Scheduling unrelated parallel machines with resource-assignable sequence dependent setup times

Rubén Ruiz[1]*, Carlos Andrés[2]

[1] Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática,

Universidad Politécnica de Valencia, Edificio 7A. Camino de Vera S/N, 46021, Valencia, Spain.

[2]Departamento de Organización de Empresas,

Universidad Politécnica de Valencia, Edificio 7D. Camino de Vera S/N, 46021, Valencia, Spain.

email: rruiz@eio.upv.es, candres@omp.upv.es.

Abstract

A novel scheduling problem that results from the addition of resource-assignable setups is presented in this paper. We consider an unrelated parallel machine problem with machine and job sequence dependent setup times. The new characteristic is that the amount of setup time do not only depend on the machine and job sequence, but also on a amount of resources assigned, which can vary between a minimum and a maximum. The aim is to give solution to real problems arising in several industries where frequent setup operations in production lines have to be carried out. These operations are indeed setups whose length can be reduced or extended according to the amount of resources assigned to them. The objective function considered is a linear combination of total completion time and the total amount of resources assigned. We present a MIP model and some fast dispatching heuristics. We carry out careful and comprehensive statistical analyses to study what characteristics of the problem affect the MIP model performance. We also study the effectiveness of the different heuristics proposed.

Keywords: scheduling, unrelated parallel machines, sequence dependent setups, total completion time, resources

*Corresponding author. Tel: +34 96 387 70 07, ext: 74946. Fax: +34 96 387 74 99

1 Introduction

Parallel machine scheduling problems contain an interesting number of characteristics that set them aside in the scheduling field. In these problems, there is a set $N = 1, 2, \dots, n$ of jobs where each one has to be processed exactly on one machine from the set $M = 1, 2, \dots, m$. Firstly, jobs have to be assigned to machines and secondly, among the jobs assigned to each machine, a processing sequence must be derived. Among parallel machine scheduling problems, the most general case is when the parallel machines are said to be unrelated. In this scenario, the processing time of each job depends on the machine to which it is assigned. This processing time is deterministic and known in advance and is denoted by p_{ij}, $i \in M$, $j \in N$.

In this paper we are interested in a complex and realistic variant of the unrelated parallel machine scheduling problem. We consider machine and sequence dependent setup times which can be separated from processing times. A setup is a non-productive period of time which usually modelizes operations to be carried out on machines after processing a job to leave them ready for processing the next job in the sequence. Therefore, S_{ijk} is the setup time to be carried out on machine i after having processed job j and before processing job k, $i \in M$, $j, k \in N, j \neq k$. Setup times frequently represent cleaning production lines and/or adding/removing elements from a line when changing from one type of job to another. A clear example stems from the ceramic tile manufacturing sector. At the glazing stage there are several unrelated parallel lines (machines) capable of glazing the different types of ceramic tile lots (jobs). After glazing a lot, the glazing line needs to be cleaned and prepared for the glazing of the next lot or job. This is the setup time which clearly depends on the line type and on the processing sequence. Modern glazing lines are easier to clean whereas old and end-of-life lines need more careful cleaning. Additionally, if the last tile lot glazed had a strong color, a much more time consuming cleaning must be carried out, specially if the next tile lot has a clear color. Hence, setup times are machine and sequence dependent.

S_{ijk} is difficult to set in practice. One of the reasons is that if more resources are devoted to a given setup operation, the setup time decreases. In the previous ceramic tile example, resources are usually cleaning machinery and/or plant personnel. The cleaning of the glazing line might take several hours if a single person is carrying out the cleaning. However, if several workers clean the line simultaneously, the setup time decreases. In this paper, apart from the machine and sequence dependent setup times, we also consider the resources assigned to each setup, which affect the time the setup needs. We will denote by R_{ijk} the amount of resources devoted to carrying out setup S_{ijk}. Usually, this amount will vary inside a predefined interval. We refer to this as resource-assignable sequence dependent setup times or RASDST in short. Obviously, assigning more resources would certainly reduce the total setup time. However, such resources are usually costly and a trade-off situation arises. Consequently, we are interested in a realistic optimization criterion which minimizes the total production time or flowtime. If we denote by C_j the completion time of job j at the shop, the total completion time or flowtime is given by

$\sum_{j=1}^{n} C_j$. Is is well known that minimizing flowtime minimizes as well cycle time and maximizes throughput (Pinedo, 2002). All these measures are very important in practice. Of course, the total resource assignment must be considered. We denote by T_{Res} the total amount of resources assigned to setups (i.e., $T_{Res} = \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1,k \neq j}^{n} R_{ijk}$). Additionally, given α and β that modelize the importance or cost of each unit of resource and flowtime, respectively, the objective function considered in this paper is $\alpha T_{Res} + \beta \sum_{j=1}^{n} C_j$, where $\alpha, \beta \geq 0$. It has to be noted that T_{Res} and total flowtime objectives are measured in very different units and total flowtime is expected to be orders of magnitude larger than the total amount of resources assigned to setups. In such scenario, the weights α and β are useful to compensate for this potential problem. Such a problem, that can be denoted as $R/S_{ijk}, R_{ijk}/\alpha T_{Res} + \beta \sum_{j=1}^{n} C_j$ by extending the notation given in Graham et al. (1979) is obviously $\mathcal{NP}$-Hard. The unrelated parallel machine problem without setups and total completion time criterion ($R//\sum_{j=1}^{n} C_j$) is in $\mathcal{P}$ according to Horn (1973) where the optimum solution can be simply derived by a reduction to an assignment problem. However, the case where the parallel machines are identical and there are only sequence independent family setup times, i.e., the problem $P/S_j/\sum_{j=1}^{n} C_j$ is already $\mathcal{NP}$-Hard as explained in Webster (1997). Therefore, by simple reduction to this problem, the unrelated parallel machines problem with resource-assignable sequence dependent setup times tackled in this paper is also $\mathcal{NP}$-Hard. As a result, heuristics are the only alternative for realistically-sized problems.
In Section 2 we review the literature of related problems, since, as we will show and to the best of our knowledge, the problem tackled in this paper has not been addressed in the literature before. Due to its novelty, a MIP model for obtaining the exact solution is presented in Section 3, along with a more detailed notation of the problem. Section 4 presents some static and dynamic dispatching heuristics for the problem, along with an exact rule for assigning setup resources, once the job assignment to machines and processing sequence is given. A complete testing and evaluation of both the MIP model as well as the heuristics is given in Section 5. Comprehensive statistical analyses help supporting the results. Lastly, some conclusions and further research topics are presented in Section 6.

2 Literature review

In the literature there are several reviews and surveys about scheduling with setup times like the ones of Yang and Liao (1999), Allahverdi et al. (1999) and more recently, Allahverdi et al. (2008). There are also specific surveys dealing with parallel machine settings like those of Cheng and Sin (1990), Lam and Xing (1997) and Mokotoff (2001). A careful examination of these reviews results in no paper dealing with the problem considered in this work.
From our perspective, unrelated parallel machine problems with sequence dependent setup times contain the principal characteristics of the problem studied here and therefore we focus the review mainly on them. Some of the earlier studies come from Marsh and Montgomery (1973)

where problems with identical as well as unrelated parallel machines and sequence dependent setup times are considered. The objective function is the minimization of total setups. Guinet (1991) considered the unrelated machines case also with sequence and machine dependent setups. Heuristics and mathematical modeling were employed to solve the mean tardiness and mean completion time criteria. In a later paper, Guinet and Dussauchoy (1993) study the same problem but in this case with identical parallel machines and makespan as well as mean completion time criteria. The authors develop some routing-based heuristics as solution methods. Lee and Pinedo (1997) propose a three-phase heuristic also for the identical machine problem but in this case for a total weighted tardiness criterion. Most literature deals with the identical parallel machine problem and finding results for the unrelated machine case with sequence dependent setup times is difficult. For example, the papers of Balakrishnan et al. (1999), Sivrikaya-Serifoglu and Ulusoy (1999), or Radhakrishnan and Ventura (2000) propose different techniques for the identical or uniform parallel machine problems with sequence dependent setup times and using different earliness and tardiness criteria. There are, however, some exceptions. In Zhu and Heady (2000), a mixed integer programming model is proposed for the unrelated machines, sequence dependent setup problem with earliness/tardiness objective. Weng et al. (2001) study the same problem but with two differences: the criterion studied is the mean completion time and the setup times depend only on the job sequence and not on the machine. The authors propose seven dispatching heuristics for the problem and compare the results in comprehensive experiments. A similar problem is dealt with in Kim et al. (2002) but with total tardiness criterion and with the additional characteristic that jobs are grouped into families and the setups are family-based. More recently, Rabadi et al. (2006) have proposed several GRASP-like heuristics for an unrelated machines case with machine and job sequence dependent setup times with makespan criterion.
As we can see, to the best of our knowledge, no scheduling problem has been studied in the literature where the setup times can be resource-assignable. However, there is a large body of literature where the processing times can be controlled. A good review of these types of problems is given by Nowicki and Zdrzalka (1990). In these cases, processing time can be controlled or "compressed" at a given cost. The closest reference we find is due to Zhang et al. (2001) where an unrelated parallel machine problem with controllable processing times is examined. In this case, the objective considered in a combination of total weighted completion time and total processing time compression costs. However, no setup times are considered. Ng et al. (2005) study a single machine group scheduling problem where jobs belong to different groups and a sequence independent setup time is incurred when changing from one group to another. In this work, both processing times as well as setups are controllable trough the assignment of divisible as well as discrete resources. However, once assigned, all setups use the same units of resource. A recent work is that of Grigoriev et al. (2007) where approximation algorithms are proposed for an unrelated parallel machine problem where there are renewable resources that can be assigned to compress processing times. The optimization criterion considered is makespan minimization.

Lastly, we find an interesting reference due to Chen (2006), where a complex problem from the semiconductor industry is approached. The problem resembles an unrelated parallel machine setting with tooling mounting and un-mounting at the machines. This mounting operations can be seen as sequence-independent setups. The interesting fact is that although the setup times are fixed, a given resource is needed for each setup.
The conclusion from this review is that the complex unrelated parallel machine with resource-assignable machine and job sequence dependent setup times problem considered in this paper is novel. To the best of our knowledge, this problem has not been considered before in the literature.

