COMPARATIVE STUDY OF HEURISTIC ALGORITHMS FOR PARALLEL MACHINE SCHEDULING WITH SEQUENCE-DEPENDENT SETUPS AND RELEASE TIMES

Vanessa Patricia Manotas Niño
Universidad de La Sabana, Chía (Cundinamarca), Colombia, vanessa.manotas@unisabana.edu.co

Elyn Lizeth Solano Charris
Universidad de La Sabana, Chía (Cundinamarca), Colombia, erlyn.solano@unisabana.edu.co

Jairo Rafael Montoya Torres
Universidad de La Sabana, Chía (Cundinamarca), Colombia, jairo.montoya@unisabana.edu.co

RESUMEN

Este trabajo considera la aplicación de algoritmos heurísticos para analizar los resultados obtenidos para el problema de programación de operaciones en máquinas paralelas con tiempos de llegada y de preparación dependientes de la secuencia. Dado que este problema es conocido por ser NP-duro en sentido fuerte para la minimización del makespan, lo resolvimos con diferentes heurísticas y sus desempeños se comparan con una solución obtenida usando una cota inferior. Estas heurísticas se basan en reglas de despacho o estrategias de generación aleatoria de los programas. Los experimentos computacionales se realizan utilizando instancias conocidas en la literatura. Los resultados muestran que las heurísticas se desempeñan muy bien comparadas con la solución obtenida con la cota inferior.

Palabras Clave: Programación de Operaciones, Máquinas en paralelo, Tiempos de Preparación, Heurísticas, Generación Aleatoria.

ABSTRACT

This paper considers the application of heuristic algorithms to analyze the results obtained for the parallel machine scheduling problem with release times and sequence-dependent setups times. Since this problem is known to be strongly NP-hard for the case of minimizing the makespan, we resolved it with different heuristics and their performance is compared with a solution obtained using a lower bound. Such heuristics are based on either dispatching rules or random schedule generation strategies. Computational experiments are performed using random-generated data taken from the literature. Our results show that the heuristics perform very well compared against the lower bound, and requiring short computational time.

Keywords: Scheduling, Parallel Machines, Setup Times, Heuristics, Randomness.

1. INTRODUCCIÓN

The assignment of scarce production resources to competing activities over time, in order to optimize certain performance criteria is known as production scheduling (Pei Chen et al., 2006). Generating a feasible schedule that best meets management's objectives is a difficult task that manufacturing firms face every day (Ozgur and Brown, 1995) and efficient production schedules can result in substantial improvements in productivity and cost reductions. In many industries, the decision to manufacture multiple products on common resources results in the need for changeover and setup activities, representing costly disruptions to production processes. Therefore, setup reduction is an important feature of the continuous improvement program of any manufacturing, and even service,

organization, and can be defined as the time required to prepare the necessary resource (e.g., machines, people) to perform a task (e.g., job, operation). Setup times can be of two types: sequence-independent and sequence-dependent. If setup time/cost depends solely on the task to be processed, regardless of its preceding task, it is called sequence-independent. On the other hand, in the sequence-dependent type, setup time depends on both the task and its preceding task (Allahverdi and Soroush, 2008).

In this paper, we consider the problem of production scheduling with sequence-dependent setup times, which can be found in various production, service, and information processing systems (Allahverdi and Soroush, 2008). For example, in a computer system application, a job requires a setup time to load a different compiler if the current compiler is not suitable. In a printing industry, a setup time is required to prepare the machine (e.g., cleaning) which depends on the colour of the current and immediately following jobs. In a textile industry, setup time for weaving and dying operations depends on the jobs sequence. In a container/bottle industry, setup time relies on the sizes and shapes of the container/bottle, while in a plastic industry different types and colours of products require setup times. Similar situations arise in chemical, pharmaceutical, food processing, metal processing, paper industries, and many other industries/areas.

