Generating instances for the two-stage multi-machine assembly scheduling problem

Carla Talens[1], Victor Fernandez-Viagas[1] and Paz Perez-Gonzalez[1]

Industrial Management, School of Engineering, University of Seville
cartafa@us.es, vfernandezviagas@us.es, pazperez@us.es

Keywords: Benchmark, Assembly, Total Completion Time.

1 Introduction and description of the problem

The two-stage assembly scheduling problem consists of sequencing n jobs in a layout composed of two stages. Each job has $m_1 + 1$ operations. In the first stage, there are m_1 dedicated parallel machines, in which the first m_1 operations are conducted, while in the assembly stage there are m_2 identical parallel machines, being $m_2 \geq 1$. Only after all m_1 operations are completed, the assembly operation may start in an assembly machine. A job j has a processing time p_{ij} on machine i in the first stage and an assembly processing time at_j. The problem under study consists of scheduling the jobs in each machine so the total completion time is minimized.

In this contribution, we focus on the development of two new benchmarks for the two-stage assembly scheduling problem because two different variants of the problem are studied. The first one consists of several dedicated parallel machines in the first stage and one assembly machine in the second stage, and, following the notation by Framinan et al. (2019), it is denoted as $DP_{m_1} \rightarrow 1||\sum C_j$. In the second one, there are m_2 identical parallel machines in the last stage with $m_2 \geq 1$. It is denoted as $DP_{m_1} \rightarrow P_{m_2}||\sum C_j$.

2 New benchmark: parameters and generation procedure

In this section, we detail the characteristics, the parameters, and the generation procedure of the two new benchmarks. According to the works by Hall and Posner (2001) and Vallada et al. (2015), the next characteristics should be taken into account in the new proposed benchmarks: *adequacy, hardness, exhaustiveness* and *amenability for statistical analysis.*

2.1 Parameters

The first testbed, denoted as $\mathcal{B}_1$, is a set of 240 instances with the following combinations of number of jobs (n), number of machines in the first stage (m_1) and number of machines in the second stage (m_2): $n \in \{50, 100, 150, 200, 250, 300\}$, $m_1 \in \{2, 4, 6, 8\}$ and $m_2 = 1$. The second testbed, denoted as $\mathcal{B}_2$, is a set of 960 instances where $n \in \{50, 100, 150, 200, 250, 300\}$, $m_1 \in \{2, 4, 6, 8\}$ and $m_2 \in \{2, 4, 6, 8\}$. The values have been taken based on Al-Anzi and Allahverdi (2006) and Allahverdi and Al-Anzi (2012). Both testbeds have 10 associated instances for each combination. Note that, we have selected both a wide range of levels and equidistant values for the number of jobs and machines to fulfil the exhaustiveness and amenability requirements.

2.2 Adequacy

With respect to the adequacy characteristic, we should generate instances which exactly suit the problem under study and not related problems. In our case, the instances should

be representative of the two-stage multi-machine assembly scheduling problem. Therefore, the relationship between this problem and the related scheduling problems, such as the Customer Order (CO) scheduling problem and the Parallel Machines (PM) scheduling problem has to be analysed. To perform this analysis, we carry out a preliminary experiment which lead us to generate the processing times in the first stage from a uniform distribution $U[1, 100]$ and in the second stage from a uniform distribution $U[1, \alpha 100]$. Parameter α is designed to balance the connection between both stages. The next values of α are tested: $\alpha = \{0.5, 1, 2, 3\}$.

We generate two preliminary small testbeds. The first one, testbed A, consists of 540 instances with $n \in \{8, 10, 12\}$, $m_1 \in \{2, 4\}$ and $m_2 = 1$, and, the testbed B consists of 1,080 instances with $n \in \{8, 10, 12\}$, $m_1 \in \{2, 4\}$ and $m_2 = \{2, 3\}$. Then, we solve them applying exact methods in order to identify representative instances of the problem under study and the related problems. To do so, firstly, we have adapted to our problem the mathematical model found in the paper by Navaei et al. (2013). Then, we have modified it to solve the CO scheduling problem. And, finally, we take the SPT rule plus the First Available Machine rule to solve the PM scheduling problem. Therefore, we hold three exact methods to solve the three different scheduling problems.

Solving each instance by the three methods we obtain the optimal sequences for each case denoted as MMA^*, CO^* and PM^*. The evaluation of the objective function of these schedules for the problem MMA are denoted as $MMA(MMA^*)$, $MMACO^*$ and $MMA(PM^*)$. These values allow us to analyse the relationship between the problem under study and the related problems (CO and PM).

