: UPM with Setup and Resources (UPMSR) with only specific processing resources

In cases where there are always setups and any type of resource is present, equations from (1) to (12) are always present. In the case with only specific processing resources (P-resources) we should add constraints relative to these resources which are (13) for limits, and (16)–(18) to avoid their overlaps.

UPMSR-S: UPMSR with only specific setup resources

In problems with setups and resources Eqs. (1)–(12) are present. In the case with only specific setup resources (S-resources), the limit for these resources are set by (14). In order to avoid overlaps when S-resources are taken into account, we should add constraints (19)–(21).

UPMSR-H: UPMSR with only unspecific shared resources

In the case of UPMSR with unspecific resources shared by both processing and setups, we need Eqs. (1)–(12) and constraints for the limit of shared resources, i.e. (15). In this case, when the resources could be used in processing or in setups indistinctly, we should take into account the contribution to the total use of this type of resource that processing and setups require. Consequently, we should consider the relationship of processing/processing, setup/setup and processing/setup. So besides constraints (22)–(26) to avoid processing/setup overlaps, we need to use constraints (16)–(18) changing superscript P for superscript H to avoid processing/processing overlap in the use of shared resources. Also we need to use constraints (19)–(21) in order to prevent overlaps in the setup/setup case of use of shared resources, changing superscript S for H. Summarizing, we should use Eqs. (1)–(12) and (15)–(26) changing superscripts P and S for H.

UPMSR-P+S: UPMSR with specific processing resources and specific setup resources

This case is a combination of cases UPMSR-P and UPMSR-S. So we need constraints (1)–(12) as general constraints and (13) and (14) as limit constraints. Finally (16)–(18) are for specific processing resources and (19)–(21) for specific setup resources constraints.

UPMSR-P+H: UPMSR with specific processing resources and unspecific shared resources

This case is a combination of cases UPMSR-P and UPMSR-H. So we need constraints (1)–(12) as general constraints (13)–(15) as limit constraints. Constraints (16)–(18) are for specific processing resources (P-resources). For shared resources we need a new set of constraints (16)–(18) changing superscripts P for H in order to avoid overlap in the case of processing/processing shared resources (H-resources). For shared resources we also need to avoid overlap in the case of setup/setup with constraints (19)–(21) changing superscripts S for H and finally constraints (22)–(26) to avoid processing/setup overlap.

Table 5
Model and TPhA with different distributions.

	Model		TPhA	
	Gap	Opt%	Gap	Opt%
U(1,100)	7.12	60.48	3.78	50.79
U(10, 100)	8.04	52.54	4.81	41.75
U(100,200)	8.94	37.94	7.69	27.62
JobCorre	12.27	32.54	9.29	27.62
MaqCorre*	9.36	48.89	7.02	32.54
Avg.	9.15	46.48	6.52	36.06

Table 6
TPhA with different dominion.

	p=U(100,200) s=U(100,200)	p=U(100,200) s=U(50,100)	p=U(50,100) s=U(100,200)	p=U(50,100) s=U(50,100)
P	4.32	5.59	3.46	4.39
S	3.81	2.19	5.49	3.86
P+S	4.43	5.34	5.62	4.47
H	4.57	5.89	3.79	4.55
P+S+H	13.87	14.46	14.58	13.39
Avg.	*6.20*	*6.69*	*6.59*	*6.13*

UPMSR-S+H: UPMSR with specific setup resources and unspecific shared resources

This case is a combination of cases UPMSR-S and UPMSR-H. So we need constraints (1)–(12) as general constraints and (14) and (15) as limit constraints. Constraints (19)–(21) are for specific setup resources (S-resources). For shared resources we need a new set of constraints (19)–(21) changing superscripts S for H in order to avoid overlap in the case of setup/setup shared resources (H-resources). For shared resources we also need to avoid the overlap processing/processing case with constraints (16)–(18) changing superscripts P for H and finally constraints (22)–(26) to avoid processing/setup overlap.

