A Reformulation for the Problem of Scheduling Unrelated Parallel Machines with Sequence and Machine Dependent Setup Times

Oliver Avalos-Rosales and **Ada Alvarez**
Universidad Autónoma de Nuevo León
Avenida Universidad s/n
San Nicolás de los Garza, N. L., 66450, México
oliver@yalma.fime.uanl.mx, *ada.alvarezs@uanl.mx*

Francisco Angel-Bello
Tecnológico de Monterrey, campus Monterrey
Avenida Eugenio Garza Sada 2501
Monterrey, N. L., 64849, México
fangel@itesm.mx

Abstract

In this paper we propose an improved formulation for an unrelated parallel machine problem with machine and job sequence-dependent setup times that outperforms the previously published formulations regarding size of instances and CPU time to reach optimal solutions. The main difference between the proposed formulation and previous ones is the way the makespan has been linearized. It provides improved dual bounds which speeds up the solution process when using a branch-and-bound commercial solver. Computational experiments show that, using this model, it is possible to solve instances four times larger than what was previously possible using other mixed integer programming formulations in the literature and obtain optimal solutions on instances of the same size up to three orders of magnitude faster.

Introduction

In this work we study a problem of scheduling n jobs on m unrelated parallel machines with the objective of minimizing the makespan, considering setup times that depend both on the machine and the sequence.

The parallel machine scheduling problem has been extensively studied, an interesting survey on parallel machines can be found in (Mokotoff 2001). Most of the literature addresses identical or uniform machines, where the processing time of a job is the same regardless of the machine where it is processed or is proportional to the speed of the machine, respectively.

Less effort has been done for studying the case where the processing time of each job depends on the machine on which it is processed, that is, the machines are unrelated (Cheng, Ding, and Lin 2004). This is a common situation in several applications where there are parallel machines with different capabilities.

From works dealing with unrelated parallel machines, only a few address the problem considering setup times. Most assume that there are no setup costs or they are independent of job sequence, however, this situation may not always be true in practice. A setup is a set of operations that should be performed on a machine after processing a job to prepare it for processing the next one. In various real world industrial/service environments these times are sequence-dependent, that is, depend not only on the job that will be processed, but also on the job processed just before (Lee and Pinedo 1997).

The number of works addressing unrelated parallel machine scheduling problems, with sequence-dependent and machine-dependent setups are even fewer. Despite of this situation appears, for example, in the textile, printed circuit boards and chemical industries (Rabadi, Moraga, and Al-Salem 2006). All that has been previously mentioned motivated us to focus on this problem.

There are several performance criteria to measure the quality of a scheduling. One of the most broadly used is the minimization of the maximum completion time of the schedule, which is known as makespan (C_{max}). It is a very important measure of performance since it gives the total time elapsed in processing all jobs under consideration (De and Morton 1980). The makespan is important in situations when a received batch of jobs is needed to be completed as soon as possible (Allahverdi 2000). This kind of situation is especially common in server farms, data centers, and compute cloud (e.g., the Amazon Elastic Compute Cloud) (Tian et al. 2010).

Garey and Johnson (1979) showed that minimizing the makespan considering two identical machines is a NP-hard problem. Indeed, a problem with unrelated machines and sequence dependent setups is also NP-hard. This, coupled to the fact that re-schedules are often required, the use of exact algorithms is usually not suitable and it is not surprising that many of the methodologies that have been developed are based on heuristics.

Regarding the unrelated parallel machine scheduling problem (UPMSP) with sequence and machine dependent setup times and makespan minimization objective, some heuristic algorithms have been developed recently. Helal, Rabadi, and Al-Salem (2006) propose a Tabu Search algorithm; Rabadi, Moraga, and Al-Salem (2006) introduce a new metaheuristic, MetaRaSP, which incorporates randomness within priority rules to construct a feasible solution. More recently, Arnaout, Rabadi, and Musa (2010) propose an ant colony algorithm; Ying, Lee, and Lin (2012) develop a simulated annealing approach which incorporates a restricted search strategy; Vallada and Ruiz (2011) present a genetic algorithm which exhibits a new crossover operator including a local search procedure; Fleszar, Charalambous, and Hindi (2012) propose a variable neighborhood descent heuristic hybridized with mathematical programming.