3 Problem and MIP model formulation

Now we give further notation and definitions for the unrelated parallel machine RASDST problem. For each possible setup time S_{ijk} we have a set of four values. The first two specify the minimum and maximum quantities of resources that can be assigned to each setup. Therefore, $R^-_{ijk}(R^+_{ijk})$ denote the minimum (maximum) resources assignable to setup S_{ijk}. Let us consider again the ceramic tile glazing lines example of Section 1. Some setups might require removing certain heavy tooling from the glazing lines and therefore a minimum amount of workers (resources) are needed. Similarly, at a given moment it might not be possible to assign more resources due to physical or operational constraints. The relation between the actual setup time and the amount of resources assigned is assumed in this work to be linear. Therefore, if the minimum resources R^-_{ijk} are assigned, the resulting setup time will be the largest possible, denoted as S^+_{ijk}. Conversely, if the maximum resources are assigned, R^+_{ijk}, the setup will be minimum (S^-_{ijk}). This relation is shown in Figure 1.

`[Insert Figure 1 about here]`

Therefore, from the linear relation, the expression that returns the amount of setup time S_{ijk} as a function of the assigned resources R_{ijk} is the following:

$$S_{ijk} = S^+_{ijk} - \frac{S^+_{ijk} - S^-_{ijk}}{R^+_{ijk} - R^-_{ijk}}(R_{ijk} - R^-_{ijk}) = S^+_{ijk} + \frac{S^+_{ijk} - S^-_{ijk}}{R^+_{ijk} - R^-_{ijk}}R^-_{ijk} - \frac{S^+_{ijk} - S^-_{ijk}}{R^+_{ijk} - R^-_{ijk}}R_{ijk} \qquad (1)$$

Using K_1 and K_2 for the first terms and for the R_{ijk} multiples, respectively we have as a result the slope-intercept form of the line relating S_{ijk} and R_{ijk}:

$$S_{ijk} = K_1 - K_2 R_{ijk} \qquad (2)$$

Therefore, K_2 is the slope of the line that relates R_{ijk} with S_{ijk} as shown in Figure 1. In other words, K_2 indicates how much the setup time S_{ijk} is reduced by each additional resource. For example, given $S^+_{ijk} = 20$, $S^-_{ijk} = 4$, $R^+_{ijk} = 10$ and $R^-_{ijk} = 2$ we would have $S_{ijk} = 20 - \frac{20-4}{10-2}(R_{ijk} - 2) = 24 - 2R_{ijk}$. As a result, each additional resource would decrease the setup time in two units.

With the definition of the setup times we can now derive a mathematical mixed integer program (MIP) to obtain the optimum solution of the unrelated parallel machine RASDST problem. The model uses the following decision variables:

$$\begin{aligned} X_{ijk} &= \begin{cases} 1, \text{ if job } j \text{ immediately precedes job } k \text{ on machine } i \\ 0, \text{ otherwise} \end{cases} \\ C_{ij} &= \text{Completion time of job } j \text{ at machine } i \\ R_{ijk} &= \text{Resources assigned to the setup between job } j \text{ and job } k \text{ on machine } i \end{aligned}$$

The objective function is:

$$\min \sum_{i\in M} \sum_{j\in N} \sum_{\substack{k\in N\\ k\neq j}} \alpha R_{ijk} + \sum_{i\in M} \sum_{j\in N} \beta C_{ij} \tag{3}$$

And the constraints are:

$$\sum_{i\in M} \sum_{\substack{j\in\{0\}\cup\{N\}\\ j\neq k}} X_{ijk} = 1, \qquad k \in N \tag{4}$$

$$\sum_{i\in M} \sum_{\substack{k\in N\\ j\neq k}} X_{ijk} \leq 1, \qquad j \in N \tag{5}$$

$$\sum_{k\in N} X_{i0k} \leq 1, \qquad i \in M \tag{6}$$

$$\sum_{\substack{h\in\{0\}\cup\{N\}\\ h\neq k, h\neq j}} X_{ihj} \geq X_{ijk}, \qquad j,k \in N, j \neq k, i \in M \tag{7}$$

$$C_{ik} + V(1 - X_{ijk}) \geq C_{ij} + K_1 - K_2 R_{ijk} + p_{ik}, \qquad j \in \{0\} \cup \{N\}, k \in N, j \neq k, i \in M \tag{8}$$

$$R_{ijk} \geq R^-_{ijk}, \qquad j,k \in N, j \neq k, i \in M \tag{9}$$

$$R_{ijk} \leq R^+_{ijk}, \qquad j,k \in N, j \neq k, i \in M \tag{10}$$

$$C_{i0} = 0, \qquad i \in M \tag{11}$$

$$C_{ij} \geq 0, \qquad j \in N,\ i \in M \tag{12}$$

$$X_{ijk} \in \{0,1\}, \qquad j \in \{0\} \cup \{N\}, k \in N, j \neq k, i \in M \tag{13}$$

The objective minimizes a linear combination of the total resources assigned (T_{Res}) and the total completion time. Constraint set (4) ensures that every job is assigned to exactly one machine and also that has exactly one predecessor. Notice the usage of dummy jobs 0 as X_{i0k}, $i \in M, k \in N$. Conversely, constraint set (5) limits the number of successors of every job to one. Set (6) limits

the number of successors of the dummy jobs to a maximum of one on each machine, i.e., dummy job 0 can have at most one successor on each machine. With set (7) we ensure that jobs are properly linked in machines since if a given job j is processed o a given machine i, it must have a predecessor h on the same machine. Constraint set (8) is central for controlling the completion times of the jobs at the machines. Basically, if a job k is assigned to machine i after job j (i.e., $X_{ijk} = 1$), its completion time C_{ik} must be greater than the completion time of j, C_{ij} plus the setup time between j and k and the processing time of k. Notice how the setup time is expressed here according to equation (2). If $X_{ijk} = 0$, then the big constant V renders the constraint redundant. An important issue arises if $R^-_{ijk} = R^+_{ijk}$. In this situation, $S^-_{ijk} = S^+_{ijk} = S_{ijk}$ and therefore, the term $K_1 - K_2 R_{ijk}$ in constraint set (8) is simply substituted by the constant S_{ijk}. Constraint sets (9) and (10) bound the possible assignable resources. Sets (11) and (12) define completion times as 0 for dummy jobs and non-negative for regular jobs, respectively. Finally, set (13) defines the binary variables.
Overall, the model contains n^2m binary variables, $n(n-1)m + nm + m$ continuous variables and a total of $n^2m + 3n(n-1)m + nm + 2n + 2m$ constraints.

4 Heuristic methods

Dispatching rules are frequently used in practice due to their intrinsic simplicity and ease of use. While outperformed by modern metaheuristics, their small footprint and sheer speed makes them preferable in highly dynamic environments like those depicted in previous sections and found in ceramic tile manufacturing. As a result, we propose several dispatching rules that complement the MIP model as regards simplicity and speed.
Shortest Processing Time (SPT) dispatching rules are known to give optimum solutions for simplistic scheduling problems with total completion time ($\sum_{j=1}^{n} C_j$) criterion (see Pinedo, 2002). Additionally, all seven heuristics proposed in Weng et al. (2001) for the $R/S_{jk}/\frac{1}{n}\sum_{j=1}^{n} w_j C_j$ problem are based on this dispatching rule, or more precisely on the Weighted Shortest Processing Time (WSPT). The main rationale behind the SPT rule for total completion time criterion is that the completion time of the first job assigned to a given machine contributes to the completion times of all other jobs assigned to the same machine. For example, picture three jobs to be assigned to a given machine with processing times $p_1 = 1$, $p_2 = 9$ and $p_3 = 10$. Discarding setups and resources, a sequence $\{1, 2, 3\}$ gives completion times of 1, 10 and 20 for each job, respectively, with a total completion time of 31. However, a sequence of $\{3, 2, 1\}$ results in completion times of 10, 19 and 20, respectively and a total completion time of 49. It is clear that processing first the shortest jobs results in a lower overall total completion time. Therefore, we also base our proposed heuristics on the SPT rule.

4.1 Shortest Processing Time with Setups resource Assignment (SPTSA)

This procedure returns, for every machine, the jobs assigned to it along with their sequence and the resources assigned to each possible setup.

1. For every job j calculate the machine with the minimum processing time, i.e., $l = \arg\min_{i \in M} p_{ij}$. Store job j, machine l and the minimum processing time p_{lj} in a vector V of size n
2. Sort vector V in increasing order of the minimum processing time of each job on all machines (p_{il})
3. For $j = 1$ to n do
   (a) Take from $V[j]$ the job k and the machine l to which assign the job k
   (b) Assign resources to the setup between the last job assigned to machine l and job k
   (c) Assign job k to machine l

After the job assignment, A final issue remains about the quantity of resources to assign to each setup (step 3b). We test three different approaches where we assign minimum, maximum, and average resources. We refer to this approaches as SPTSA$_-$, SPTSA$_+$ and SPTSA$_{av}$, respectively. The average resources are just calculated with the expression $\frac{R^+_{ijk}+R^-_{ijk}}{2}$. Notice that when the first job is assigned to a given machine, resources are assigned to the initial setup. However, without loss of generality, we consider this initial setup (and the corresponding resources) to be zero.

4.2 Shortest Processing and Setup Time with Setups resource Assignment (SPSTSA)

SPSTSA is very similar to SPTSA, the only difference is that average setups are also considered along with the minimum processing times for deriving the dispatching order of jobs. Therefore, the first two steps of the SPTSA are modified as follows:

1. For every job j calculate the minimum processing time and average setup time for all machines, i.e., calculate an index I as follows:

$$I_j = \min_{i \in M} \left(p_{ij} + \frac{1}{n-1} \sum_{k=1, k \neq j}^{n} \frac{S^+_{ijk} + S^-_{ijk}}{2} \right) \tag{14}$$

   Let l be the machine where the minimum I_j has been obtained. Store job j, machine l and I_j in a vector V of size n

2. Sort vector V in increasing order of I_j

The other steps are equal to SPTSA. Contrary to SPTSA, SPSTSA also considers the expected setup time between a job assigned to a given machine and all the possible successors and thus giving an estimation of the processing time plus setup time which in turn will increase the completion time of succeeding jobs.

4.3 Dynamic Job Assignment with Setups resource Assignment (DJASA)

Both previous dispatching rules have a main drawback. In an unrelated parallel machine problem, the minimum processing time, or minimum index I for two jobs can be on the same machine l. Therefore, assigning both jobs to l might not be the best solution since the first job might be assigned to l and the second job to another machine where the processing time or index I might be slightly larger. In this other solution, the total completion time is likely to be lower. DJASA solves this problem by dynamically considering all pending jobs and all machines at each iteration. The procedure is the following:

1. Add all jobs to a list of unscheduled jobs ρ
2. While $\rho \neq \varnothing$ do
    (a) For every pending job in ρ ($\rho_{(j)}$) and for every machine $i \in M$ do
        i. Temporarily assign resources to the setup between job $\rho_{(j)}$ and the previous job assigned to machine i
        ii. Assign $\rho_{(j)}$ to machine i
        iii. Calculate the objective $\alpha T_{Res} + \beta \sum_{j=1}^{n} C_j$ after the resource and job assignment
    (b) Let j and l be the job and machine that has resulted in the minimum overall objective increase, respectively
    (c) Assign resources to the setup between job j and the previous job assigned to machine l
    (d) Assign job j to l
    (e) Remove job j from ρ

As with SPTSA and SPSTSA, a decision has to be made on the quantity of resources to assign. Therefore, three methods are derived depending on wether minimum, maximum or average resources are assigned at steps (2(a)i) and (2c). We will refer to the three resulting DJASA methods as DJASA$_-$, DJASA$_+$ and DJASA$_{av}$, respectively.
For SPTSA and SPSTSA, the overall complexity is $\mathcal{O}(n \log n)$ since the most expensive operation is the sorting of the processing times and I index, respectively. For DJASA, a total of $\frac{n(n+1)}{2}m$ job insertions are carried out, which results in a $\mathcal{O}(n^2 m)$ complexity.

