

Unrelated Parallel Machines Scheduling with Resource-Assignable Sequence Dependent Setup Times

Rubén Ruiz

Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática, Universidad Politécnica de Valencia, Valencia, Spain, rruiz@eio.upv.es

Carlos Andrés

Departamento de Organización de Empresas, Universidad Politécnica de Valencia, Valencia, Spain, candres@omp.upv.es

Setup operations on machines usually need of tooling and/or personnel to be carried out. Therefore, the length of the setup might not only depend on the machine and job sequence, but also on the number of resources assigned. This situation is common in real problems arising in several industries where setup operations in production lines are frequent. These operations are indeed setups whose length can be reduced or extended according to the number of resources assigned to them. We deal with an unrelated parallel machine scheduling problem with a linear combination of total completion time and the total number of resources assigned as an objective. We present a MIP model and some fast dispatching heuristics. We carry out computational experiments to study what characteristics of the problem affect the MIP model performance and to the effectiveness the different heuristics proposed.

Keywords: Machine Scheduling, Production Scheduling, Real World Scheduling, Heuristic Search.

1 Introduction

In this short paper, we schedule n jobs on m unrelated parallel machines. As known, first jobs have to be assigned to machines and second, among the jobs assigned to each machine, a processing sequence must be derived. We denote by p_{ij} the processing time of job j at machine i . 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 . A clear example of setups 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.

S_{ijk} is difficult to set in practice as resources are involved. We will denote by R_{ijk} the amount of resources devoted to carrying out setup S_{ijk} . We refer to this as resource-assignable sequence dependent setup times or RASDST in short. Resources are usually costly and a trade-off situation arises between cost and total setup time. Consequently, we are interested in a realistic optimization criterion which minimizes the total production time or flowtime and the resources employed. We denote by T_{Res} the total number of resources assigned to setups. 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 $\alpha, \beta \geq 0$. Such a problem, is obviously \mathcal{NP} -Hard since it is a generalization of other simpler \mathcal{NP} -Hard problems. For example, the case

where the parallel machines are identical and there are only sequence independent family setup times is already \mathcal{NP} -Hard as explained in [10].

In the literature there are several reviews and surveys about scheduling with setup times like the ones of [1] and more recently, [2]. There are also specific surveys dealing with parallel machine settings like those of [4] and [6]. A careful examination of these reviews results in no paper dealing with the problem considered in this work. A related study is due to [3] where a MIP model for a uniform parallel machines problem and earliness/tardiness criteria is proposed. In [12], a mixed integer programming model is proposed for the unrelated machines, sequence dependent setup problem with earliness/tardiness objective. [11] and [5] study similar problems. More recently, [8] 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.

2 Problem and MIP model formulation

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} . Similarly, $S_{ijk}^-(S_{ijk}^+)$ denote the minimum (maximum) possible setup time. The relation between the actual setup time and the number of resources is assumed to be linear. Therefore, the expression that returns the amount of setup time S_{ijk} as a function of the assigned resources R_{ijk} is the following:

$$S_{ijk} = S_{ijk}^+ - \frac{S_{ijk}^+ - S_{ijk}^-}{R_{ijk}^+ - R_{ijk}^-}(R_{ijk} - R_{ijk}^-) = S_{ijk}^+ + \frac{S_{ijk}^+ - S_{ijk}^-}{R_{ijk}^+ - R_{ijk}^-}R_{ijk}^- - \frac{S_{ijk}^+ - S_{ijk}^-}{R_{ijk}^+ - R_{ijk}^-}R_{ijk}.$$
Using K_1 and K_2 for the first term 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} : $S_{ijk} = K_1 - K_2R_{ijk}$. K_2 is the slope of the line that relates R_{ijk} with S_{ijk} . In other words, K_2 indicates how much the setup time S_{ijk} is reduced by each additional resource. With this we define the following model:

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

$$\min \sum_{i \in M} \sum_{j \in N} \sum_{k \in N, k \neq j} \alpha R_{ijk} + \sum_{i \in M} \sum_{j \in N} \beta C_{ij} \tag{1}$$

$$\sum_{i \in M} \sum_{j \in \{0, N\}, j \neq k} X_{ijk} = 1, \quad k \in N \tag{2}$$

$$\sum_{i \in M} \sum_{k \in N, j \neq k} X_{ijk} \leq 1, \quad j \in N \tag{3}$$

$$\sum_{k \in N} X_{i0k} \leq 1, \quad i \in M \tag{4}$$

$$\sum_{h \in \{0, N\}, h \neq k, h \neq j} X_{ihj} \geq X_{ijk}, \quad j, k \in N, j \neq k, i \in M \tag{5}$$

$$C_{ik} + V(1 - X_{ijk}) \geq C_{ij} + K_1 - K_2R_{ijk} + p_{ik}, \quad j \in \{0, N\}, k \in N, j \neq k, i \in M \tag{6}$$

$$R_{ijk} \geq R_{ijk}^-, \quad j, k \in N, j \neq k, i \in M \tag{7}$$

$$R_{ijk} \leq R_{ijk}^+, \quad j, k \in N, j \neq k, i \in M \quad (8)$$

$$C_{i0} = 0, \quad i \in M \quad (9)$$

$$C_{ij} \geq 0, \quad j \in N, i \in M \quad (10)$$

$$X_{ijk} \in \{0, 1\}, \quad j \in \{0, N\}, k \in N, j \neq k, i \in M \quad (11)$$