As stated by (Allahverdi and Soroush, 2008), in today's manufacturing scheduling problems it is of significance to efficiently utilize various resources. Treating setup times separately from processing times allows operations to be performed simultaneously and hence improves resource utilization. The benefits of reducing setup times include (Allahverdi and Soroush, 2008): reduced expenses, increased production speed, increased output, reduced lead times, faster changeovers, increased competitiveness, increased profitability and satisfaction, enabling lean manufacturing, smoother flows, broader range of lot sizes, lower total cost curve, fewer stock-outs, lower inventory, lower minimum order sizes, higher margins on orders above minimum, faster deliveries, and increased customer satisfaction. The importance and benefits of incorporating setup times in scheduling research has been investigated by many researchers, see for instance works of (Krajewski et al., 1987), (Flynn, 1987), (Kogan and Levner, 1998), (Liu and Chang, 2000), (Trovinger and Bohn, 2005).

This paper studies the problem of identical parallel machines with sequence-dependent setup times and unequal release times. Formally, the problem can be described as follows: A set of n jobs has to be executed on one and only one of a set of m identical machines arranged in parallel. Job j, with j= 1, n, is characterized by its integer processing time pj and a non-negative integer release date rj . There are setup times dependent on the processing sequence of jobs on machines, but independent on the machines themselves. That is, on any of the parallel machine if job k is executed on the machine immediately after job j, a setup time sjk is needed. We consider the objective of minimizing the makespan (maximum completion time of all jobs) of the schedule. Using the classical notation in Scheduling Theory, this problem is noted at (Pm|rj, sjk|Cmax).

This problem has been studied in the literature. (Guinet, 1993) proposed a mathematical formulation to minimize the makespan or the total completion time of jobs. Heuristics and meta-heuristics procedures have been proposed for several objective functions, such as due-date related objectives, e.g. (Sivrikaya and Ulusoy, 1999), (Bilge et al., 2004), (Pfund et al., 2008) or flowtime related objectives, e.g. (Webster and Azizoglu, 2001), (Abdekhodaee and Wirth, 2002), (Abdekhodaee et al., 2004). (Guinet, 1993) also suggested that the makespan minimization problem when all jobs have equal release dates (rj = 0,□j), is equivalent to the Vehicle Routing Problem (VRP) with service times. The problem with jobs arriving at different release dates (rj ≥ 0,□j), has been little studied in the literature, to the best of our knowledge. (Nessah et al., 2077) considered the objective of minimizing total completion time of jobs (problem $(P_m|r_j, s_{jk}|\sum_{j=1}^{n} C_j)$. The problem under study in this paper, (Pm|rj, sjk|Cmax), has only been studied by (Kurz and Askin, 2001) and (Montoya et al., 2010). The first authors proposed several heuristics algorithms, including multiple insertions and a genetic algorithm, and also derived a data-dependent lower bound for the makespan criterion. Their compared their heuristics between them and but they neither computed the optimal makespan nor compare the performance of their heuristics against the optimum nor the lower bound. The second authors proposed heuristics procedure based on random generation of schedules.

In this paper we compare two different heuristics. The first procedure, which is based on the hybridization of various simple dispatching rules, is called MOPAM (Makespan Optimization for Parallel Machines). The second

procedure, called RI (that stands for Random Insertion heuristic), is based on the random generation of job execution sequences. Our aim is to analyze the performance of both heuristics when solving the problem of identical parallel machines with sequence-dependent setup times and unequal release times.

The remainder of this paper is organized as follows. Section 2 presents the description of the heuristics. Section 3 shows the computational experiments. The paper ends in section 4 by presenting some concluding remarks.

2. DESCRIPTION OF HEURISTIC ALGORITHMS

This section describes the heuristic algorithms analyzed in this paper. The first heuristic called MOPAM is based on the combination of various dispatching rules to determine the sequencing order of jobs, while the second algorithm is based on the idea of using random generation of processing sequences.

2.1 MOPAM ALGORITHM

The first algorithm presented in this paper is a new procedure based on the hybridization of various simple dispatching rules for solving the problem Pm|rj, sjk|Cmax. The heuristic was developed taking into consideration rules for prioritizing jobs according to the processing time pj of each job j, the release dates rj of each job j, and the setup times sjk. The idea behind the algorithm is to mix several classical dispatching rules so as to prioritize jobs when designing the execution sequence. The algorithm is described next in detail in figure 1.

MOPAM Algorithm

Initialization

Enter the number n of jobs.
For each job, enter its processing time p_j and its release date r_j.
Enter the setup times S_{jk} for each pair of jobs j and k, with $j \neq k$.