Only a few works develop exact methods to solve the problem addressed in this paper. Tran and Beck (2012) propose a Benders decomposition-based method for minimizing the makespan, while Rocha et al. (2008) present a branch and bound approach for minimizing the makespan plus the weighted tardiness.

Problem Description

The following assumptions and notations are used to describe the problem:

- There is a set M of m parallel machines.
- Machines are continuously available, and each machine can handle one job at a time without preemption, that is, once the processing of a job has started, it cannot be interrupted.
- There is a set N of n jobs to be scheduled.
- All the jobs are available at time zero. No precedence constraints among jobs are imposed.
- Each job j has associated a processing time p_{ij} in each machine i.
- There is a setup time s_{ijk} of the machine i for processing job k just after job j.
- The objective is to minimize the makespan C_{max}. Using the term span to denote the completion time of a machine, the makespan denotes the maximum span in the solution of the problem.

Fig. 1 shows a graphical representation of a solution to the addressed problem with 17 jobs and 3 machines. In the figure, the blank blocks mean setup times, which are asymmetric, and the gray blocks mean processing times.

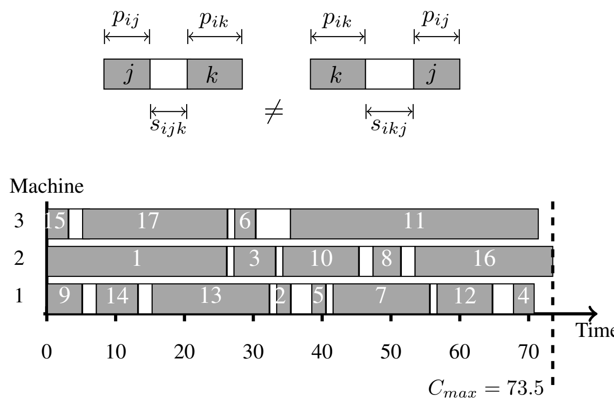

Figure 1: Graphical representation of a solution.

Mathematical Model

In this section we present a model for the UPMSP with sequence-dependent and machine-dependent setup times, with the objective of minimizing the makespan. Other models can be found in (Rabadi, Moraga, and Al-Salem 2006; Vallada and Ruiz 2011).

For describing our model, let us introduce the following variables.

$$X_{ijk} = \begin{cases} 1, & \text{if job } j \text{ is scheduled before job } k \text{ in machine } i, \\ 0, & \text{otherwise.} \end{cases}$$

C_j : Completion time of job j.

O_i : Completion time of machine i (*span*).

We will denote by N_0 the set N plus a dummy job 0 and V denotes a very large number. The variables X_{i0k} and X_{ij0} are used to specify which jobs will be processed at first and at the end, respectively. While the processing time and setup times asociated to the dummy job are zero ($p_{i0} = 0$, $s_{i0k} = 0$ and $s_{ij0} = 0$).

The model can be stated as

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

Subject to:

$$\sum_{i\in M}\sum_{\substack{j\in N_0\\ j\neq k}} X_{ijk} = 1, \qquad \forall k \in N, \tag{2}$$

$$\sum_{i\in M}\sum_{\substack{k\in N_0\\ j\neq k}} X_{ijk} = 1, \qquad \forall j \in N, \tag{3}$$

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

$$\sum_{\substack{k\in N_0\\ k\neq j}} X_{ijk} - \sum_{\substack{h\in N_0\\ h\neq j}} X_{ihj} = 0, \quad \forall j \in N, \quad \forall i \in M, \tag{5}$$

$$C_k - C_j + V(1 - X_{ijk}) \geq s_{ijk} + p_{ik}, \quad \forall j \in N_0, \forall k \in N, j \neq k, \quad \forall i \in M, \tag{6}$$

$$C_0 = 0, \tag{7}$$

$$\sum_{\substack{j\in N_0\\ j\neq k}}\sum_{k\in N} (s_{ijk} + p_{ik})X_{ijk} = O_i, \qquad \forall i \in M, \tag{8}$$

$$O_i \leq C_{max}, \qquad \forall i \in M, \tag{9}$$

$$X_{ijk} \in \{0, 1\}, \quad \forall j \in N_0, \forall k \in N, j \neq k, \quad \forall i \in M, \tag{10}$$