UPMSR-P+S+H: UPMSR with specific processing resources, specific setup resources and unspecific shared resources

This case is a combination of cases UPMSR-P, UPMSR-S and UPMSR-H. So we need constraints (1)–(12) as general constraints, (13)–(15) as limit constraints. Constraints (16)–(18) are for specific processing resources (P-resources). Constraints (19)–(21) are for specific setup resources (S-resources). For shared resources we need a new set of constraints (16)–(18) changing superscripts P for H in order to avoid overlap in the case of processing/processing shared resources. We also need a new set of constraints (19)–(21) changing superscripts S for H in order to avoid overlap in the case of setup/setup shared resources. Finally, constraints (22)–(26) are to avoid processing/setup overlap.

Other distributions

In order to make this paper more clear, only balanced instances used in papers such as Arnaout, Rabadi, and Musa (2010), Tran, Araujo, and Beck (2016) or Fanjul-Peyro, Ruiz, and Perea (2019) are exposed. However, other distributions used in papers such as Fanjul-Peyro, Perea, and Ruiz (2017) or Fleszar and Hindi (2018) will be studied now.

These distributions include five different types of processing times, three uniform random distributions and two more distributions with high dependence of jobs and machines, respectively. The five distributions are:

- $U(1, 100), U(10, 100), U(100, 200)$ which generate random numbers between the lower and higher limits (both included) for each p_{ik}.
- Correlated jobs where each job k has a fixed time with independence of machine according to $p_k = U(1, 100)$ to which is added an additional time $p'_{ik} = U(1, 20)$. Therefore $p_{ik} = p_k + p'_{ik}$.
- Correlated machines where each machine i has a fixed time to process a job with independence of which job it is, according to $p_i = U(1, 100)$ to which is added an additional time $p'_{ik} = U(1, 20)$. Therefore $p_{ik} = p_i + p'_{ik}$.

In Table 5 we studied for small instances the five processing distribution times proposed, with setup distribution time of U(1,49) and U(1,99) and five resource types i.e. specifically for processing (P), specifically for setups (S), shared resources (H), when types P and S are present (P+S), and when all types are present (P+S+H). Notice that the number of instances rises to 3,750. Since the model returns no feasible solutions up to 30 jobs when shared resources are present (cases H and P+S+H), only jobs from 10 to 30 are considered in these cases and jobs from 10 to 50 for other cases (P, S, P+S) as in the main part of the paper.

As Table 5 shows, in all cases the Three Phase Algorithm improves gap results of the model and the model can demonstrate more solutions as optimal. Higher values of gap are shown in the Job Correlated case, and U(1,100) distribution gives lower values. In the Machine Correlated case, the model gives a poor value for an instance of 50 jobs which is discarded since Table 2 is used to compare the model with TPhA.

For the five types of resources considered here, the average TPhA (6.52%) obtains a similar value for gap to the average of these five types of instances used in the main part of the paper (6.16%). But the model, with a 9.15% of average gap, worsens the instances results of the paper which are on average 6.21% for these five types. However, with respect to the percentage of optimal solutions, these distributions obtain better results than the ones in the paper. Here the model finds 46.48% of optimal solutions versus 32.06% in the paper, and TPhA here finds 36.06% versus 22.22% in the paper.

In Arnaout, Rabadi, and Musa (2010) a balanced distribution of setup and processing times are used. However, a processing dominion or setup dominion can be also used. In order to maintain the same dispersion of 100 units as most commonly used in the literature and to fix similar values for maximum and minimum values we use the following distributions:

- Balanced: p=U(100,200); s=U(100,200).
- Processing dominion: p=U(100,200); s=U(50,100).
- Setup dominion: p=U(50,100); s=U(100,200).

In Table 6 we also include the results obtained in the paper for balanced distribution of p=U(50,100) and s=U(50,100) in order to compare it.

On average, the results are similar for all these distributions. For each distribution it seems that some types of resources make the problem easier or harder, which depends on dominion. The balanced distributions have intermediate values. No appreciable differences can be found in the use of the two different balanced distributions studied.

Calibrations

Some aspects should be clarified as to why in the paper we use different components. One of these is the use of Gurobi or Cplex solvers. In the case of the model, Cplex begins to fail at 40 jobs for easier types of resources whereas Gurobi has no problem up to 50 jobs. In the case of TPhA, Table 7 shows that results improve using Gurobi in a set of 150 medium instances of calibration. Another question is why to use CP for phase 3a (timing with assignment and sequencing variables fixed). The reason is because using MILP

Table 7
Variants of TPhA.