4.4 Optimal resource assignment

Once the assignment of the different jobs to the unrelated parallel machines and the sequence among them is known, several observations can be derived from the resources assigned to the setups:

- Shortening setups early in the machine' sequences is likely to have a large effect on total completion times
- Saving resources (expanding setups) late in the sequence has a lesser effect on total completion times
- The most interesting setups are those with a high slope K_2 according to expression (2)

The last observation is particularly interesting, a high slope K_2 means that large reductions in setup time are obtained by assigning additional resources. Given the linearity of the relation between S_{ijk} and R_{ijk}, the same reductions are obtained for each additional resource in the range $[R^-_{ijk}; R^+_{ijk}]$. As a result, the following assignment procedure can be applied:

1. For each machine i and for each job k assigned to i except the last do
   (a) Let j be the predecessor of k at i or 0 if job k is the first job assigned to i
   (b) Let h be the number of successors of job k assigned to machine i
   (c) The resources R_{ijk} assigned to the setup between jobs j and k at machine i are re-assigned according to the following expression:

$$R_{ijk} = \begin{cases} R^+_{ijk}, & \text{if } K_2 \cdot h \cdot \beta > \alpha \\ R^-_{ijk}, & \text{otherwise} \end{cases} \tag{15}$$

It is straightforward to see that the previous re-assignment of setup resources is optimal. Reducing the setup time one unit will reduce the completion times of all jobs processed after the setup in one unit as well. In a simple case where $K_2 = \alpha = \beta = 1$, if two jobs are assigned to a given machine after a setup, increasing the resources of this setup in one unit will improve the objective. In this case, and giving the linearity of K_2, the maximum resources (minimum setup) should be assigned to obtain the maximum savings.

This optimal resource assignment procedure is very simple and, given a schedule with all the job assignments to machines along with their sequences, it takes only $\mathcal{O}(n)$ steps to optimally re-assign resources. This procedure can be applied to the resulting solution of SPTSA, SPSTSA and DJASA. In the case of SPTSA and SPSTSA, the original resource assignment will be changed and therefore we refer to these methods with the optimal resource assignment as SPTSA* and SPSTSA*, respectively. Notice that for DJASA, since jobs are assigned dynamically, the way resources are assigned during the algorithm might have an effect on the final job sequences, independently of the final re-assignment. Therefore, apart from the three DJASA procedures (DJASA$_+$, DJASA$_-$ and DJASA$_{av}$), we have three more that result from the application of the re-assignment procedure: DJASA$^*_+$, DJASA$^*_-$ and DJASA$^*_{av}$

4.5 Application example

We present an example application of the DJASA$_{av}$ and DJASA$^*_{av}$ heuristics to a small four job, 2 machine problem. The processing times are given in Table 1 while the minimum and maximum resources and setups are given in Table 2.

`[Insert Tables 1 and 2 about here]`

Now we apply the dynamic dispatching rule DJASA$_{av}$. Let us work with $\alpha = 30$ and $\beta = 1$, i.e., each unit of resource is 30 times more expensive or important than each flowtime unit. Additionally, we are going to assume that only an integer quantity of resources can be assigned. Initially, all four jobs are unscheduled so $\rho = 1, 2, 3, 4$. Since we assume than the initial setups are zero, the job with the shortest processing time on all machines is assigned. From Table 1 we see that the lowest processing time corresponds to job 3 on machine 2 ($p_{23} = 27$). Therefore job 3 is scheduled first on machine 2. We remove job 3 from ρ.
Now we consider jobs all pending jobs $\rho = 1, 2, 4$ at machines 1 and 2:

1. Job 1, machine 1: $C_{11} = C_{10} + S_{101} + p_{11} = 0 + 0 + 79 = 79$. 79 units increase in objective.
2. Job 1, machine 2: Resources assigned: $\frac{R^+_{231}+R^-_{231}}{2} = 3$ which correspond to $S_{231} = 63.5$. $C_{21} = C_{23} + S_{231} + p_{21} = 27 + 63.5 + 45 = 135.5$. $30 \cdot 3 + 135.5 = 225.5$ units increase in objective.
3. Job 2, machine 1: $C_{12} = C_{10} + S_{102} + p_{12} = 0 + 0 + 51 = 51$. 51 units increase in objective.
4. Job 2, machine 2: Resources assigned: $\frac{R^+_{232}+R^-_{232}}{2} = 3$ which correspond to $S_{232} = 59.5$. $C_{22} = C_{23} + S_{232} + p_{22} = 27 + 59.5 + 78 = 164.5$. $30 \cdot 3 + 164.5 = 254.5$ units increase in objective.
5. Job 4, machine 1: $C_{14} = C_{10} + S_{104} + p_{14} = 0 + 0 + 43 = 43$. **43** units increase in objective.
6. Job 4, machine 2: Resources assigned: $\frac{R^+_{234}+R^-_{234}}{2} = 3.5 \rightarrow 4$ which correspond to $S_{234} = 36.3$. $C_{24} = C_{23} + S_{234} + p_{24} = 27 + 36.3 + 90 = 153.3$. $30 \cdot 4 + 153.3 = 273.3$. 273.3 units increase in objective.

The lowest increase in the objective comes after assigning job 4 to machine 2. Therefore, job 4 is assigned first on machine 1. We remove job 4 from ρ.
Now we consider jobs all pending jobs $\rho = 1, 2$ at machines 1 and 2:

1. Job 1, machine 1: Resources assigned: $\frac{R^+_{141}+R^-_{141}}{2} = 2$ which correspond to $S_{141} = 54.5$. $C_{11} = C_{14}+S_{141}+p_{11} = 43+54.5+79 = 176.5$. $30\cdot 2+176.5 = 236.5$. 236.5 units increase in objective.

2. Job 1, machine 2: Resources assigned: $\frac{R^+_{231}+R^-_{231}}{2} = 3$ which correspond to $S_{231} = 63.5$. $C_{21} = C_{23} + S_{231} + p_{21} = 27 + 63.5 + 45 = 135.5$. $30 \cdot 3 + 135.5 = 225.5$. **225.5** units increase in objective.

3. Job 2, machine 1: Resources assigned: $\frac{R^+_{142}+R^-_{142}}{2} = 3.5 \to 4$ which correspond to $S_{142} = 36$. $C_{12} = C_{14} + S_{142} + p_{12} = 43 + 36 + 51 = 130$. $30 \cdot 4 + 130 = 250$. 250 units increase in objective.

4. Job 2, machine 2: Resources assigned: $\frac{R^+_{232}+R^-_{232}}{2} = 3$ which correspond to $S_{232} = 59.5$. $C_{22} = C_{23}+S_{232}+p_{22} = 27+59.5+78 = 164.5$. $30\cdot 3+164.5 = 254.5$. 254.5 units increase in objective.

The lowest increase in the objective comes after assigning job 1 to machine 2. Job 1 is assigned second on machine 2 and 3 units of resource are assigned to setup S_{231}. We remove job 1 from ρ. Now we consider the last pending job in $\rho = 2$ at machines 1 and 2:

1. Job 2, machine 1: Resources assigned: $\frac{R^+_{142}+R^-_{142}}{2} = 3.5 \to 4$ which correspond to $S_{142} = 36$. $C_{12} = C_{14} + S_{142} + p_{12} = 43 + 36 + 51 = 130$. $30 \cdot 4 + 130 = 250$. **250** units increase in objective.

2. Job 2, machine 2: Resources assigned: $\frac{R^+_{212}+R^-_{212}}{2} = 3.5 \to 4$ which correspond to $S_{212} = 34$. $C_{22} = C_{21} + S_{212} + p_{22} = 135.5 + 34 + 78 = 247.5$. $30 \cdot 4 + 247.5 = 367.5$. 367.5 units increase in objective.

In this last step, job 2 is assigned second on machine 2 and 4 units of resource are assigned to setup S_{142}. After removing job 2 from ρ we have that $\rho = \varnothing$ so the procedure is finished. In total there are $4+3 = 7$ resources assigned. The total completion time is $43+130+27+135.5 = 335.5$ so the objective value is $30 \cdot 7 + 335.5 = 545.5$. A Gantt chart with this solution along with all details is shown in Figure 2.

`[Insert Figure 2 about here]`

At this point, we apply the optimal resource assignment. We only have two setups in the example, $S_{142} = 36$ with $R_{142} = 4$ and $S_{231} = 63.5$ with $R_{231} = 3$. There is one job after each setup on both cases. Therefore:

1. Setup at machine 1 between jobs 4 and 2 ($K_2 = 25$): $K_2 \cdot 1 \cdot 1 \not> 30$. Therefore R_{142} is re-assigned to $R^-_{142} = 3$. The objective value decreases by 5 units ($\alpha - K_2$).

2. Setup at machine 2 between jobs 3 and 1 ($K_2 = 35.5$): $K_2 \cdot 1 \cdot 1 > 30$. Therefore R_{231} is re-assigned to $R^+_{231} = 4$. The objective value decreases by 5.5 units ($K_2 - \alpha$).

