Heuristics for the Distributed blocking Flow shop scheduling problem

Ribas I[1], Companys R[2]

[1] Departament d'Organització d'Empreses
Universitat Politècnica de Catalunya, BarcelonaTech.
Barcelona, Spain
e-mail: imma.ribas@upc.edu

[2] Escola Politècnica Superior d'Edificació
Universitat Politècnica de Catalunya, BarcelonaTech.
Barcelona, Spain
e-mail: ramon.companys@upc.edu

Keywords: distributed blocking flow shop; distributed permutation flow shop; heuristics

1. Introduction

Distributed manufacturing is a common situation for large enterprises that compete in a globalized market. Due to the current globalization trends, production has shifted from single factory production to a multi-factory production network (Behnamian & Fatemi Ghomi, 2014). In this environment, the scheduling problems deal with the allocation of jobs to factories and the scheduling of jobs in each plant. Since the flow shop configuration is the most common processing layout, the flow shop scheduling problem has been studied greatly since the seminal paper of Johnson (1954). However, its extension to a multi-plant environment was first presented by Naderi & Ruiz (2010), referring to it as the Distributed Permutation Flow Shop Scheduling Problem (DPFSP). According to these authors, the DPFSP is defined as a set N of *n* jobs must be processed by set G of F identical factories. Each factory has the same set M of *m* machines. The processing times of all the tasks of a given job do not change from factory to factory. The objective is to minimize the maximum makespan among factories.

After the publication of Naderi & Ruiz (2010), several authors ((Fernandez-Viagas & Framinan, 2014; Gao, Chen, Deng, & Liu, 2012; Gao, Chen, & Deng, 2013; Gao, Chen, & Liu, 2012; Gao & Chen, 2012; Lin, Ying, & Huang, 2013; Liu & Gao, 2010; Naderi & Ruiz, 2014; Wang, Wang, Liu, & Xu, 2013; Xu, Wang, Wang, & Liu, 2013)) proposed various heuristics to solve this problem, but the blocking constraint has been considered in any of them. The blocking flow shop scheduling problem allows many productive systems to be modeled when there are no buffers between consecutive machines. In general, it is useful for those systems that have a production line without a drag system that forces a job to be transferred between two consecutive stations at pre-established time. Some industrial examples can be found in the iron and steel industry (Gong, Tang, & Duin, 2010); in the treatment of industrial waste and the manufacture of metallic parts (Martinez, Dauzère-Pérès, Guéret, Mati, & Sauer, 2006); or in a robotic cell, where a job may block a machine while waiting for the robot to pick it up and move it to the next stage (Sethi, Sriskandarajah, Sorger, Blazewicz, & Kubiak, 1992). The blocking constraint tends to increase the completion time of jobs, because the processed job cannot leave the machine if the next machine is busy. Therefore, the heuristics designed to schedule jobs in this environment have to consider this fact in order to minimize the idle time of machines due to possible blockage. The distributed blocking flow shop scheduling problem (DBFSSP) deals with the allocation and scheduling of jobs in a multi-factory production network with the blocking constraint present in the manufacturing system. To the best of our knowledge, no paper dealing with the distributed blocking permutation flow shop has been published until now. Hence, it is interesting to study this problem in order to design specific procedures, since the adaptation of those designed for the DPFSP probably perform worse than those procedures that consider its characteristics.

In this paper we compare the performance of three variants of two types of heuristics; iterated greedy algorithm (IGA) and iterated local search algorithms (ILS). The variants of each algorithm consist of three constructive procedures to build the initial solution, two of them created specifically for the problem under consideration.

2. Initial solution procedures

The DBPFSP needs to deal with two related decisions: the allocation of jobs to factories and the sequence of jobs assigned to each plant. In this research we have used three constructive heuristics named TR2, HPF23 and RC1_m to generate the initial solution of the heuristics. Each constructive procedure uses a different approach to build the solution.

The TR2 firstly generates a sequence of jobs by ordering them according to the trapezium rule (Companys, 1966), next the jobs are assigned to each plant according to the allocation rule (2) defined in Naderi & Ruiz (2010) that consist of assigning job *j* to the factory which completes it at the earliest time.

The HPF23 generates the sequence according to the HPF2 rule (Ribas & Companys, 2015) and then the sequence is divided in F (number of factories) fractions by assigning a similar load ($\Sigma P_i/F$) to each plant. Finally, the sequence of jobs assigned to each plant is improved by an insertion procedure similar to that used in the second step of NEH (Nawaz, Enscore Jr, & Ham, 1983).