To determine how is the relationship between the three problems we calculate the Relative Percentage Deviation (RPD) of the $MMACO^*$ and $MMA(PM^*)$, as follows:

$$RPD_{hs} = \frac{C_{hs} - C_s^*}{C_s^*} \cdot 100 \tag{1}$$

with C_{hs} the total completion time obtained by $h \in (MMA(CO^*), MMA(PM^*))$ in instance s ($s = 1, \ldots, S$) and C_s^* the minimum completion time known for instance s. Then, the ARPD is computed as the average of all the instances.

Low values of ARPD mean that the problems are highly connected, i.e. the fact that $ARPD_{MMA(CO^*)}$ is low means that the instances can be solved by methods of the CO scheduling problem. Contrarily, high values of ARPD denote that the problem under study is not related to the other problem.

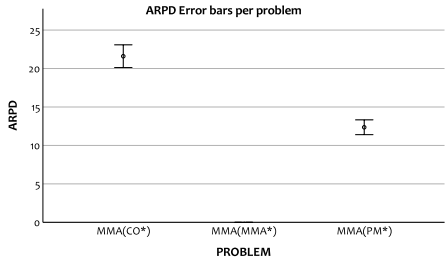

Fig. 1: ARPD of exact methods when $m_2 = 1$ and $\alpha = 2$

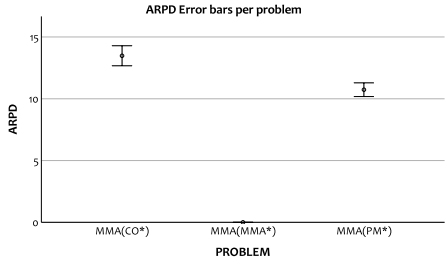

Fig. 2: ARPD of exact methods when $m_2 \geq 2$ and $\alpha = 2$

For the $DP_{m_1} \rightarrow 1||\sum C_j$ problem, Figures 1 and 2 show the relationship with CO and PM when $\alpha = 2$, since the ARPD in both cases are high (greater than 10), so the

instances are representative of our problem. When $\alpha = 0.5$, the instances seem to be more representative of the CO scheduling problem and, when $\alpha = 3$, more representative of the PM scheduling problem. The same results are obtained for the $DP_{m_1} \rightarrow P_{m_2}||\sum C_j$ problem as it can be seen in Figure 2.

Next, we should verify that the behaviour of these instances are also fulfilled when the number of jobs and machines increases. As we can not solve bigger instances applying exact methods, we solve the preliminary small testbeds by an approximate method, and then, we compare these results with those obtained solving bigger instances with the same approximate method. In this case, we apply the heuristic by Framinan and Perez-Gonzalez (2017), adapted to $DP_{m_1} \rightarrow P_{m_2}||\sum C_j$ in the case with more than one machine in the second stage.. As it can be seen in Figures 3 and 4, when the number of jobs, n, increases, the instances have the same behaviour as when n is low.