As a result of the optimal resource assignment we have a new objective value of $545.5 - 5 - 5.5 = 535$. The resulting Gantt chart is given in Figure 3.

```
[Insert Figure 3 about here]
```

5 Computational Evaluation

In this section our aim is to test the MIP model and the proposed heuristics for this novel unrelated parallel machine RASDST problem. In order to do so, a comprehensive benchmark has been defined. As has been shown in the previous example problem, apart from the number of jobs n and the number of machines m, several sets of data have to be generated. These are the processing times p_{ij} and the minimum and maximum resources and setups R^-_{ijk}, R^+_{ijk}, S^-_{ijk} and S^+_{ijk}, respectively. The MIP model is not expected to be usable for large instances, given the number of variables and constraints needed. Therefore, two sets of instances are defined. For the small instances, all the following combinations of n and m are tested: $n = \{6, 8, 10\}$ and $m = \{3, 4, 5\}$. For the large instances the combinations are $n = \{50, 75, 100\}$ and $m = \{10, 15, 20\}$. In all cases, the processing times are generated according to a uniform distribution in the range $[1, 99]$ as it is usual in the scheduling literature. For the minimum and maximum resources we consider two combinations. In the first one the minimum and maximum resources are uniformly distributed in the ranges $[1, 3; 3, 5]$. For the second case the uniform distributions are $[1, 5; 5, 10]$. This means that R^-_{ijk} is distributed as $U[1, 3]$ and $U[1, 5]$ and R^+_{ijk} as $U[3, 5]$ and $U[5, 10]$. However, only two combinations are considered. For the setup times we also work with two combinations $U[1, 50; 50, 100]$ and $U[50, 100; 100, 150]$. These two cases modelize medium and large setup times, respectively. Notice that there is a possibility of R^-_{ijk} being equal to R^+_{ijk}. We think this captures the reality in which some setups might not be resource-assignable. In such cases the value of S^-_{ijk} is equal to S^+_{ijk}.

As a conclusion, for the small instances we have three values for n, three for m, two combinations of resources and two of setups. 10 instances are generated for each case so there are 360 small instances. The same applies for the large instances so the total number of instances considered is 720. In order to study both the MIP model and the heuristics, we set the weights for the objective function as $\alpha = 50$ and $\beta = 1$. The rationale behind this decision is that resources are usually scarce and more expensive than each unit of flowtime. Also, flowtime values are much higher that total resources. Furthermore, this cost structure resembles common practices in the ceramic tile industries.

5.1 MIP model performance study

We construct a model in LP format for each instance in the small set. The resulting 360 models are solved with CPLEX 9.1 on a Pentium IV 3.2 GHz computer with 1 GByte of RAM memory. We have refrained from developing an ad-hoc branch and bound method mainly due to the fact that obtaining good lower bounds for the RASDST problem is probably a unfruitful venue of research. The reason is that lower bounds are extremely weak in the presence of sequence dependent setup times. In simpler scheduling environments no good lower bounds exist in the presence of sequence dependent setup times since the amount of setups depends on the sequence. Furthermore, commercial solvers for parallel machine problems with setups have also been used by Balakrishnan et al. (1999) and Zhu and Heady (2000).
We solve all instances with two different time limits, 5 and 60 minutes. We think that allowing more time is not realistic since scheduling problems need to be solved in a matter of minutes in a production floor. As we will see, not all instances are solved within these time limits. Consequently, for each run we record a categorical variable which we refer to as "type of solution" with two possible values 0 and 1. Type 0 means that an optimum solution was found, in which the objective value and the time needed for obtaining it are recorded. Type 1 means that the time limit was reached and at the end a feasible integer solution was found. The gap is recorded in such a case. The averaged results for all levels of n, m, R_{ijk} and S_{ijk} are given in Table 3. Each cell gives the average of the 10 replicated instances. The percentage of instances for which a type 0 solution was found (%Opt) and the average time needed for this optimum solution (Avg. Time) are displayed. The percentage of instances with a type 1 solution (%Limit) is also shown in the table along with the observed gap (GAP%). Finally, the average number of variables and constraints is also given.

`[Insert Table 3 about here]`

Some direct conclusions can be drawn from Table 3. For example, all instances with $n = 6$ are solved to optimality under both stopping time criteria. However, for $n = 8$, only a few optimum solutions are obtained within 5 minutes CPU time. For 60 minutes CPU times, all $n = 8$ instances are solved. For $n = 10$ no optimum solution could be found for any instance even with 60 minutes CPU time. The effect of other factors is also easy to see. Increasing the number of machines m results in instances that are easier to solve. At first, this observation might seem counterintuitive. However, with fewer machines, more jobs are assigned to each machine and the importance of the job sequence and sequence dependent setup times, as well as assignable resources, becomes more visible. The largest CPU times among the solved instances are observed for $n = 8$ and $m = 3$. The same applies to the largest gaps in unsolved instances ($n = 10$ and $m = 3$). The effect of the resources is also interesting. From the Table it seems that having more resources (i.e., $U[1, 5; 5, 10]$) results in instances that are easier to solve. On the other hand, larger setups ($U[50, 100; 100, 150]$) result in harder instances. All these observations are important since they allow us to characterize which elements from the novel unrelated parallel machines RASDST problem affect performance. Lastly, we can see that the number of variables and constraints (as reported by CPLEX after model reduction techniques) are largely affected by n and m, as the other factors do not seem to contribute significantly.

Overall, the performance of the MIP model is good, specially when compared to results from the literature. Balakrishnan et al. (1999) solved problems of up to $n = 10$ and $m = 4$ but on a simpler setting with uniform parallel machines and no resources. Their MIP model contains significantly less variables due to the triangular law of inequality assumption in the setup times, i.e., Balakrishnan et al. (1999) assume that $S_{ijk}+S_{ikl} \geq S_{ijl}$. Obviously, in our RASDST problem this assumption is unacceptable since the length of the setups might change greatly depending on the resources assigned. Zhu and Heady (2000) work with an unrelated parallel machines case and use a similar variable definition to the one employed in this paper. In this case, the largest size of problem solved is $n = 9$ and $m = 3$ needing almost 5,400 seconds of CPU time. Considering that in our model, apart from the job assignment and the job sequencing we have also the assignable-resources, we conclude that the MIP model performance is good.

5.2 MIP model statistical analysis

All previous discussion is solely based on observed average values. It is important to carry out statistical testing in order to have a sound basis for the conclusions reached. We use a not so common advanced statistical technique called Automatic Interaction Detection (AID). AID recursively bisects experimental data according to one factor into mutually exclusive and exhaustive sets that explain the variability in a given response variable in the best statistically significant way. AID was originally proposed by Morgan and Sonquist (1963). Kass (1980) improved the initial AID by including statistical significance testing in the partition process and by allowing multi-way splits of the data. The improved version is called Chi-squared Automatic Interaction Detection (CHAID). Biggs et al. (1991) developed an improved version known as Exhaustive CHAID. AID techniques are a common tool in the fields of Education, Population Studies, Market Research, Psychology, etc. CHAID was recently used by Ruiz et al. (2008) also for the analysis of a MIP model.
We adopt the Exhaustive CHAID method for the analysis of the MIP model results. The procedure starts with all data contained in a root node. From this point, all controlled factors are studied and the best multi-way split of the root node according to the levels of each factor is calculated. A statistical significance test is carried out to rank the factors on the splitting capability. The procedure is applied recursively to all nodes until no more significant partitions can be found. A decision tree is constructed from the splitting procedure. This tree allows a visual and comprehensive study of the effect of the different factors on the studied response variable. We use SPSS DecisionTree 3.0 software. The factors n, m, Resources and Setups are controlled.

We introduce all data of both stopping CPU time criteria, so the factor Time is also controlled. The response variable is the type of solution with two possible values (0 and 1). We set a high confidence level for splitting of 99.9% and a Bonferroni adjustment for multi-way splits that compensates the statistical bias in multi-way paired tests. The full decision tree is shown in Figure 4.

`[Insert Figure 4 about here]`

As with Table 3, the most important factor on determining the type of solution is n. As we can see, the statistical test with which the root node has been split has a very large Chi-Square value and an Adjusted P-Value of essentially 0. This means that the split is statistically significant at a very high confidence level. This first level split statistically confirms that n is the most influential factor on the type of solution. For instances of 8 jobs the situation is very interesting. First of all, with 60 minutes CPU time, all instances are solved (second level split). For 5 minutes CPU time, the tree has many other levels. The third split corresponds to the number of machines m. According to this, instances with 4 or less machines are more difficult than instances with 5 machines. Lastly, for $n = 8$ Time=5 and $m = 5$, large setups result in harder instances.
It is also interesting to repeat the analysis but changing the response variable to the observed gap percentage. In this case, the response variable is no longer categorical but rather continuous. The resulting tree is very large and is not shown here for reasons of space. However, the following observations can be made from the results:

- For $n = 8$ and Time=5 there is a large tree with one first split on m. In this case, the gap percentage increases with decreasing values of m
- For $n = 8$, Time=5 and the different splits on m there are further splits. The overall observation is that the gap percentage is larger for Resources $= U[1, 5; 5, 10]$ and Setups $= U[50, 100; 100, 150]$. This is consistent for all the multi-way splits according to the values of m.
- For $n = 10$ and Time=5 the next split is based on Setups, not in m as in the previous case. Again, larger setups result in larger gap percentages
- For $n = 10$, Time=5 and for the two values of setups, the trend for the resources is reversed, i.e., more resources result in slighter lower gap percentages
- For $n = 10$ and Time=60 the next split is based in m, with larger gap percentages as m decreases
- For $n = 10$, Time=60 and all the values for m, the following splits are either based on Setups or on Resources. Again, larger setups result larger gap percentages.
- For $n = 10$, Time=60, values of m and all values of Setups, higher amount of resources results in lower gap percentages.

- The largest gap percentage under 60 minutes CPU time is observed for the following combination of values: $n = 10$, $m = 3$, Setups=$U[50, 100; 100, 150]$ and Resources=$U[1, 3; 3, 5]$. Here the gap percentage is almost 60%.

After this comprehensive analysis we know that for the MIP model the number of jobs and machines clearly affect the difficulty. To be precise, a high n/m ratio results in the most difficult problems. Large setups and low quantity of assignable resources also contribute to the difficulty.

5.3 Heuristic evaluation

We proceed now to test the performance of the proposed heuristics. Recall that there are three main methods each one with three possible resource assignment variations: SPTSA$_-$, SPTSA$_+$, SPTSA$_{av}$, SPSTSA$_-$, SPSTSA$_+$, SPSTSA$_{av}$, DJASA$_-$, DJASA$_+$ and DJASA$_{av}$. Finally, we can apply the optimal resource assignment procedure to SPTSA, SPSTSA and to the three DJASA variants, this gives us five more methods: SPTSA*, SPSTSA*, DJASA$^*_-$, DJASA$^*_+$ and DJASA$^*_{av}$. As a result, there are 14 different methods. All heuristics have been coded in Delphi 2006 and have been tested on the same computer used for testing the MIP model.
We test each method on the small instances, where we measure the average percentage increase ($AVRPD$) over the optimum solution (or best solution found) given by the MIP model. For the large instances we test the average percentage deviation over the best solution found by all heuristics. The $AVRPD$ is measured as follows:

$$AVRPD = \frac{1}{R}\sum_{r=1}^{R}\frac{Heu_{sol_r} - Opt_{sol_r}}{Opt_{sol_r}}100 \qquad (16)$$

where Heu_{sol_r} is the heuristic solution obtained by any of the 14 methods on instance replicate r. Opt_{sol_r} is the MIP optimum or best solution known for that instance replicate and for the small instances or the best solution known across all results for that instance replicate for the large instances. All instances, as well as the best known solutions are available from `http://www.upv.es/gio/rruiz`. Table 4 gives the $AVRPD$ values for the small instances. The results are shown for every possible combination of n, m, Setups and Resources. Recall that there are 10 instances per each combination and therefore each cell contains the $AVRPD$ across 10 values.