Constraint set (2) ensures that every job has exactly one predecessor. Conversely, constraint set (3) limits the number of successors of every job to one. Set (4) limits the number of successors of the dummy jobs to a maximum of one on each machine, With set (5) we ensure that jobs are properly linked in machines since if a given job j is processed on a given machine i , it must have a predecessor h on the same machine. Constraint set (6) controls the completion times of the jobs at the machines. Constraint sets (7) and (8) bound the possible assignable resources. Sets (9) and (10) define completion times as 0 for dummy jobs and non-negative for regular jobs, respectively. Finally, set (11) defines the binary variables.

3 Heuristic methods

Shortest Processing Time (SPT) dispatching rules are known to give optimum solutions for simple scheduling problems with the total completion time ($\sum_{j=1}^n C_j$) criterion (see [7]). Additionally, all seven heuristics proposed in [11] for the $R/S_{jk}/\frac{1}{n} \sum_{j=1}^n w_j C_j$ problem are based on this dispatching rule. It is clear than processing first the shortest jobs results in a lower overall total completion time. Therefore, we also base our proposed heuristics on the SPT rule.

3.1 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

A final issue remains about the quantity of resources to assign to each setup (step 3b). For the moment we assign the average resources $(R_{ijk}^+ + R_{ijk}^-)/2$.

3.2 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 the minimum processing time and average setup time for all machines, i.e., calculate an index I as follows:

$$I_j = \min_{i \in M} \left(p_{ij} + \frac{1}{n-1} \sum_{k=1, k \neq j}^n \frac{S_{ijk}^+ + S_{ijk}^-}{2} \right) \quad (12)$$

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

3.3 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. 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 $\rho \neq \emptyset$ 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, average resources are assigned. For SPTSA and SPSTSA, the overall complexity is $\mathcal{O}(n \log n)$. In DJASA, a total of $\frac{n(n+1)}{2}m$ steps are needed, which results in a $\mathcal{O}(n^2m)$ complexity.

3.4 Optimal resource assignment

Once the assignment of the different jobs to the unrelated parallel machines and the sequence among them is known we can do a better re-assignment of resources. 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:

$$R_{ijk} = \begin{cases} R_{ijk}^+, & \text{if } K_2 \cdot h \cdot \beta > \alpha \\ R_{ijk}^-, & \text{otherwise} \end{cases} \quad (13)$$

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 has a complexity of only $\mathcal{O}(n)$ and can be easily applied to the resulting solution of the previous methods.

4 Computational Evaluation

A comprehensive benchmark has been defined for the evaluation. We generate all the data: processing times p_{ij} and the minimum and maximum resources and setups (R_{ijk}^- , R_{ijk}^+ , S_{ijk}^- and S_{ijk}^+). The MIP model is not expected to be usable for large instances. 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. Two resources settings are considered where the resources are uniformly distributed in the ranges $U[1, 3; 3, 5]$ and $U[1, 5; 5, 10]$, respectively. For the setup times we also work with two combinations $U[1, 50; 50, 100]$ and $U[50, 100; 100, 150]$. A total of 720 instances are generated. 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.

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. Commercial solvers have been used for parallel machine problems with setups by [3] and [12] and for more complex problems by [9]. We solve all small instances with two different time limits, 5 and 60 minutes. 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 and type 1 means that the time limit was reached and at the end a feasible integer solution was found. Due to the large data set, and to obtain sound conclusions, it is important to carry out statistical testing. We have carried out statistical testing with decision trees. This approach was used recently by [9] also for the analysis of a MIP model. The resulting tree is shown in Figure 1. Some direct conclusions can be drawn from the decision tree. 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 time, 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. The effect of the resources is also interesting. Having more resources (i.e., $U[1, 5; 5, 10]$)

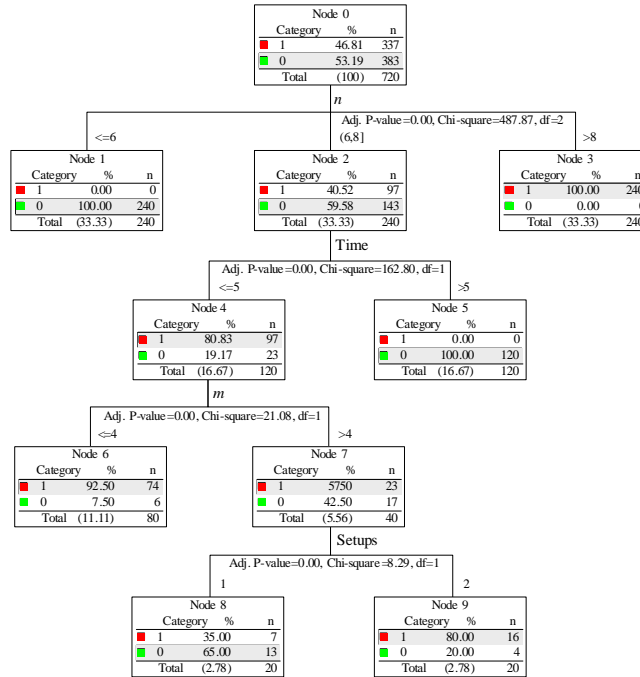


Figure 1: Decision tree of the MIP model results. Response variable type of outcome. Setups 1 and 2 refer to $U[1, 50; 50, 100]$ and $U[50, 100; 100, 150]$, respectively.