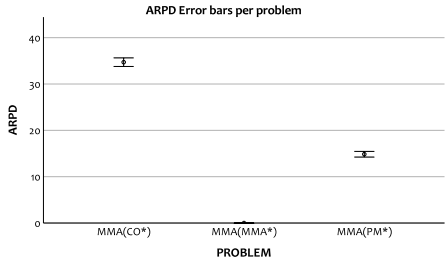

Fig. 3: ARPD of the heuristic when $m_2 = 1$ and $\alpha = 2$

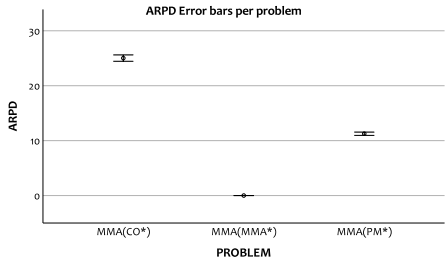

Fig. 4: ARPD of the heuristic when $m_2 \geq 2$ and $\alpha = 2$

2.3 Empirical hardness

In order to determine the empirical hardness, we use the following approach which is based on the difference between a well-performing metaheuristic and a bound of the problem (see Taillard, 1990 and Vallada et al., 2015). For each combination of n, m_1 and m_2, we generate 1,000 instances. So, in total 120,000 instances are generated: 24,000 for the combinations when $m_2 = 1$ and 96,000 for the combinations when $m_2 \geq 2$. 10 instances are selected according to the following procedure: To test the difficulty of each instance by, firstly, computing a lower bound and, secondly, solving it by the Iterated Greedy, IG, algorithm developed by Ruiz and Stützle (2007). We have adapted the lower bound developed by Blocher and Chhajed (2008), which is computed by Equation 2, to the problem under study, where $w_j = \sum_{\forall i} \frac{p_{ij}}{m_1}$. The IG works as follows: The NEH heuristic (Nawaz et al., 1983) gives the initial solution of the IG. The central procedures are the destruction and the construction phases. Then, a local search procedure is carried out by improving each solution generated in the construction phase. Finally, the new sequence is accepted or not as the incumbent solution for the next iteration by a Simulated Annealing.

$$LB = \sum_j \left(max\{ \sum_{\forall k \leq j} \lceil w_j \rceil, max_{\forall i} \sum_{\forall k \leq j} p_{ij} \} \right) + \sum_{\forall j} p_{jA} \tag{2}$$

We set a stopping criterion depending on the size of the instance and the complexity of computing the objective function (the total completion time) of this problem: $n \cdot m/2 \cdot 90/1000$ milliseconds, where m is equal to $m_1 + 1$. Then, we obtain, for each instance, the

ARPD of the IG with respect to the lower bound. Following the idea by Vallada et al. (2015), the higher the $ARPD_{IG}$, the harder the instance is, i.e. if the solution founded by the IG is further from the theoretical lower bound, the instance is hard to solve.

So, the procedure to obtain the hardest instances per combination is as follows: the $ARPD_{IG}$ for the 1,000 instances are sorted in descending order. Then, the 10 first instances per combination are selected to be part of the new benchmark. As a result, a new benchmark with 240 instances ($DP_{m_1} \rightarrow 1||\sum C_j$) and another one with 960 instances ($DP_{m_1} \rightarrow P_{m_2}||\sum C_j$) are generated.

Regarding the exhaustiveness, the benchmark consists of a large number of instances, 240 and 960 respectively, and different size of the parameters and different combinations have been considered. Finally, the benchmark is amenable for statistical analysis since the levels of the parameters are equidistant and all the levels have been combined to generate the instances.

Acknowledgements

This research has been funded by the Spanish Ministry of Science and Innovation, under the project "PROMISE" with reference DPI2016-80750-P.

Bibliography

Al-Anzi, F. S. and Allahverdi, A. (2006). A Hybrid Tabu Search Heuristic for the Two-Stage Assembly Scheduling Problem. *International Journal of Operations Research*, 3(2):109–119.

Allahverdi, A. and Al-Anzi, F. (2012). A new heuristic for the queries scheduling problem on distributed database systems to minimize mean completion time. In *Proceedings of the 21st International Conference on Software Engineering and Data Engineering, SEDE 2012*.

Blocher, J. D. and Chhajed, D. (2008). Minimizing customer order lead-time in a two-stage assembly supply chain. *Annals of Operations Research*, 161(1):25–52.

Framinan, J. M. and Perez-Gonzalez, P. (2017). The 2-stage assembly flowshop scheduling problem with total completion time: Efficient constructive heuristic and metaheuristic. *Computers and Operations Research*, 88:237–246.

Framinan, J. M., Perez-Gonzalez, P., and Fernandez-Viagas, V. (2019). Deterministic assembly scheduling problems: A review and classification of concurrent-type scheduling models and solution procedures. *European Journal of Operational Research*, 273:401–417.

Hall, N. and Posner, M. (2001). Generating experimental data for computational testing with machine scheduling applications. *Operations Research*, 49(6):854–865.

Navaei, J., Fatemi Ghomi, S. M. T., Jolai, F., Shiraqai, M. E., and Hidaji, H. (2013). Two-stage flow-shop scheduling problem with non-identical second stage assembly machines. *International Journal of Advanced Manufacturing Technology*, 69(9-12):2215–2226.

Nawaz, M., Enscore Jr., E., and Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, 11(1):91–95.

Ruiz, R. and Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177(3):2033–2049.

Taillard, E. (1990). Some efficient heuristic methods for the flow shop sequencing problem. *European Journal of Operational Research*, 47(1):65–74.

Vallada, E., Ruiz, R., and Framinan, J. (2015). New hard benchmark for flowshop scheduling problems minimising makespan. *European Journal of Operational Research*, 240(3):666–677.