`[Insert Table 4 about here]`

As expected, DJASA is the best method over SPSTSA and this later one is in turn better than SPTSA. The $AVRPD$ values drop strongly for DJASA in all situations. Among the three resource assignment methods, ($_-$, $_+$ and $_{av}$) the best solutions are obtained by assigning the maximum resources and the worst solutions by assigning the lowest resources. This is consistent for the three heuristics. Obviously, increasing the value of α would favor assigning less resources. As expected, the optimal resource assignment procedure improves the results always since the

procedure cannot deteriorate the solutions. What is more interesting is to actually quantify the improvements. For example, comparing the best SPTSA method, SPTSA$_+$ with SPTSA$^*_+$ we see a difference of 5.2%. This difference is 4.6% for SPSTSA and a mere 0.4% for DJASA. However, the differences are much larger if we compare other initial resource assignments. The trend for n and m is not clear. It seems that larger values of m result in higher $AVRPD$ values. What seems clear is the effect of the setups and resources. Higher number of resources and lower duration of setups results in higher deviations.
Overall, the best result is given by algorithm DJASA$^*_+$ with an average deviation from optimum solutions of 11.1%. In reality, deviation from optimum solution can only be given for instances with $n \leq 8$. In this case, the $AVRPD$ is 12.41%. While being relatively large, it is fairly small considering that DJASA is a simple dispatching rule.

As with the case of the MIP model, the previous table contains observed averages alone. We carry out a full experimental analysis using the ANOVA technique. In the experiment we control n, m, R, S and Algorithm as factors. Algorithm is a factor with 14 levels where 1=SPTSA$_-$, 2=SPSTSA$_-$,..., 14=DJASA$^*_{av}$, i.e., we follow the same ordering as in Table 4. We consider all 10 instance replicates so there is a total of $3 \cdot 3 \cdot 2 \cdot 2 \cdot 14 = 504$ treatments. Since we have 10 different instances at each group, the number of experimental units is $504 \cdot 10 = 5,040$. All methods are deterministic, so no replicates are calculated. The parametric ANOVA assumptions are accepted after a careful study of the experiment's residuals.
From the results of the experiments, all controlled factors as well as many interactions of order two are significant. We find that most of the observed differences in the overall performance of the different algorithms are statistically significant. A straightforward way of analyzing these differences is via a means plot with multiple comparison confidence intervals. Figure 5 shows a means plot for the different algorithms with Honest Significant Difference (HSD) Tukey intervals at 99% confidence. These intervals counteract the bias in multiple pairwise comparisons. Overlapping intervals indicate that no statistically significant differences are observed between the overlapped levels of the studied factor.

`[Insert Figure 5 about here]`

According to the plot, there are no statistically significant differences among the following algorithms (1,2), (7,8), (4,5,10,11), (3), (9,12) and finally (6,13,14). Of interest is this last group, formed by DJASA$_+$, DJASA$^*_+$ and DJASA$^*_{av}$. However, this is an overall plot, i.e., across all instances. A more careful examination (not shown here) indicates that for many instances DJASA$^*_+$ gives statistically better solutions than DJASA$^*_{av}$. As a matter of fact, among the 360 small instances, DJASA$^*_+$ gives better solution than DJASA$^*_{av}$ in 215. Both heuristics give the same solution in 96 cases and only on 49 cases DJASA$^*_{av}$ gives better solution.
All other observations made from the results are statistically confirmed. Increasing n results in statistically better solutions for all algorithms. The same applies to lower values of m, low

amount of resources and large setups. Interestingly, these two last factors showed a reverse effect on the MIP model.

Now we test the results of the heuristics for the large instances. We cannot longer compare the solutions obtained against the optimum ones since these are not known. Furthermore, we cannot assume that the observed average deviations of the heuristics from optimum solutions in the small instances will hold for large instances. However, it is still interesting to observe how the different methods perform. Table 5 gives the $AVRPD$ values for the large instances.

`[Insert Table 5 about here]`

It can be observed than DJASA produces much better results than all the other methods. Also, the optimal resource assignment method improves the results significantly. Much of the discussion from the small instances' results applies to these large instances as well. A second experimental design results in the means plot for the algorithm factor of Figure 6.

`[Insert Figure  6  about here]`

More or less the same statistical differences from the small instances are observed in the large set. There are however some differences. The observed clusters are (1,2), (7,8), (4,5), (10,11), (9), (3), (6,12) and (13,14). There are more clusters as the intervals are narrower due to the less observed variance. As with the small instances, a large number of jobs and small number of machines produces the highest deviations. The trend observed for the resources and setups is maintained.

A much desirable characteristic of the proposed heuristics is their speed. Among the 5,040 results gathered with the 14 algorithms in the large instances, the largest observed elapsed CPU time (not wall time) has been 15.6 milliseconds. For all instances in the small set, the CPU time needed is below of what can be reliably measured. For the large instances, the slowest method is DJASA$^*_{av}$ but as said, the CPU times are in the low milliseconds range. Recall that the proposed heuristics have a very low complexity. These heuristics can therefore be used in real-time scheduling environments.

6 Conclusions and future research

In this paper, a novel complex scheduling problem is considered. The setting consists in a number of unrelated parallel machines where machine and sequence dependent separable setup times are present. The novelty resides in that the duration of the setups depend on assignable resources. This characteristic has been named as Resource Assignable Sequence Dependent Setup Times or RASDST in short. A combination of total assigned resources and total completion time is used as a criterion. A MIP model and some fast heuristics have been proposed. The model has been

carefully and comprehensively evaluated by means of advanced statistical tools. The results of the study allow us to identify which characteristics of the problem have an effect on difficulty. Overall, the performance of the MIP model is deemed as good considering the large number of constraints and variables needed.
Three dispatching heuristics and an optimal resource assignment rule have been also proposed. The heuristics, along with many options and variants, have been tested on small as well as on large problems. The results, supported also by statistical experimentation, indicate that the dynamic job dispatching rule with the optimal resource assignment procedure, a method referred to as DJASA$^*_+$, is the best heuristic among those proposed. Furthermore, the CPU times of the proposed heuristics are negligible and are therefore useful in real-time scheduling scenarios.
We consider this is a good starting work on a practical new type of problems where setup times are resource-assignable. These types of setups are very common in production lines where the setups represent cleaning and or adjustments or machines and where these operations require plant personnel. Hence, the more personnel, the faster the setup.
There are a number of interesting research lines. First and foremost, local search heuristics and metaheuristic techniques will surely improve the results of the different proposed dispatching heuristics. Another important research direction is to limit the number of available resources, both as a maximum quantity over all the programming period or as a maximum availability at every possible moment (i.e., resource calendars). This would allow to modelize situations in which a limited number of employees have to carry out the different setup operations across different working shifts.

Acknowledgments

The authors are indebted to the referees and editor for a close examination of the paper, which has increased its quality and presentation. This work is partially funded by the Spanish Ministry of Science and Innovation, under the project "SMPA - Advanced Parallel Multiobjective Sequencing: Practical and Theoretical Advances" with reference DPI2008-03511/DPI. The authors should also thank the IMPIVA - Institute for the Small and Medium Valencian Enterprise, for the project OSC with references IMIDIC/2008/137, IMIDIC/2009/198 and IMIDIC/2010/175.

References

Allahverdi, A., Gupta, J. N. D., and Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *OMEGA, The international Journal of Management Science*, 27(2):219–239.

Allahverdi, A., Ng, C. T., Cheng, T. C. E., and Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3):985–1032.

Balakrishnan, N., Kanet, J. J., and Sridharan, S. V. (1999). Early/tardy scheduling with sequence dependent setups on uniform parallel machines. *Computers & Operations Research*, 26(2):127–141.

Biggs, D., De Ville, B., and Suen, E. (1991). A method of choosing multiway partitions for classification and decision trees. *Journal of Applied Statistics*, 18(1):49–62.

Chen, J.-F. (2006). Unrelated parallel machine scheduling with secondary resource constraints. *International Journal of Advanced Manufacturing Technology*, 26(3):285–292.

Cheng, T. C. E. and Sin, C. C. S. (1990). A state-of-the-art review of parallel machine scheduling research. *European Journal of Operational Research*, 47(3):271–292.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5:287–326.

Grigoriev, E., Sviridenko, M., and Uetz, M. (2007). Unrelated parallel machine scheduling with resource dependent processing times. *Mathematical Programming: Series A and B*, 110(1):209–228.

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

Guinet, A. and Dussauchoy, A. (1993). Scheduling sequence dependent jobs on identical parallel machines to minimize completion time criteria. *International Journal of Production Research*, 31(7):1579–1594.

Horn, W. A. (1973). Minimizing average flow time with parallel machines. *Operations Research*, 21(3):846–847.

Kass, G. V. (1980). An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, 29(2):119–127.

Kim, D. W., Kim, K. H., Jang, W., and Chen, F. F. (2002). Unrelated parallel machine scheduling with setup times using simulated annealing. *Robotics & Computer Integrated Manufacturing*, 18(3-4):223–231.

Lam, K. and Xing, W. (1997). New trends in parallel machine scheduling. *International Journal of Operations & Production Management,*, 17(3):326–338.

Lee, Y. H. and Pinedo, M. (1997). Scheduling jobs on parallel machines with sequence dependent setup times. *European Journal of Operational Research*, 100(3):464–474.

Marsh, J. D. and Montgomery, D. C. (1973). Optimal procedures for scheduling jobs with sequence-dependent changeover times on parallel processors. *AIIE Technical Papers*, pages 279–286.

Mokotoff, E. (2001). Parallel machine scheduling problems: a survey. *Asia-Pacific Journal of Operational Research*, 18(2):193–242.

Morgan, J. A. and Sonquist, J. N. (1963). Problems in the analysis of survey data and a proposal. *Journal of the American Statistical Association*, 58:415–434.

Ng, C. T., Edwin Cheng, T. C., Janiak, A., and Kovalyov, M. Y. (2005). Group scheduling with controllable setup and processing times: Minimizing total weighted completion time. *Annals of Operations Research*, 133:163–174.

Nowicki, E. and Zdrzalka, S. (1990). A survey of results for sequencing problems with controllable processing times. *Discrete Applied Mathematics*, 26(2-3):271–287.

Pinedo, M. (2002). *Scheduling: Theory, Algorithms, and Systems.* Prentice Hall, Upper Saddle, N.J, second edition.