Objective (1) minimizes the makespan of the solution. Constraints (2) establish that every job has exactly one predecessor, while constraints (3) establish that every job has exactly one sucessor. Constraints (4) ensure that at most one job is scheduled as the first job on each machine. Constraints (5) are the so-called "flow conservation constraints". They ensure that if a job is scheduled in a machine, then a predecessor and a successor must exist in the same machine. Constraints (6) provide a right processing order, avoiding loops. Basically they establish that, if $X_{ijk} = 1$, then the completion time of job k must be greater than the completion time of job j . If $X_{ijk} = 0$, the constraint becomes redundant. Constraint (7) sets to zero the completion time of the dummy job. In conjunction with constraints (6) it guarantees that completion time of all jobs is positive. Constraints (8) compute, for each machine, the time it finishes processing its last job. Constraints (9) define the maximum completion time. Finally, constraints (10) define the binary variables.

The proposed model has n^2m binary variables, $n + m$ continuous variables and $n^2m + nm + 2n + 3m$ constraints. In what follows we will denote it by MILM$_{new}$

It should be remarked that constraints (8) and (9) distinguish our model from previously published formulations (Rabadi, Moraga, and Al-Salem 2006; Vallada and Ruiz 2011), where it is common to linearize the makespan as

$$C_j \leq C_{max}, \quad \forall j \in N, \tag{11}$$

the maximum of the completions times of the jobs.

Let us denote MILM$_{prev}$ the mixed integer linear model obtained by replacing contraints (8)-(9) by constraints (11).

When we tried to solve MILM$_{prev}$ or other previous models using the Branch and Bound algorithm (B&B), we observed that the lower bound yielded by the linear relaxation was very weak. This motivated us to propose a different representation for the objective function. This representation was obtained from the analysis of the structure of the solutions to the linear relaxation and the mixed integer model.

Specifically, solving the linear relaxations for different data instances, we observed that the values of the C_j are near to zero. With these values for these variables, also the objective function value is near to zero.

Taking into account what was observed, we decided to linearize the makespan as the maximum of the spans of the machines, by adding the constraints (8)-(9) to the model MILM$_{prev}$ instead of (11).

It is not difficult to see that any feasible solution satisfies these constraints, which means that they are valid inequalities to the model. On the other hand, constraints (8) compute easily the time at which each machine processes its last job (O_i), as the sum of the processing time for each job plus the setup time for the subsequent job. In addition, they do not depend on the large constant V and force O_i to take a positive value if any of the variables X_{ijk} is positive. It also forces C_{max} to take a positive value.

With the proposed linearization, the lower bound provided by the linear relaxation was considerably improved and this allowed accelerating the solution process based on B&B.

In the following section the performance of MILM$_{new}$ is evaluated.

Computational Results

The previous formulations have the same linearization of the makespan. Eventhough they have different constraints, these formulations have a similar performance, as can be seen in the results reported in (Vallada and Ruiz 2011; Tran and Beck 2012). To test the performance of the new proposed linearization we only compare MILM$_{new}$ and MILM$_{prev}$, which differ on the makespan linearization.

We used two sizes of instances: small and medium. Small instances were taken from (Vallada and Ruiz 2011) and are available at http://soa.iti.es. Medium instances were generated for conducting the experiments in this work, since we could not find any in the literature.

For the set of small instances there are four groups of machine numbers (2,3,4,5) and four groups of job numbers (6,8,10,12). The setup times were uniformly distributed in four ranges: 1-9, 1-49, 1-99 and 1-124. The processing times were uniformly distributed between 1-99. For each of the 64 groups, there are 10 instances, making a total of 640 small instances.

We generated medium instances in a similar way as Vallada and Ruiz (2011) generated the small instances. There are the following number of jobs and number of machines: n = $\{15,20,25,30,35,40\}$ and $m = \{2,3,4,5\}$. The setup times were uniformly distributed in three ranges: 1-49, 1-99 and 1-124. The processing times were uniformly distributed between 1-99. For each of the 72 groups, there are 10 instances, making a total of 720 medium instances.