	Gurobi	Cplex	3bCP
P	4.51	5.58	6.39
S	0.99	1.68	4.46
P+S	4.14	5.41	6.31
H	5.98	6.29	6.35
P+S+H	13.47	14.41	12.78
Avg.	5.82	6.67	7.26

Table 8
Time study of TPhA.

						% decrease	
	5′	15′	30′	1h	3h	5′ vs 3h	30′ vs 3h
P	5.81	5.40	4.51	4.10	3.95	32.02	12.33
S	1.95	1.22	0.99	0.84	0.67	65.55	32.32
P+S	4.96	4.46	4.14	4.02	3.45	30.47	16.67
H	6.80	6.14	5.98	5.83	5.76	15.40	3.67
P+S+H	13.84	13.78	13.47	13.47	13.24	4.31	1.73
Avg.	*6.67*	*6.20*	*5.82*	*5.65*	*5.41*	*29.55*	*13.34*

in these cases returns no feasible solutions in jobs higher than 100 jobs. Remember that the phase 3a is done with each master solution found and it is necessary to find a feasible solution. The phase 3b is done only with the optimal solutions of master. In these cases we have previous feasible solutions coming from not optimal master solutions which have been made feasible. If TPhA finds difficulties in exiting phase 3b in some instances, we have a feasible previous solution. But if TPhA fails in phase 3a perhaps no high quality solution could be found. That is the reason why in phase 3a it is better to use CP but in phase 3b MILP should be used, i.e. If MILP finds it hard to solve a particular instance in phase 3b, we have a previous solution, but if not, MILP gives solutions that are a bit better than CP in phase 3b as we will see below. Finally, the use of CP in phase 3b (sequencing and timing together) versus the use of MILP shows a difference in favor of MILP as Table 7 shows. With these values we make the decisions of which ones should be selected to be used in the paper. However, the differences in Table 7 are small. In the ANOVA test, P-value of 0.51 shows that no significant differences in means are present. In a brief study of larger instances, it shows minor differences between average values observed in Table 7. Moreover, for large instances it is recommendable to use CP in phase 3b since MILP can produce problems of out of memory.

In order to delimit the time to be used in TPhA, we test a set of 150 calibration instances in a time test shown in Table 8. As can be seen in Table 8 each increase of time improves the results a bit (between 3% and 7%), however the difference between 5 min and 180 min improves up to 65% in one case (S) with an average of 29%. The difference is reduced to 13% on average with a maximum of 32% (S) when we consider 30 min vs 180 min. Moreover, 30 min is the minimum time to obtain feasible solutions for the largest instances of 400 jobs and 8 machines. However, ANOVA with *P*-value of 0.53 shows that no significant differences between means are present.

References

Akyol Özer, E., & Saraç, T. (2019). MIP models and a matheuristic algorithm for an identical parallel machine scheduling problem under multiple copies of shared resources constraints. *An Official Journal of the Spanish Society of Statistics and Operations Research, 27*, 94–124. doi:10.1007/s11750-018-00494-x.

Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research, 246*(2), 345–378. doi:10.1016/j.ejor.2015.04.004.

Arnaout, J., Rabadi, G., & Musa, R. (2010). A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. *Journal of Intelligent Manufacturing, 21*(6), 693–701. doi:10.1007/s10845-009-0246-1.

Avalos-Rosales, O., Angel-Bello, F., & Alvarez, A. (2015). Efficient metaheuristic algorithm and re-formulations for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times. *The International Journal of Advanced Manufacturing Technology, 76*(9-12), 1705–1718. doi:10.1007/s00170-014-6390-6.

Bektur, G., & Saraç, T. (2019). A mathematical model and heuristic algorithms for an unrelated parallel machine scheduling problem with sequence-dependent setup times, machine eligibility restrictions and a common server. *Computers & Operations Research, 103*, 46–63. doi:10.1016/j.cor.2018.10.010.

Błazewicz, J., Kubiak, W., Röck, H., & Szwarcfiter, J. (1987). Minimizing mean flowtime with parallel processors and resource constraints. *Acta Informatica, 24*(5), 513–524. doi:10.1007/BF00263292.