Rabadi, G., Moraga, R. J., and Al-Salem, A. (2006). Heuristics for the unrelated parallel machine scheduling problem with setup times. *Journal of Intelligent Manufacturing*, 17(1):85–97.

Radhakrishnan, S. and Ventura, J. A. (2000). Simulated annealing for parallel machine scheduling with earliness-tardiness penalties and sequence-dependent set-up times. *International Journal of Production Research*, 38(10):2233–2252.

Ruiz, R., Sivrikaya Şerifoğlu, F., and Urlings, T. (2008). Modeling realistic hybrid flexible flowshop scheduling problems. *Computers & Operations Research*, 35(4):1151–1175.

Sivrikaya-Serifoglu, F. and Ulusoy, G. (1999). Parallel machine scheduling with earliness and tardiness penalties. *Computers & Operations Research*, 26(8):773–787.

Webster, S. T. (1997). The complexity of scheduling job families about a common due date. *Operations Research Letters*, 20(2):65–74.

Weng, M. X., Lu, J., and Ren, H. (2001). Unrelated parallel machines scheduling with setup consideration and a total weighted completion time objective. *International Journal of Production Economics*, 70(3):215–226.

Yang, W.-H. and Liao, C.-J. (1999). Survey of scheduling research involving setup times. *International Journal of Systems Science*, 30(2):143–155.

Zhang, F., Tang, G. C., and Chen, Z. L. (2001). A 3/2-approximation algorithm for parallel machine scheduling with controllable processing times. *Operations Research Letters*, 29(1):41–47.

Zhu, Z. and Heady, R. (2000). Minimizing the sum of earliness/tardiness in multi-machine scheduling: a mixed integer programming approach. *Computers & Industrial Engineering*, 38(2):297–305.

	i	1	2
j	1	79	45
	2	51	78
	3	32	27
	4	43	90

Table 1: Processing times (p_{ij}) for the example problem.

Resources, (R^-_{ijk}; R^+_{ijk})

	$i=1$				$i=2$			
	k				k			
j	1	2	3	4	1	2	3	4
1	-	3;4	1;4	3;5	-	3;4	3;4	2;4
2	1;5	-	1;5	1;3	2;2	-	3;4	2;4
3	3;3	1;3	-	2;5	2;4	1;5	-	2;5
4	1;3	3;4	3;5	-	2;4	2;5	2;3	-

Setups, (S^-_{ijk}; S^+_{ijk})

	$i=1$				$i=2$			
	k				k			
j	1	2	3	4	1	2	3	4
1	-	47;95	28;51	17;63	-	34;58	41;52	29;53
2	10;51	-	35;58	50;57	30;30	-	44;55	31;60
3	28;28	5;57	-	18;50	28;99	44;75	-	19;71
4	20;89	36;61	22;93	-	40;97	23;76	21;83	-

Table 2: Minimum and maximum resources and setup times (R^-_{ijk}, R^+_{ijk}, S^-_{ijk} and S^+_{ijk}) for the example problem.

		Time	5				60			
		Resources	1		2		1		2	
n	m	Setups	1	2	1	2	1	2	1	2
6	3	%Opt	100	100	100	100	100	100	100	100
		Avg. Time	2.93	4.22	2.3	2.35	2.93	4.22	2.28	2.37
		%Limit	0	0	0	0	0	0	0	0
		GAP%	-	-	-	-	-	-	-	-
		Vars.	205	205	212.7	213.6	205	205	212.7	213.6
		Constr.	371	371	386.4	388.2	371	371	386.4	388.2
	4	%Opt	100	100	100	100	100	100	100	100
		Avg. Time	1.98	2.11	0.9	1.71	1.95	2.12	0.91	1.71
		%Limit	0	0	0	0	0	0	0	0
		GAP%	-	-	-	-	-	-	-	-
		Vars.	274.4	273.8	283.7	284.8	274.4	273.8	283.7	284.8
		Constr.	492.8	491.6	511.4	513.6	492.8	491.6	511.4	513.6
	5	%Opt	100	100	100	100	100	100	100	100
		Avg. Time	0.82	1.31	0.58	0.85	0.83	1.31	0.58	0.86
		%Limit	0	0	0	0	0	0	0	0
		GAP%	-	-	-	-	-	-	-	-
		Vars.	343.3	342.6	353.6	354.7	343.3	342.6	353.6	354.7
		Constr.	613.6	612.2	634.2	636.4	613.6	612.2	634.2	636.4
8	3	%Opt	0	0	10	0	100	100	100	100
		Avg. Time	-	-	260.95	-	1261.73	2129.81	623.84	1296.6
		%Limit	100	100	90	100	0	0	0	0
		GAP%	32.7	41.88	21.78	34.58	-	-	-	-
		Vars.	366.3	365.7	378.4	376.8	366.3	365.7	378.4	376.8
		Constr.	679.6	678.4	703.8	700.6	679.6	678.4	703.8	700.6
	4	%Opt	0	0	50	0	100	100	100	100
		Avg. Time	-	-	188.01	-	1184.41	1438.36	342.02	755.86
		%Limit	100	100	50	100	0	0	0	0
		GAP%	28.3	32.09	15.93	19.05	-	-	-	-
		Vars.	490.1	487.7	504.4	505	490.1	487.7	504.4	505
		Constr.	904.2	899.4	932.8	934	904.2	899.4	932.8	934
	5	%Opt	40	10	90	30	100	100	100	100
		Avg. Time	206.04	140.28	152.23	225.62	380.2	816.46	177.87	410.18
		%Limit	60	90	10	70	0	0	0	0
		GAP%	16.24	24.43	11.73	15.22	-	-	-	-
		Vars.	605.5	608	629	630	605.5	608	629	630
		Constr.	1112	1117	1159	1161	1112	1117	1159	1161
10	3	%Opt	0	0	0	0	0	0	0	0
		Avg. Time	-	-	-	-	-	-	-	-
		%Limit	100	100	100	100	100	100	100	100
		GAP%	60.59	66.05	54.69	61.36	50.66	59.7	45.6	54.54
		Vars.	569.2	570	590.6	589.7	569.2	570	590.6	589.7
		Constr.	1071.4	1073	1114.2	1112.4	1071.4	1073	1114.2	1112.4
	4	%Opt	0	0	0	0	0	0	0	0
		Avg. Time	-	-	-	-	-	-	-	-
		%Limit	100	100	100	100	100	100	100	100
		GAP%	55.65	60.52	50.19	54.7	45.65	52.13	40.12	46.12
		Vars.	757.2	758.5	786.5	787.8	757.2	758.5	786.5	787.8
		Constr.	1418.4	1421	1477	1479.6	1418.4	1421	1477	1479.6
	5	%Opt	0	0	0	0	0	0	0	0
		Avg. Time	-	-	-	-	-	-	-	-
		%Limit	100	100	100	100	100	100	100	100
		GAP%	51.84	61.63	44.7	52.89	41.47	50.29	33.41	41.3
		Vars.	951	947	986.1	985.4	951	947	986.1	985.4
		Constr.	1777	1769	1847.2	1845.8	1777	1769	1847.2	1845.8

Table 3: Results of the MIP model averaged across all replicated instances. Resources 1 and 2 refer to $U[1,3;3,5]$ and, $U[1,5;5,10]$, respectively. Setups 1 and 2 refer to $U[1,50;50,100]$ and $U[50,100;100,150]$, respectively.

n	m	S	R	SPTSA$_-$	SPSTSA$_-$	DJASA$_-$	SPTSA$_+$	SPSTSA$_+$	DJASA$_+$	SPTSA$_{av}$	SPSTSA$_{av}$	DJASA$_{av}$	SPTSA*	SPSTSA*	DJASA$^*_-$	DJASA$^*_+$	DJASA$^*_{av}$
6	3	1	1	57.4	58.5	36.5	46.2	50.6	7.6	52.2	54.6	21.3	37.1	42.7	19.3	7.2	7.2
6	3	1	2	117.1	124.8	72.2	55.5	61.9	12.4	88.7	95.0	52.4	50.5	56.9	38.3	12.4	26.7
6	3	2	1	44.5	47.1	20.7	29.3	36.2	6.6	36.9	41.0	14.3	26.4	31.2	11.2	5.7	7.2
6	3	2	2	127.7	128.9	90.9	65.9	76.6	15.6	95.6	101.7	57.7	63.0	73.8	43.5	15.6	20.5
6	4	1	1	82.5	84.4	31.1	65.4	66.4	9.8	71.7	75.0	23.1	57.9	60.1	23.3	9.8	16.8
6	4	1	2	169.5	155.8	88.8	82.7	89.3	15.8	124.3	123.2	61.4	78.7	80.7	54.9	15.8	20.2
6	4	2	1	128.7	124.5	32.4	110.0	111.0	10.3	116.7	113.3	23.9	104.3	102.0	17.7	9.8	15.7
6	4	2	2	172.3	175.0	80.7	94.0	82.1	20.1	132.7	129.1	60.1	93.2	82.0	41.4	20.1	37.0
6	5	1	1	176.7	165.1	30.7	136.6	120.6	12.1	152.5	140.6	21.7	125.0	112.5	13.5	12.1	13.5
6	5	1	2	337.4	416.1	77.4	204.1	232.5	23.9	270.6	324.9	60.2	199.1	232.5	44.9	23.9	34.0
6	5	2	1	224.7	177.0	37.6	194.7	145.7	16.5	209.3	158.3	25.7	185.6	135.0	26.7	16.3	17.0
6	5	2	2	287.0	244.7	67.5	161.8	155.2	13.4	226.9	206.4	48.1	159.2	150.8	36.7	13.4	17.5
8	3	1	1	53.1	48.9	25.4	42.3	38.9	13.1	46.9	43.5	21.7	33.1	30.1	16.4	10.8	11.5
8	3	1	2	96.5	97.3	66.0	37.5	34.4	15.1	67.8	65.7	46.7	35.1	33.7	33.7	15.1	21.8
8	3	2	1	39.3	38.1	22.3	32.5	27.9	9.8	36.7	33.2	16.3	26.3	24.6	10.4	8.8	7.8
8	3	2	2	90.3	88.1	53.2	47.4	38.4	11.8	69.0	63.6	40.3	45.7	37.4	28.5	11.8	20.9
8	4	1	1	60.7	63.4	30.3	47.5	49.0	10.4	53.4	54.1	22.6	39.2	42.1	16.5	9.4	9.9
8	4	1	2	134.9	135.0	78.1	65.2	65.6	15.6	101.4	100.8	56.8	63.1	63.7	45.2	15.6	21.5
8	4	2	1	54.7	61.9	29.0	43.8	50.7	8.5	48.8	55.8	22.9	38.0	46.5	15.7	8.5	12.7
8	4	2	2	143.7	134.1	76.7	74.8	67.7	14.4	110.5	101.9	53.1	71.8	65.4	40.9	14.4	21.7
8	5	1	1	84.4	75.5	39.4	52.2	50.0	9.7	67.1	60.2	27.4	50.3	46.5	26.3	9.7	13.5
8	5	1	2	186.4	201.0	89.1	96.1	101.0	11.1	144.2	150.0	62.1	90.3	99.2	46.8	11.1	22.0
8	5	2	1	93.9	76.8	30.0	66.5	63.9	10.8	80.7	71.4	24.1	62.9	58.6	19.7	9.8	13.0
8	5	2	2	159.9	144.4	80.6	61.4	62.0	11.2	112.9	104.3	52.6	60.5	61.1	42.3	11.2	21.8
10	3	1	1	36.8	36.0	19.3	31.8	30.5	12.0	34.5	33.4	20.5	19.6	18.9	12.7	9.9	10.5
10	3	1	2	91.2	91.0	57.8	41.3	35.7	13.0	65.3	64.2	44.8	34.6	30.6	34.8	13.0	20.1
10	3	2	1	28.1	24.6	12.0	24.5	21.7	8.1	27.1	21.6	13.2	16.8	14.7	6.9	6.6	5.7
10	3	2	2	62.6	68.3	42.5	32.1	37.1	4.7	47.6	52.7	27.1	29.1	34.4	20.3	4.6	11.0
10	4	1	1	51.2	50.6	27.6	34.9	37.0	11.2	43.7	45.3	21.6	29.5	31.1	17.3	10.1	10.7
10	4	1	2	124.3	118.3	75.5	50.8	49.0	11.8	87.2	83.6	56.6	47.9	45.0	50.2	11.4	22.9
10	4	2	1	44.7	44.6	20.7	42.3	38.7	6.1	43.8	42.3	15.1	32.3	30.5	11.4	5.4	6.1
10	4	2	2	90.1	90.5	61.1	38.6	43.5	10.8	64.5	67.0	41.9	35.6	40.7	32.9	10.8	17.5
10	5	1	1	55.8	59.8	33.0	36.6	35.1	7.8	45.2	47.2	23.8	30.5	30.2	23.4	7.2	11.1
10	5	1	2	163.1	142.1	86.0	93.1	74.0	12.9	126.9	109.1	64.0	89.8	72.0	51.0	12.9	28.2
10	5	2	1	54.7	51.5	26.3	43.5	36.6	4.1	48.9	44.9	16.7	35.8	32.9	13.6	4.1	5.7
10	5	2	2	143.3	140.7	76.6	76.3	78.0	6.7	110.1	109.9	53.4	73.8	77.1	43.6	6.7	21.1
Average				**113.0**	**110.7**	**50.7**	**68.3**	**66.5**	**11.5**	**90.6**	**88.6**	**36.5**	**63.1**	**61.9**	**28.7**	**11.1**	**16.7**