The experiments were conducted on a Pentium Dual Core PC with a 2.00 GHz and 3 GB RAM processor, under Ubuntu 11.1. The formulation was implemented using the concert technology of CPLEX 12.2. The solver was allowed to run for one hour. If the solver was unable to reach the optimal solution within this time, the best integer solution found is reported.

Instances are grouped by number of machines and number of jobs. Therefore, results are averaged over all instances belonging to each group, that is, 40 for small-size groups and 30 for medium-size groups.

Table 1 shows results comparing formulations with different linearizations of the makespan, that is, MILM$_{new}$ and MILM$_{prev}$. Columns 1 and 2 refer to the size of the instances in terms of number of jobs and number of machines, respectively. Entries in columns 3, 4 and 5 exhibit how many instances were unsolved to optimality for each group of instances (#Uns), the average solution time (Time, in seconds) and the average gap (GAP), respectively, for the MILM$_{prev}$. Entries in columns 6, 7 and 8 exhibit how many instances were unsolved to optimality for each group of instances, the average solution time consumed and the average gap, respectively, for the MILM$_{new}$. The GAP for each instance is reported by CPLEX as the relative diference between the integer solution found and the proven best possible objective solution value;

$$GAP = 100 * \frac{best\_obj\_int - best\_lower\_bound}{best\_obj\_int},$$

where $best\_obj\_int$ is the objective value of the best feasible solution found and the $best\_low\_bound$ is the best lower bound found, both obtained from the solver.

The model MILM$_{prev}$ is able to obtain the optimal solution for all the instances with six and eight jobs. Regarding the 10 jobs instances, is able to optimally solve all the instances with three, four and five machines and 6 over 40 instances with two machines. For 12 jobs case, is able to optimally solve all the instances with five machines and 2 over 40, 12 over 40 and 37 over 40, instances with two, three

n	m	MILM$_{prev}$			MILM$_{new}$		
		#Uns	Time	GAP	#Uns	Time	GAP
6	2	0	0.385	0.0	0	0.096	0.0
	3	0	0.239	0.0	0	0.174	0.0
	4	0	0.227	0.0	0	0.208	0.0
	5	0	0.248	0.0	0	0.210	0.0
8	2	0	11.516	0.0	0	0.198	0.0
	3	0	4.871	0.0	0	0.336	0.0
	4	0	1.327	0.0	0	0.395	0.0
	5	0	0.852	0.0	0	0.373	0.0
10	2	6	1217.743	1.5	0	0.340	0.0
	3	0	148.933	0.0	0	0.630	0.0
	4	0	22.809	0.0	0	0.769	0.0
	5	0	6.591	0.0	0	0.725	0.0
12	2	38	3507.183	48.7	0	0.644	0.0
	3	28	3035.808	16.9	0	0.949	0.0
	4	3	747.822	1.6	0	2.103	0.0
	5	0	103.244	0.0	0	1.855	0.0

Table 1: Comparison between MILM$_{prev}$ and MILM$_{new}$ in small instances.

and four machines, respectively. Note that in this case the results show a large average CPU time and GAP's. The performance of the model MILM$_{prev}$ is similar to the previous approaches in literature.

Using the proposed model MILM$_{new}$ the average time to reach optimal solutions are smaller than three seconds for all the small instances.

Table 2 shows the performance of MILM$_{new}$ on medium instances. The model MILM$_{prev}$ is not included in this experiment, since it shows a similar performance than the previous models and they are not able to solve instances with 20 machines in 10800 seconds (Tran and Beck 2012). Columns 1 and 2 refer to the size of the instances in terms of number of jobs and number of machines, respectively. Entries in column 4 shows the average solution time (in seconds) and column 5 exhibit the average GAP.

As can be observed this model is capable of solving to optimality all the 15-jobs and 20-jobs instances, 99.1% of the 25-jobs instances and 86.6% of the 30-jobs instances, 75% of the 35-jobs instances and 74.1% of the 40-jobs instances.

Concluding Remarks

In this paper we have discussed the unrelated parallel machine scheduling problem with sequence and machine dependent setup times. We developed a mixed integer linear model for minimizing the makespan. The main difference between this model and previously published formulations is that the makespan has been linearized as the maximum of the completion times of the machines. It provides improved dual bounds which speeds up the solution process when using a branch-and-bound comercial solver.