Finally, in RC1_m instead of sequencing the jobs first and then allocate them to the plants, the jobs and factories are considered together. The first step is to select the first job to be assigned to each plant which is done according to the bicriteria index $R(i)$, equation (1). This index considers the contribution of the job to the completion time (minimum sum of its processing times, P_i) and the front delay generated. The selected job is the one whose R(i) value is smaller.

$$R(i) = \lambda \cdot \frac{2 \cdot \sum_{j=1}^{m} (m-j) \cdot p_{j,i}}{m-1} + (1-\lambda) \cdot \sum_{j=1}^{m} p_{j,i} \qquad (1)$$

Next, a factory is first selected in order to proceed with the other jobs. The factory selected is the one which has the last machine available sooner. After the plant is selected, a job is chosen according to index $ind2(f,i)$, calculated as in (2), where f is one of the factories and σ_f is the sequence of jobs already sequenced in plant f.

$$ind2(f,i) = \mu \cdot \left(\sum_{j=1}^{m} (C_{j,k+1}(\sigma_f * i) - c_{j,k}(\sigma_f) - p_{j,i})\right) + (1-\mu) \cdot \sum_{j=1}^{m} p_{j,i} \qquad (2)$$

3. Heuristics for the DBFSP

In this research we have designed an Iterated local Search algorithm (ILS) and an Iterated Greedy Algorithm (IGA) to solve the DBFSP. For each one, we have implemented three variants, each one uses one of the constructive procedures presented in the previous section.

The ILS is composed of four components: the *Initial Solution procedure*, a *Perturbation Mechanism* that modifies the current solution σ leading to an intermediate solution σ', a *Local Search* procedure that returns an improved solution σ'', and an *Acceptance Criterion* that decides to which solution the next perturbation is applied.

The *Perturbation Mechanism* implemented selects randomly a job, from one of the factories and inserts it in the best position of another plant again randomly selected. This procedure is repeated *d* times.

The *Local Search* is divided in two parts; the first part a job is randomly selected from the plant which has the maximum makespan. This job is inserted in the best position of a plant randomly selected. If the maximum makespan among all plants has diminished the sequence is kept and the procedure is repeated with the plant that has now the maximum makespan; else a new job of the critical plant is selected and the process is restarted. This part is finished when all jobs of the critical plant had been selected or after a limited number of iteration is reached. The second part swaps two jobs, one from the critical plant and another from another plant selected randomly and if the maximum makespan among plant has diminished the change is kept. This process is repeated in the same way of the first part. These two parts are repeated while the maximum makespan improves.

The *Acceptance Criterion* uses a procedure similar to the one used in the simulated annealing algorithm.

The IGA has a similar outline than the ILS. The difference between both is the *Perturbation Mechanism*. In this algorithm, *d* jobs are selected and extracted from the plant to which are

assigned. Next, each job is tried to be inserted in each plant and it is assigned to the plant which leads to the minimum makespan.

3.1. Computational evaluation

We compared the three versions of ILS and the three of IGA in order to analyze their behavior. We named each one with the name of the constructive procedure used to build the initial solution plus "_IGA" or "_ILS" respectively. The test was done on the Taillard's (Taillard, 1993) instances adapted to the multi plant environment as in Naderi & Ruiz (2010).

The performance was measured by the Relative Percentage Deviation (RPD) from the best solution (minimum makespan), which was obtained during the experiment using all combinations of values. Therefore, RPD is calculated as in (3):

$$RPD = \frac{C\,max_k - Best_k}{Best_k} \cdot 100\,, \qquad (3)$$

where $Cmax_{,k}$ is the average makespan obtained in 5 runs on instance k and $Best_k$ is the minimum $Cmax$ obtained in this instance by any heuristic and run.

We carried out a multifactorial ANOVA on the results of these heuristics and all instances. The response variable was the RPD, and the factors are the algorithm, n, m and F being all of them significant (p-$value$=0.00).