Table 4: Average percentage deviations from MIP model solutions for the 14 tested heuristics. Small instances. S and R represent Setups and Resources, respectively. R=1 and 2 represent $U[1, 50; 50, 100]$ and $U[50, 100; 100, 150]$, respectively. S=1 and 2 represent $U[1, 3; 3, 5]$ and $U[1, 5; 5, 10]$, respectively.

n	m	S	R	SPTSA$_-$	SPSTSA$_-$	DJASA$_-$	SPTSA$_+$	SPSTSA$_+$	DJASA$_+$	SPTSA$_{av}$	SPSTSA$_{av}$	DJASA$_{av}$	SPTSA*	SPSTSA*	DJASA$^*_-$	DJASA$^*_+$	DJASA$^*_{av}$
50	10	1	1	34.9	36.1	3.1	55.5	51.1	16.1	43.7	44.0	16.8	24.7	25.7	0.2	11.7	4.2
50	10	1	2	70.8	69.7	30.9	40.2	39.5	0.3	56.3	55.2	30.1	30.8	30.0	21.5	0.0	12.1
50	10	2	1	25.2	24.9	2.4	31.8	33.0	8.5	29.3	29.6	10.3	17.2	17.4	0.1	4.9	1.6
50	10	2	2	52.4	53.4	19.6	31.8	31.9	0.2	42.0	42.6	18.8	26.2	26.2	12.9	0.0	5.9
50	15	1	1	43.3	43.6	8.2	41.5	41.3	2.2	42.0	42.5	11.7	26.0	25.7	2.9	0.8	1.1
50	15	1	2	110.8	106.9	54.4	57.6	55.7	0.0	84.3	81.2	42.2	50.4	48.5	35.5	0.0	13.2
50	15	2	1	42.9	42.6	7.5	43.9	43.4	1.9	43.3	43.3	8.7	31.8	31.2	4.7	0.5	1.2
50	15	2	2	86.3	88.4	39.7	50.6	54.4	0.0	68.0	71.9	29.9	45.5	48.1	25.9	0.0	9.6
50	20	1	1	71.2	66.3	20.6	58.9	56.2	0.6	65.6	61.5	17.1	45.6	42.2	14.8	0.2	3.7
50	20	1	2	156.9	155.7	75.6	79.8	83.9	0.0	118.5	119.9	53.8	73.1	76.6	47.7	0.0	16.1
50	20	2	1	63.6	60.9	14.6	54.1	53.1	0.4	58.3	56.2	11.4	44.0	43.8	9.2	0.0	2.7
50	20	2	2	130.2	132.4	55.5	69.3	72.2	0.0	99.9	102.7	39.9	65.8	69.0	33.7	0.0	13.0
75	10	1	1	36.4	36.5	1.5	70.9	72.0	34.9	54.3	55.1	27.9	28.9	29.6	0.0	21.2	9.1
75	10	1	2	42.7	41.0	9.0	37.6	37.8	2.3	40.2	40.0	19.0	19.6	19.1	3.4	0.2	3.9
75	10	2	1	22.3	23.2	0.9	39.4	39.0	20.9	30.4	30.8	17.5	17.8	18.9	0.0	12.8	6.0
75	10	2	2	32.8	33.5	5.9	31.0	29.3	2.2	32.1	31.5	12.5	17.4	17.2	1.6	0.8	2.3
75	15	1	1	41.0	37.5	3.0	58.1	54.2	15.2	50.4	47.6	16.5	29.1	25.5	0.5	11.3	3.5
75	15	1	2	73.7	75.7	29.8	46.3	48.4	0.3	59.8	61.9	29.6	34.4	37.5	20.4	0.0	9.8
75	15	2	1	31.8	33.0	1.6	43.7	43.4	9.7	37.4	37.8	11.3	24.6	24.9	0.0	6.2	2.7
75	15	2	2	60.9	61.0	20.7	39.4	40.9	0.1	50.1	50.7	19.4	32.5	33.2	13.2	0.0	6.1
75	20	1	1	45.5	43.1	4.9	48.2	46.1	3.2	47.0	44.4	11.5	28.4	26.0	1.2	1.7	1.0
75	20	1	2	104.3	108.0	48.2	57.8	56.1	0.1	81.3	82.1	40.7	50.1	48.7	31.8	0.0	14.8
75	20	2	1	36.5	38.0	4.1	42.6	41.8	2.4	39.0	40.2	8.2	27.0	27.8	1.1	1.5	0.5
75	20	2	2	86.4	87.3	33.8	54.3	53.6	0.0	70.6	70.7	28.4	47.5	46.5	22.5	0.0	10.2
100	10	1	1	38.0	39.1	0.9	89.0	90.2	56.6	66.3	66.1	41.4	32.7	34.0	0.0	32.8	13.1
100	10	1	2	36.6	37.3	3.3	46.4	46.8	14.7	41.4	42.4	24.6	20.1	20.2	0.0	9.4	6.8
100	10	2	1	25.0	24.7	0.6	52.9	53.3	29.4	38.9	39.1	21.7	22.2	22.2	0.0	16.8	6.8
100	10	2	2	27.6	27.5	2.1	32.0	32.7	8.3	29.9	30.2	15.7	16.3	16.7	0.0	5.1	4.8
100	15	1	1	38.8	38.9	1.6	71.0	68.8	30.8	56.0	54.7	27.4	30.6	30.0	0.0	21.2	9.2
100	15	1	2	57.0	54.7	12.7	43.3	42.9	1.1	50.3	49.3	23.1	27.3	26.7	6.5	0.0	6.9
100	15	2	1	28.9	30.5	1.0	45.3	49.0	17.8	37.5	40.0	15.8	23.7	25.2	0.0	11.9	4.9
100	15	2	2	42.3	43.9	9.4	33.3	34.9	1.0	38.0	39.7	15.7	22.5	23.7	5.0	0.0	4.9
100	20	1	1	41.3	39.8	2.2	59.4	55.7	15.6	49.7	47.9	18.3	29.6	28.4	0.0	11.2	5.1
100	20	1	2	83.0	82.2	29.9	50.5	48.4	0.3	66.9	65.2	31.1	38.6	37.9	20.1	0.0	11.8
100	20	2	1	34.2	36.2	1.5	46.9	49.9	8.8	39.9	43.5	11.2	27.1	29.4	0.0	6.2	1.9
100	20	2	2	62.8	63.1	21.2	42.0	43.0	0.1	52.4	53.4	21.7	34.0	34.4	14.0	0.0	8.7
Average				**56.1**	**56.0**	**16.2**	**49.9**	**49.8**	**8.5**	**53.1**	**53.2**	**22.2**	**32.3**	**32.5**	**9.7**	**5.2**	**6.6**

Table 5: Average percentage deviations from best solutions for the 14 tested heuristics. Large instances. S and R represent Setups and Resources, respectively. R=1 and 2 represent $U[1,50;50,100]$ and $U[50,100;100,150]$, respectively. S=1 and 2 represent $U[1,3;3,5]$ and $U[1,5;5,10]$, respectively.