Computational experiments on a set of 1360 instances ranging from 6 to 40 jobs indicate the superiority of the proposed model, since it can solve to optimality instances up to four times larger than other formulations using a similar CPU time and obtain optimal solutions on instances of the same size up to three orders of magnitude faster. Therefore, this model could help to extend the size of instances that could be solved by exact methods developed for this problem. In addition, the way for linearizing the makespan could be used in other scheduling problems having the same objective function if a good representation of the spans is obtained, for example, in the case of a single machine Ángel-Bello et al. (2011) obtained improved dual bounds to the span deriving a valid inequality.

n	m	MILM$_{new}$		
		#Opt	Time	GAP
15	2	30	1.2	0.0
	3	30	10.1	0.0
	4	30	12.1	0.0
	5	30	9.4	0.0
20	2	30	2.2	0.0
	3	30	22.2	0.0
	4	30	127.8	0.0
	5	30	203.0	0.0
25	2	30	15.5	0.0
	3	30	68.9	0.0
	4	30	344.8	0.0
	5	29	1006.8	0.1
30	2	30	32.7	0.0
	3	29	406.2	0.0
	4	27	1294.7	0.1
	5	18	2274.0	1.1
35	2	30	45.9	0.0
	3	30	328.7	0.0
	4	21	1998.0	0.4
	5	9	3148.9	3.3
40	2	30	96.1	0.0
	3	30	687.5	0.0
	4	24	1942.0	0.2
	5	5	3408.2	3.8
All	720	662		

Table 2: Performance of the MILM$_{new}$ on medium instances.

Acknowledgments

This work has been partly supported by the Research Chair in Industrial Engineering of Tecnológico de Monterrey (ITESM Research Fund CAT128) and the Mexican National Council of Science and Technology (grant CB 167019)

References

Allahverdi, A. 2000. Minimizing mean flowtime in a two-machine flowshop with sequence-independent setup times. *Computers & Operations Research* 27(2):111–127.

Ángel-Bello, F.; Álvarez, A.; Pacheco, J.; and Martínez, I. 2011. A single machine scheduling problem with availability constraints and sequence-dependent setup costs. *Applied Mathematical Modelling* 35(4):2041–2050.

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

Cheng, T.; Ding, Q.; and Lin, B. 2004. A concise survey of scheduling with time-dependent processing times. *European Journal of Operational Research* 152(1):1–13.

De, P., and Morton, T. 1980. Scheduling to minimize makespan on unequal parallel processors. *Decision Sciences* 11(4):586–602.

Fleszar, K.; Charalambous, C.; and Hindi, K. 2012. A variable neighborhood descent heuristic for the problem of makespan minimisation on unrelated parallel machines with setup times. *Journal of Intelligent Manufacturing* 23(5):1949–1958.

Garey, R., and Johnson, D. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Series of Books in the Mathematical Sciences. W. H. Freeman.

Helal, M.; Rabadi, G.; and Al-Salem, A. 2006. A tabu search algorithm to minimize the makespan for the unrelated parallel machines scheduling problem with setup times. *International Journal of Operations Research* 3(3):182–192.

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

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

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

Rocha, P.; Ravetti, M.; Mateus, G.; and Pardalos, P. 2008. Exact algorithms for a acheduling problem with unrelated parallel machines and sequence and machine-dependent setup times. *Computers & Operations Research* 35(4):1250–1264.

Tian, K.; Jiang, Y.; Shen, X.; and Mao, W. 2010. Makespan minimization for job co-scheduling on chip multiprocessors. Technical Report WM-CS-2010-08, Department of Computer Science, College of William & Mary.

Tran, T., and Beck, J. 2012. Logic-based benders decomposition for alternative resource scheduling with sequence dependent setups. In *20th European Conference on Artificial Intelligence*, 774–780.

Vallada, E., and Ruiz, R. 2011. A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research* 211:612–622.

Ying, K.; Lee, Z.; and Lin, S. 2012. Makespan minimization for scheduling unrelated parallel machines with setup times. *Journal of Intelligent Manufacturing* 23(5):1795–1803.