Algorithm

1. List all jobs by increasing order of the release dates r_j, that is, using First-in-first-out (FIFO) rule. Ties are broken by increasing order of processing time p_j, that is, using the Longest Processing Time First (LPT) rule.
2. At any time a machine is free, schedule the first job in the list defined in step 1. If more than one machine is available at the same time, the job is assigned to the machine for which the lower setup time S_{jk} is required.
3. Compute the makespan of the schedule (C_{max}) once all jobs have been executed.

Figure 1. MOPAM Algorithm

2.2 RANDOM-INSERTION (RI) HEURISTIC

This algorithm for problem Pm | rj, sjk | Cmax is based on a random insertion strategy, in which random numbers are generated from an equilikely distribution between 1 and n, in order to define the position of a job in the schedule. A certain number of iterations are required so as to improve the initial solution (schedule) (Montoya et al., 2010). The algorithm is described in detail in figure 2.

Random-Insertion Algorithm

Initialization

1. Enter the number n of jobs.
2. Enter the number m of identical parallel machines.
3. For each job, enter its processing time p_j and its release date r_j.
4. Enter setup times s_{jk} for each pair of jobs j and k, with $j \neq k$.
5. Define the number of iterations (*niter*).

Algorithm

6. Compute the number of jobs to be scheduled on the machines. For the first (*m-1*) machines, this bound is computed as $\lfloor n/m \rfloor$. The *m*-th machine has assigned the other jobs.
7. Set $h = 1$, the first iteration.
8. Generate an integer random number *R* from an equilikely distribution between 1 and *n*.
9. Schedule job R on the first machine with available positions. If this job has already been assigned, repeat from step 8.
10. Repeat from step 8 until all jobs have been scheduled.
11. Ensuring that release dates are respected, compute $C_{\max}^{h}$, the makespan for the schedule of iteration *h*.
12. Do $h = h+1$ and repeat from step 8 while $h \leq niter$ (that is, until the number of iterations is reached).
13. Select the schedule with $min_h\ C_{\max}^{h}$ (that is, select the schedule with minimum makespan over all the iterations).

Figure 2. Random-Insertion algorithm

3. COMPUTATIONAL EXPERIMENTS

This section presents the computational environment and the analysis of the results of our experiments.

3.1 COMPUTATIONAL ENVIRONMENT AND BENCHMARK INSTANCES

Experiments were conducted on a PC with Windows XP operating system with a Pentium Dual-Core 1.73GHz bi-processor. Both heuristic algorithms were programmed using Visual Basic for Applications (VBA) for MS Excel®.

Experiments were carried out using the same instances than in (Montoya et al., 2010). Those instances was inspired by the structure proposed by (Chu, 2003) and later extended by (Nessah et al., 2007) to consider setup times. Integer processing times were generated from a uniform distribution [1,100]. Integer release dates were generated using a uniform distribution [0; α x n], where n is the number of jobs to be scheduled and α is a real number with values 0.6, 1.5 and 3.0. Integer setup times were generated from a uniform distribution [0; min Pj]. Five instances for each of value of α were generated. Problems with 10, 20, 50 or 100 jobs were considered. We considered here configurations with 3 or 5 identical parallel machines. A full factorial experimental design gave a total of 120 testing scenarios. For the case of Random-Insertion heuristic, its random behavior required running 10 replications for each instance scenario and the best sequence (i.e., the sequence with minimum value of the makespan) was registered as the final solution.

3.2 RESULTS AND ANALYSIS

Table 1 show, for the two heuristics, MOPAM, and RI compared with the LB, the average value of the best makespan (C_{max}) for each group of instances. The lower bound is thus computed as presented in (Kurz and Askin, 2001):

$$LB_{C_{max}}^{Pm} = max\ \{LB1, LB2\} \quad (1)$$

Where *LB1* and *LB2* are, respectively, computed as:

$$LB1 = \frac{1}{m}\left\{\sum_{k=1}^{n}\left[p_j + \min_{j\in\{1,\ldots,n\}} s_{jk}\right]\right\} \quad (2)$$

$$LB2 = \max_{j}\left\{r_j + p_j + \min_{k\in\{1,\ldots,n\}} s_{jk}\right\} \quad (3)$$

We can see that the computational time increases with the number of jobs and decreases with the number of machine regardless the algorithm type. We can also see that the proposed method is able to provide good solutions comparing with the Lower Bound and fast performance.