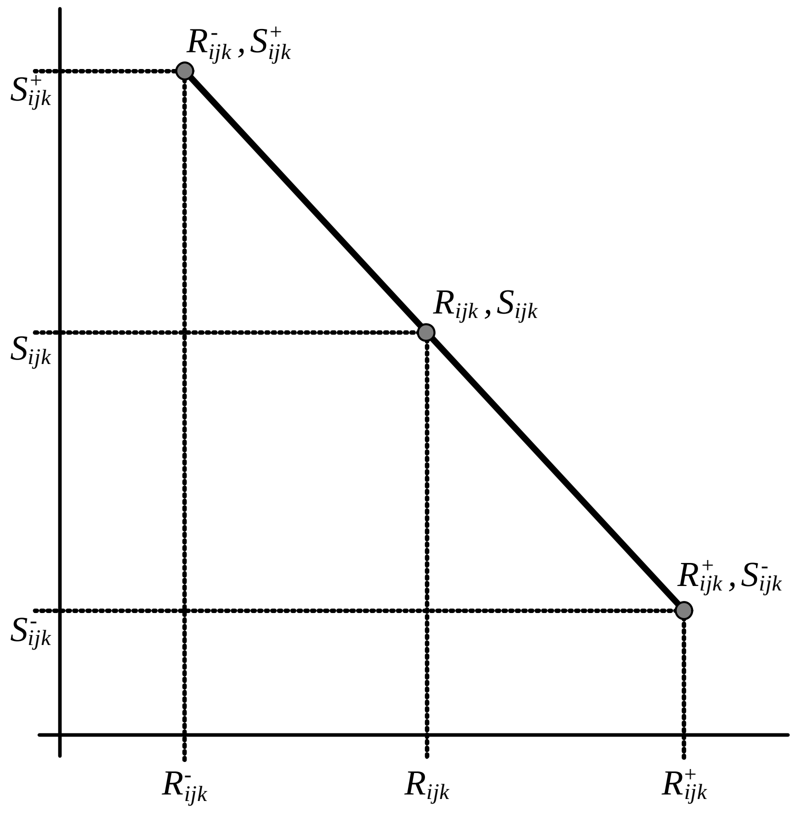

Figure 1: Relation between setup time S_{ijk} and amount of assigned resources R_{ijk}. Notice that R_{ijk} does not need to be a continuous quantity.

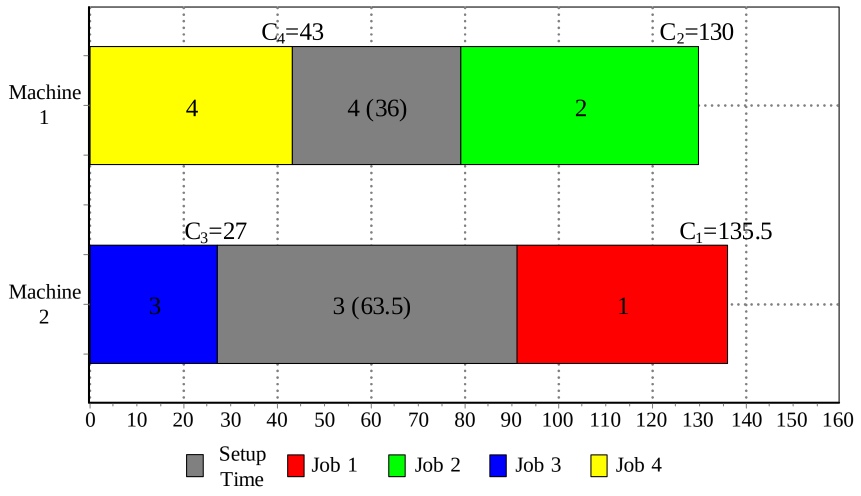

Figure 2: Gantt chart with the solution of the example problem after applying the DJASA$_{av}$ dynamic dispatching rule.

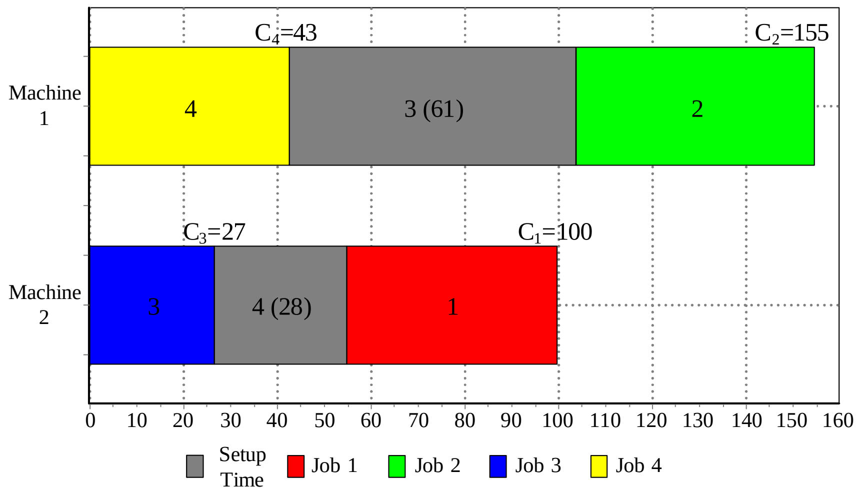

Figure 3: Gantt chart with the solution of the example problem after applying the DJASA$_{av}$ dynamic dispatching rule and the optimal resource assignment procedure (DJASA$^*_{av}$).

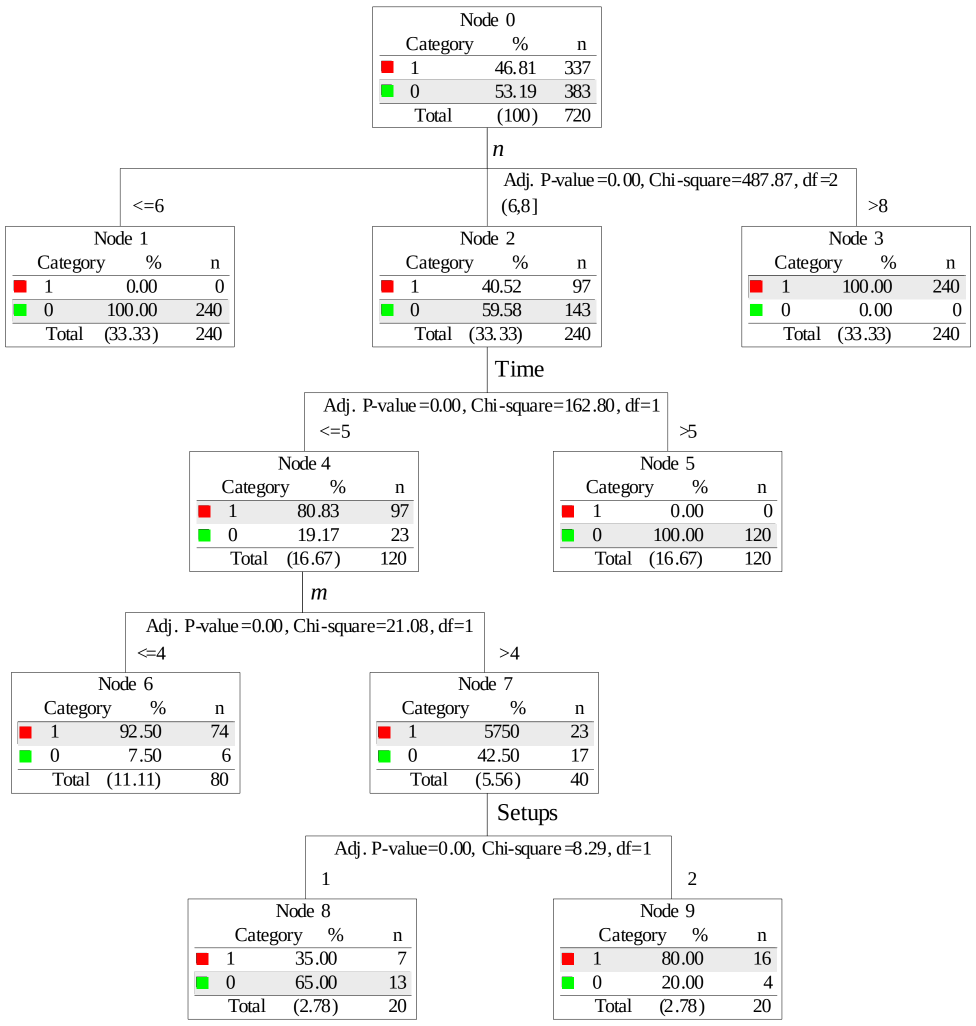

Figure 4: Decision tree of the MIP model results. Response variable type of outcome. Setups 1 and 2 refer to $U[1, 50; 50, 100]$ and $U[50, 100; 100, 150]$, respectively.

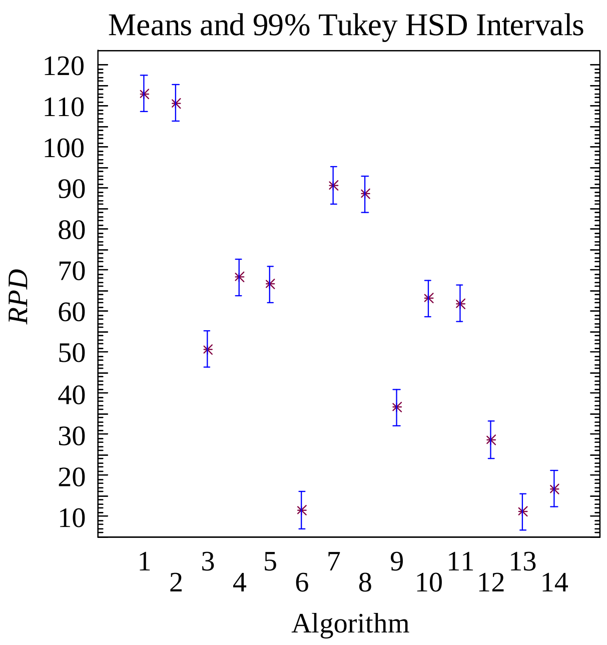

Figure 5: Average RPD Means plot and Honest Significant Difference confidence intervals for the factor type of Algorithm. Small instances.

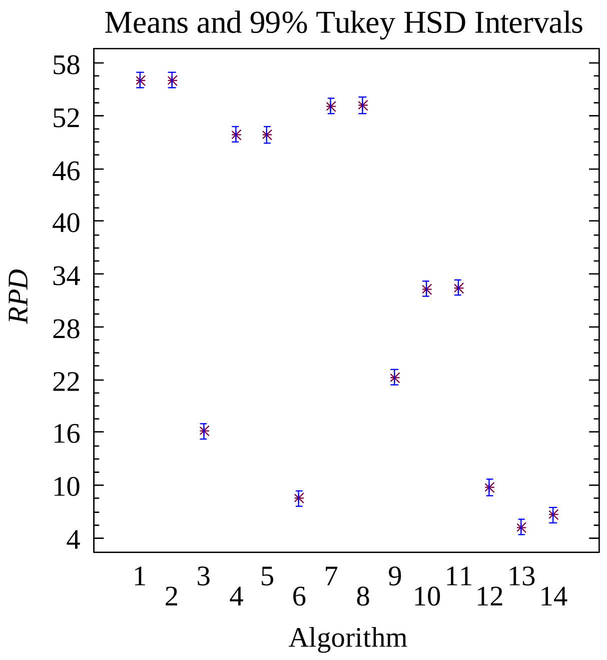

Figure 6: Average RPD Means plot and Honest Significant Difference confidence intervals for the factor type of Algorithm. Large instances.