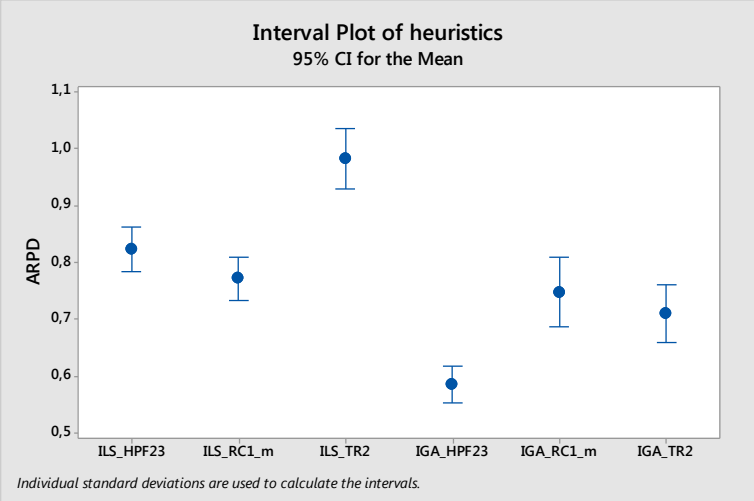

Figure 1. Average RPD of each heuristic by number of factories

To analyze the differences between heuristics, we built the corresponding mean plot with the confidence interval at 95% for the heuristic factor (Figure 1), which is the most significant. From this figure we can observe that the IGA_HPF23 is the one with best performance. The differences between the ILS and IGA that use RC1_m have similar performance. However, these algorithms with TR2 have a totally different behavior. From a more detailed analysis we saw that the number of factories has a great influence in the behavior of these heuristics, especially the IGA. The next step is to compare these algorithms against others proposed in the literature for the DPFSP adapted to the blocking case.

Acknowledgements

Imma Ribas and Ramon Companys are partially supported by the Spanish Ministry of Science and Innovation, under the project RESULT - Realistic Extended Scheduling Using Light Techniques, with reference DPI2012-36243-C02-01.

References

Behnamian, J., & Fatemi Ghomi, S. M. T. (2014). A survey of multi-factory scheduling. Journal of Intelligent Manufacturing, 1–19.

Companys, R. (1966). Métodos heurísticos en la resolución del problema del taller mecánic. Estudios Empresariales, 5(2), 7–18.

Fernandez-Viagas, V., & Framinan, J. M. (2014). A bounded-search iterated greedy algorithm for the distributed permutation flowshop scheduling problem. International Journal of Production Research, 53(4), 1111–1123.

Gao, J., & Chen, R. (2012). A hybrid genetic algorithm for the distributed permutation flowshop scheduling problem.

Gao, J., Chen, R., & Deng, W. (2013). An efficient tabu search algorithm for the distributed permutation flowshop scheduling problem. International Journal of Production Research, 51(3), 641–651.

Gao, J., Chen, R., Deng, W., & Liu, Y. (2012). Solving multi-factory flowshop problems with a novel variable neighbourhood descent algorithm. Journal of Computational Information Systems, 8(5), 2025–2032.

Gao, J., Chen, R., & Liu, Y. (2012). A Knowledge-based Genetic Algorithm for Permutation Flowshop Scheduling Problems with Multiple Factories. International Journal of Advancements in Computing Technology, 4(7), 121–129. doi:10.4156/ijact.vol4.issue7.13

Gong, H., Tang, L., & Duin, C. W. (2010). A two-stage flow shop scheduling problem on a batching machine and a discrete machine with blocking and shared setup times. Disruption Management, 37(5), 960–969.

Johnson, S. M. (1954). Optimal two-and three-stage production schedules with set up times included. Naval Research Logistics Quarterly, 1, 61–68.

Lin, S.-W., Ying, K.-C., & Huang, C.-Y. (2013). Minimising makespan in distributed permutation flowshops using a modified iterated greedy algorithm. International Journal of Production Research, 51(16), 5029–5038.

Liu, H., & Gao, L. (2010). A discrete electromagnetism-like mechanism algorithm for solving distributed permutation flowshop scheduling problem. Proceedings - 2010 International Conference on Manufacturing Automation, ICMA 2010, 156–163.

Martinez, S., Dauzère-Pérès, S., Guéret, C., Mati, Y., & Sauer, N. (2006). Complexity of flowshop scheduling problems with a new blocking constraint. European Journal of Operational Research, 169(3), 855–864.

Naderi, B., & Ruiz, R. (2010). The distributed permutation flowshop scheduling problem. Computers & Operations Research, 37(4), 754–768

Naderi, B., & Ruiz, R. (2014). A scatter search algorithm for the distributed permutation flowshop scheduling problem. European Journal of Operational Research, 239(2), 323–334.

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

Ribas, I., & Companys, R. (2015). Efficient heuristic algorithms for the blocking flow shop scheduling problem with total flow time minimization. Computers & Industrial Engineering, 87, 30–39. doi:10.1016/j.cie.2015.04.013

Sethi, S. P., Sriskandarajah, C., Sorger, G., Blazewicz, J., & Kubiak, W. (1992). Sequencing of parts and robot moves in a robotic cell. International Journal of Flexible Manufacturing Systems, 4, 331–358.

Taillard, E. (1993). Benchmarks for basic scheduling problems. European Journal of Operational Research, 64(2), 278–285.

Wang, S., Wang, L., Liu, M., & Xu, Y. (2013). An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem. International Journal of Production Economics, 145(1), 387–396.

Xu, Y., Wang, L., Wang, S., & Liu, M. (2013). An effective hybrid immune algorithm for solving the distributed permutation flow-shop scheduling problem. Engineering Optimization, 46(9), 1269–1283. doi:10.1080/0305215X.2013.827673