Table 1. Average Makespan

# Jobs	Algorithm Type				Lower Bound	
	Random-Insertion		MOPAM			
	# Machines					
	m = 3	m = 5	m = 3	m = 5	m = 3	m = 5
10	218.4	172.2	233.7	177.5	217	163.8
20	433.5	314.2	451	577.1	512.9	258.4
50	916.3	569.5	927	578.2	866.9	520.1
100	1796.2	1114.2	1815.8	1103.4	1741.1	1044.7

Table 2 summarizes the performance of the Heuristics MOPAM and RI with the LB (Equations (4) and (5)) and the performance of the MOPAM with the RI (Equation (6)). The performance of proposed heuristic was computed as the deviation obtained from the solutions obtained with the RI and the LB:

$$\%dev = \frac{(C_{max}^{RI} - C_{max}^{LB})}{C_{max}^{LB}} \times 100\% \quad (4)$$

$$\%dev = \frac{(C_{max}^{MOPAM} - C_{max}^{LB})}{C_{max}^{LB}} \times 100\% \quad (5)$$

$$\%dev = \frac{(C_{max}^{RI} - C_{max}^{MOPAM})}{C_{max}^{MOPAM}} \times 100\% \quad (6)$$

Table 2. Average deviation

# Jobs	% dev of the Heuristics with LB				% dev Between Heuristics	
	RI		MOPAM			
	m = 3	m = 5	m = 3	m = 5	m = 3	m = 5
10	9.4	6.1	16.5	9.3	7.1	3.2
20	0.4	2.8	4.3	22.4	4	0.6
50	5.7	9.5	6.8	11	-12	-5.4
100	3.2	6.7	4.5	5.9	1	-1
Average	4.7	6.3	8.0	12.2	0.0	-0.7

For further precise analysis of the results, we conduct an analysis of variance (ANOVA) to studying the effects of three independent factors: number of jobs (10, 20, 50, 100), algorithm type (RI, MOPAM) and number of machines (3, 5) on a response dependent factor which measure the average value of the optimal makespan. Table 3, summarizes the results of ANOVA test. The results indicate that both number of jobs and number of machines factors and their interaction has statistically significant effect on optimal makespan at the 95% confidence level. The forms of the interaction are shown in Figure 1. The plots are worked out using Statgraphics Plus®.

Table 3. ANOVA

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A: # Jobs	5.22712E6	3	1.74237E6	356.09	**0.0000**
B: Algorithm Type	18442.1	2	9221.05	1.88	0.2317
C: # Machines	521118.	1	521118.	106.50	**0.0000**
AB	15571.1	6	2595.18	0.53	0.7701
AC	402817.	3	134272.	27.44	**0.0007**
BC	8126.86	2	4063.43	0.83	0.4804
Residual	29358.1	6	4893.02		
Total (corrected)	6.22256E6	23			