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. Overall, the efficiency of the MIP model is good, specially when compared to results from the literature. [3] solved problems of up to $n = 10$ and $m = 4$ but on a simpler setting with uniform parallel machines. Their MIP model contains significantly less variables due to the triangular law of inequality assumption in the setup times, i.e., [3] assume that $S_{ijk} + S_{ikl} \geq S_{ijl}$. Obviously, in our RASDST problem this assumption does not hold. [12] 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 in this latter case. 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.

We proceed now to test the performance of the proposed heuristics. Recall that there are three main methods where we assign average resources: SPTSA, SPSTSA and DJASA. We can apply the optimal resource assignment procedure to these methods which results in three more heuristics: SPTSA*, SPSTSA* and DJASA*. As a result, there are six 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 all instances, where we measure the average percentage increase ($AVRPD$) over the optimum solution (or best solution found) given by the MIP model in the small instances set. For the large instances we test the average percentage deviation over the best solution found by all heuristics. All instances, as well as the best known solutions are available from

<http://www.upv.es/gio/rruiz>. Table 1 gives the *AVRPD* values for the small and large instances. For reasons of space, only global average results are reported. As expected, DJASA is the

	SPTSA	SPSTSA	DJASA	SPTSA*	SPSTSA*	DJASA*
Average small	90.6	88.6	36.5	63.1	61.9	16.7
Average large	53.1	53.2	22.2	32.3	32.5	6.6

Table 1: Average percentage deviations from MIP model solutions for the six tested heuristics. Small and large instances.

best method over SPSTSA and this later one is in turn better than SPTSA. The *AVRPD* values drop strongly for DJASA in all situations. As expected, the optimal resource assignment procedure improves the results always since the procedure cannot deteriorate the solutions. Although not shown, higher number of resources and lower duration of setups results in higher deviations. This is an interesting result since just contradicts the findings of the MIP model. As with the case of the MIP model, the previous table contains observed averages alone. We carried out full experimental analysis using the ANOVA technique. In the experiment we control n , m , R , S and Algorithm as factors. 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 of Table 1 are statistically significant.

A much desirable characteristic of the proposed heuristics is their speed. Among all results gathered with the six 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* 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.

5 Conclusions

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, three dispatching heuristics and an optimal resource assignment rule have been proposed. The results 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 and the future research lines are numerous and promising.

References

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

- [2] A. Allahverdi, C.T. Ng, T.C.E. Cheng and M.Y. Kovalyov (2007), A survey of scheduling problems with setup times or costs, *European Journal of Operational Research*, In press.
- [3] N. Balakrishnan, J.J. Kanet and S.V. Sridharan (1999), Early/tardy scheduling with sequence dependent setups on uniform parallel machines, *Computers & Operations Research* **26**(2), 127 – 141.
- [4] T.C.E. Cheng and C.C.S. Sin (1990), A state-of-the-art review of parallel machine scheduling research, *European Journal of Operational Research* **47**(3), 271 – 292.
- [5] D.W. Kim, K.H. Kim, W. Jang and F.F. Chen (2002), Unrelated parallel machine scheduling with setup times using simulated annealing, *Robotics & Computer Integrated Manufacturing* **18**(3-4), 223 – 231.
- [6] E. Mokotoff (2001), Parallel machine scheduling problems: a survey, *Asia-Pacific Journal of Operational Research* **18**(2), 193 – 242.
- [7] M. Pinedo (2002), *Scheduling: Theory, Algorithms, and Systems*, Prentice Hall, Upper Saddle, N.J, second edition.
- [8] G. Rabadi, R.J. Moraga, and A. Al-Salem (2006), Heuristics for the unrelated parallel machine scheduling problem with setup times, *Journal of Intelligent Manufacturing* **17**(1), 85 – 97.
- [9] R. Ruiz, F. Sivrikaya Şerifoğlu and T. Urlings (2007), Modeling realistic hybrid flexible flowshop scheduling problems, *Computers & Operations Research*, In press.
- [10] S.T. Webster (1997), The complexity of scheduling job families about a common due date, *Operations Research Letters* **20**(2), 65 – 74.
- [11] M.X. Weng, J. Lu and H. Ren (2001), Unrelated parallel machines scheduling with setup consideration and a total weighted completion time objective, *International Journal of Production Economics* **70**(3), 215 – 226.
- [12] Z. Zhu and R. Heady (2000), Minimizing the sum of earliness/tardiness in multi-machine scheduling: a mixed integer programming approach, *Computers & Industrial Engineering* **38**(2), 297 – 305.