Błazewicz, J., Lenstra, J. K., & Rinnooy Kan, A. H. (1983). Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics, 5*(1), 11–24. doi:10.1016/0166-218X(83)90012-4.

Daniels, R. L., Hua, S. Y., & Webster, S. (1999). Heuristics for parallel-machine flexible-resource scheduling problems with unspecified job assignment. *Computers & Operations Research, 26*(2), 143–155. doi:10.1016/S0305-0548(98)00054-9.

Edis, E. B., & Oguz, C. (2011). In *Parallel machine scheduling with additional resources: A Lagrangian-based constraint programming approach: 6697* (pp. 92–98). Lecture Notes in Computer Science. doi:10.1007/978-3-642-21311-3_10.

Edis, E. B., & Ozkarahan, I. (2012). Solution approaches for a real-life resource-constrained parallel machine scheduling problem. *The International Journal of Advanced Manufacturing Technology, 58*, 1141–1153. doi:10.1007/s00170-011-3454-8.

Edis, E. B., Oguz, C., & Ozkarahan, I. (2013). Parallel machine scheduling with additional resources: Notation, classification, models and solution methods. *European Journal of Operational Research, 230*(3), 449–463. doi:10.1016/j.ejor.2013.02.042.

Fanjul-Peyro, L., Perea, F., & Ruiz, R. (2017). Models and matheuristics for the unrelated parallel machine scheduling problem with additional resources. *European Journal of Operational Research, 260*(2), 482–493. doi:10.1016/j.ejor.2017.01.002.

Fanjul-Peyro, L., Ruiz, R., & Perea, F. (2019). Reformulations and an exact algorithm for unrelated parallel machine scheduling problems with setup times. *Computers & Operations Research, 101*, 173–182. doi:10.1016/j.cor.2018.07.007.

Fleszar, K., & Hindi, K. S. (2018). Algorithms for the unrelated parallel machine scheduling problem with a resource constraint. *European Journal of Operational Research, 271*(3), 839–848. doi:10.1016/j.ejor.2018.05.056.

Graham, R., Lawler, E., Lenstra, J., & Rinnooy Kan, A. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics, 5*, 287–326. doi:10.1016/S0167-5060(08)70356-X.

Grigoriev, A., Sviridenko, M., & Uetz, M. (2007). Machine scheduling with resource dependent processing times. *Mathematical Programming, 110*(1), 209–228. doi:10.1007/s10107-006-0059-3.

Guinet, A. (1991). Textile production systems: A succession of non-identical parallel processor shops. *Journal of the Operational Research Society, 42*(8), 655–671. doi:10.1057/jors.1991.132.

Lenstra, J., Rinnooy Kan, A., & Brucker, P. (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics, 1*, 343–362. doi:10.1016/S0167-5060(08)70743-X.

Lodi, A., Martello, S., & Monaci, M. (2002). Two-dimensional packing problems: A survey. *European Journal of Operational Research, 141*(2), 241–252. doi:10.1016/S0377-2217(02)00123-6.

Pinedo, M. L. (2016). *Scheduling: Theory, algorithms and systems* (5th ed.). New York, USA: Springer. doi:10.1007/978-3-319-26580-3.

Słowinski, R. (1980). Two approaches to problems of resource allocation among project activities: A comparative study. *Journal of the Operational Research Society, 31*(8), 711–723. doi:10.1057/jors.1980.134.

Tran, T., Araujo, A., & Beck, J. (2016). Decomposition methods for the parallel machine scheduling problem with setups. *Informs Journal on Computing, 28*(1), 1–195. doi:10.1287/ijoc.2015.0666.

Vallada, E., & Ruiz, R. (2011). A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research, 211*(3), 612–622. doi:10.1016/j.ejor.2011.01.011.

Villa, F., Vallada, E., & Fanjul-Peyro, L. (2018). Heuristic algorithms for the unrelated parallel machine scheduling problem with one scarce additional resource. *Expert Systems with Applications, 93*, 28–38. doi:10.1016/j.eswa.2017.09.054.

Zheng, X., & Wang, L. (2016). A two-stage adaptive fruit fly optimization algorithm for unrelated parallel machine scheduling problem with additional resource constraints. *Expert Systems with Applications, 65*, 28–39. doi:10.1016/j.eswa.2016.08.039.