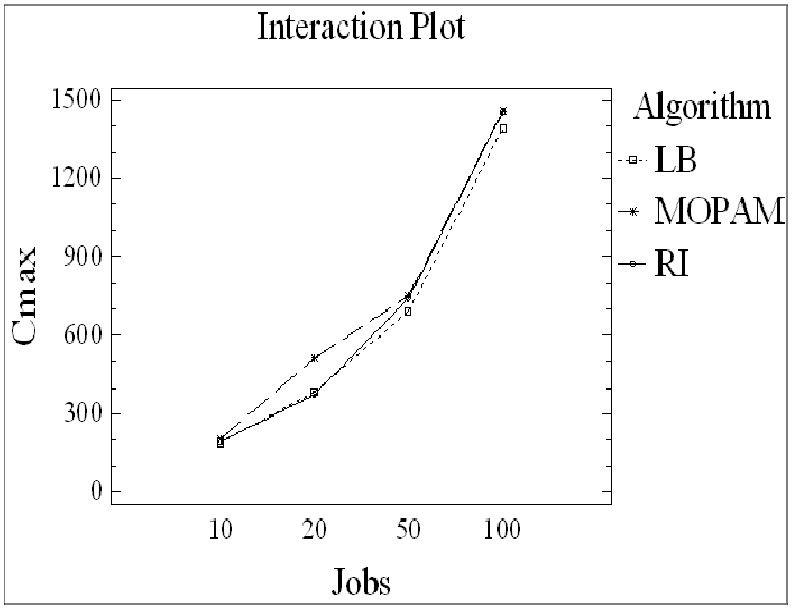

Figure 3. Interaction plot between algorithm type and number of jobs

To analyze the behavior of the different algorithms in the different situation, we performed paired *t*-test to compare difference between means. We use a significance level of 5% and we assume that the alternative hypothesis for this test is the differences between means C_{max} of the heuristics are unequal.

All results derived are listed in Table 4 and 5. The results show that, there is a significant difference between mean values of makespan of heuristics and the lower bound for large-sized instances, while no significant difference between them is present for the case of small-sized instances. Regardless of the number of jobs, the mean of the heuristics is significantly different.

For the case of differences between average makespan with m = 5 machines, the mean difference between both heuristic and the LB is significant, while no difference exists when comparing the MOPAM algorithm with the RI algorithm.

Table 4. T test results for the differences between means with m = 3

# Jobs			
	LB - RI	**LB - MOPAM**	**RI - MOPAM**
10	0.943	0.39	0.0027
20	0.299	0.416	0.017
50	3.78E-07	5.47E-08	0.010
100	3.25E-04	3.76E-04	0.00296

Table 5. T test results for the differences between means with m = 5

# Jobs			
	LB - RI	**LB - MOPAM**	**RI - MOPAM**
10	0.067	0.03	0.219
20	2.88E-08	5.74E-07	0.789
50	5.66E-09	4.83E-07	0.038
100	2.91E-06	1.11E-04	0.0516

We observed that studying the differences between mean makespan values for two treatments are interesting. Hence, it becomes important to perform a deeper analysis of makespan variability. For that reason, we performed Fisher test to compare difference between variances. As previously, we use a significance level of 5%.and assume that the alternative hypothesis is that variances are unequal.

Table 6. F test results for the differences between variances with m = 3

# Jobs			
	LB - RI	**LB - MOPAM**	**RI - MOPAM**
10	6.52E-06	1.18E-04	0.187
20	9.71E-09	1.27E-07	0.238
50	0.326	0.276	0.436
100	0.489	0.457	0.467

Table 7. F test results for the differences between variances with m = 5

# Jobs			
	LB - RI	**LB - MOPAM**	**RI - MOPAM**
10	0.033	0.370	0.065
20	0.148	0.041	0.238
50	0.364	0.137	0.224
100	0.409	0.410	0.327

When making a comparison between variances with m = 3 machines, we observed that there is no difference between the variances of makespan values of the heuristics, but there is a significant difference between the heuristics and the lower bound values for small instances. For m = 5 machines, we did not find sufficient statistical evidence to conclude that the variances are different.

4. CONCLUDING REMARKS

This paper considered the problem of scheduling jobs on identical parallel machines environments subject to release times and sequence-dependent setup times. Setup times, defined in general as the time required to prepare the necessary resource to perform a job, add complexity for the analysis of scheduling problems. Because this problem is strongly NP-hard, this paper proposed the performance analysis of two heuristic algorithms. The first one is a new algorithm called MOPAM that takes into consideration several dispatching rules on the process time p_j of jobs, their release dates r_j, and setup times s_{jk}. The second algorithm was based on randomness to generate various execution sequences. Computational experiments were performed in order to analyze the performance of both heuristic against each other and against a lower bound, suing random-data taken from the literature. This experiment showed that the newly presented heuristic MOPAM gives good schedules with a makespan value that is between about 7% and 10% of the lower bound solution, regardless of the number of jobs. For future works, the advantages of using simple dispatching rules or randomly-generated schedules might be better explored on the resolution of other complex scheduling problems.

ACKNOWLEDGMENTS

The work presented in this paper was performed under research grant EICEA-53-2010 from Universidad de La Sabana, Colombia.

REFERENCES

Abdekhodaee, A.H. and Wirth, A. (2002). "Scheduling parallel machines with a single server: some solvable cases and heuristics". Computers and Operations Research. Vol 29. pp. 295-315.

Abdekhodaee, A.H., Wirth, A. and Gan H.S. (2004). "Equal processing and equal setup time cases of scheduling parallel machines with a single server". Computers and Operations Research. Vol 31. pp. 1867-1889.

Allahverdi, A. and Soroush, H.M. (2008). "The significance of reducing setup times/setup costs". European Journal of Operational Research. Vol. 187. pp. 978-984.

Bilge, U., Kirac, F., Kurtulan, M. and Pekgun, P. (2004). "A tabu search algorithm for parallel machine total tardiness problem". Computers and Operations Research. Vol 31. pp. 397-414.

Chen, K. P., Lee, M. S., Pulat, P. S. and Moses, S. A. (2006). "The shifting bottleneck procedure for job-shops with parallel machines". International Journal of Industrial and Systems Engineering. Vol. 1. pp 244-262.

Chu, C. (2003). "Efficient heuristics to minimize total flow time with release dates". Operations Research Letters. Vol. 12. pp. 321-330.

Flynn, B.B. (1987). "The effects of setup time on output capacity in cellular manufacturing, International". Journal of Production Research. Vol. 25. pp. 1761-1772.

Guinet, A. (1993). "Scheduling sequence-dependent jobs on identical parallel machines to minimize completion time criteria". International Journal of Production Research. Vol 31. pp. 1579-1594.

Kogan, K. and Levner, E. (1998). "A polynomial algorithm for scheduling small-scale manufacturing cells served by multiple robots". Computers and Operations Research. Vol. 25. pp. 53-62.

Krajewski, L.J., King, B.E., Ritzman, L.P. and Wong, D.S. Kanban. (1987). "MRP and shaping the manufacturing environment". Management Science. Vol. 33. pp. 39-57.

Kurz, M.E. and Askin, R.G. (2001). "Heuristic scheduling of parallel machines with sequence-dependent set-up times". International Journal of Production Research. Vol 39. pp. 3747-3769.

Liu, C.Y. and Chang, S.C. (2000). "Scheduling flexible flow shops with sequence-dependent setup effects". IEEE Transactions on Robotics and Automation. Vol. 16. pp. 408-419.

Montoya-Torres, J.R., Soto-Ferrari, M. and Gonzalez-Solano, F. (2010). "Production scheduling with sequence dependent setups and j ob release times". Dyna. Vol 163. pp. 260-269.

Nessah, R., Chu, C. and Yalaoui, F. (2007). "An exact method for $Pm|r_j, s_{jk}|\sum_{j=1}^{n} C_j$) problem". Computers and Operations Research. Vol 34. pp. 2840-2848.

Ozgur, C.O. and Brown, J.R. "A two-stage traveling salesman procedure for the single machine sequence-dependent scheduling problem, Omega". The International Journal of Management Science. Vol. 23. 1995. pp. 205-219.

Pfund, M., Fowler, J.W., Gadkari, A. and Chen, Y. (2008). "Scheduling jobs on parallel machines with setup times and ready times". Computers and Industrial Engineering. Vol 54. pp. 764-782.

Sivrikaya-Serifoglu, F. and Ulusoy, G. (1999). "Parallel machine scheduling with earliness and tardiness penalties". Computers and Operations Research. Vol 26. pp. 773-787.

Trovinger, C. S. and Bohn, R.E. (2005). "Setup time reduction for electronics assembly: Combining simple (SMED) and IT-based methods". Production and Operations Management. Vol. 14. pp. 205-217.

Webster, S. and Azizoglu, M. (2001). "Dynamic programming algorithms for scheduling parallel machines with family setup times". Computers and Operations Research. Vol 28. pp. 127-137.

Authorization and Disclaimer

Dear Editor,

We hereby affirm that the content of this manuscript is original. Furthermore the manuscript has not been neither published elsewhere fully or partially or any language nor submitted for publication (fully or partially) elsewhere simultaneously.

We have read the "Author Instructions" of the conference and all authors are agree with its contents and assigning the publication rights to LACCEI Proceedings.