Expert Systems With Applications 114 (2018) 345–356

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

New efficient constructive heuristics for the hybrid flowshop to minimise makespan: A computational evaluation of heuristics

Victor Fernandez-Viagas*, Jose M. Molina-Pariente, Jose M. Framinan

Industrial Management, School of Engineering, University of Seville, Camino de los Descubrimientos s/n, Seville 41092, Spain

ARTICLE INFO

Article history:
Received 5 May 2018
Revised 26 July 2018
Accepted 27 July 2018
Available online 29 July 2018

Keywords:
Scheduling
Hybrid flowshop
Heuristics
Makespan
Computational evaluation
HFS
Memory-based constructive heuristics

ABSTRACT

This paper addresses the hybrid flow shop scheduling problem to minimise makespan, a well-known scheduling problem for which many constructive heuristics have been proposed in the literature. Nevertheless, the state of the art is not clear due to partial or non homogeneous comparisons. In this paper, we review these heuristics and perform a comprehensive computational evaluation to determine which are the most efficient ones. A total of 20 heuristics are implemented and compared in this study. In addition, we propose four new heuristics for the problem. Firstly, two memory-based constructive heuristics are proposed, where a sequence is constructed by inserting jobs one by one in a partial sequence. The most promising insertions tested are kept in a list. However, in contrast to the Tabu search, these insertions are repeated in future iterations instead of forbidding them. Secondly, we propose two constructive heuristics based on Johnson's algorithm for the permutation flowshop scheduling problem. The computational results carried out on an extensive testbed show that the new proposals outperform the existing heuristics.

1. Introduction

The flowshop scheduling problem is one of the most active research areas within Operations Research (see e.g. Fernandez-Viagas, Ruiz, & Framinan, 2017; Framinan, Gupta, & Leisten, 2004; Ruiz & Maroto, 2005 for reviews on the topic). In the flowshop layout, n jobs have to be processed on m stages, each one composed of a single machine, following each job the same route of stages. The problem then consists in obtaining the best sequence of jobs in each machine according to a certain objective (typically the minimisation of makespan, see e.g. Fernandez-Viagas & Framinan, 2014, or the total completion times, see e.g. Fernandez-Viagas & Framinan, 2017a). However, in many manufacturing scenarios several machines in parallel are used to perform an operation as it serves to increase the capacity and/or throughput; to balance the use of the stages; and to decrease the influence of the bottleneck machine (Naderi, Ruiz, & Zandieh, 2010). This flowshop problem with parallel machines in each stage is usually denoted as the Hybrid Flowshop Scheduling (HFS) problem or flexible flowshop scheduling problem. In this paper we address the HFS with the objective of makespan minimisation which is known to aim at minimising production run and maximising machine utilisation. The problem can be denoted as $HFm||C_{max}$ or $FFm||C_{max}$ following Graham, Lawler, Lenstra, and Rinnooy Kan (1979) and, alternatively, by $FHm, ((PM^k)_{k=1}^m)||C_{max}$ following Ruiz and Vázquez-Rodríguez (2010).

Since the problem under consideration is known to be NP-hard by Gupta, 1988 (for the problem even when there are two stages: one with two machines and the other one with a single machine), and by Rinnooy Kan, 1976 (for the problem with a single stage with more than two machines), many approximated algorithms have been developed in the literature (see in this regard the reviews by Ribas, Leisten, & Framinan, 2010; Ruiz & Vázquez-Rodríguez, 2010 and e.g. Chung, Sun, & Liao, 2017; Dios, Fernandez-Viagas, & Framinan, 2018; Ying & Lin, 2018; Zhong & Shi, 2018). In these reviews, most contributions focus on the HFS with identical parallel machines and the maximum completion time or makespan (denoted as C_{max}) as objective, which is also the problem under consideration here. Despite the different heuristics proposed for the problem (see Section 2), we are not aware of any computational evaluation comparing all of them under the same conditions, and only partial comparisons have been performed in the existing literature, using a small subset of heuristics and/or different sets of instances for each comparison.

Among the heuristics proposed for the problem, the NEH (originally proposed by Nawaz, Enscore, & Ham, 1983 for flowshop scheduling and adapted for our problem by Brah & Loo, 1999) seems to be, up to now, one of the best heuristics for the prob-

* Corresponding author.
E-mail addresses: vfernandezviagas@us.es (V. Fernandez-Viagas), jmolina1@us.es (J.M. Molina-Pariente), framinan@us.es (J.M. Framinan).

https://doi.org/10.1016/j.eswa.2018.07.055
0957-4174/

lem, due to its extensive use as initial solution of metaheuristics or as a reference procedure for other constructive heuristics. Despite the excellent performance of the NEH for a range of scheduling problems, recent research has shown different strategies to enhance it: On the one hand, the use of the original objective function of the problem to select the best partial sequence in a heuristic must not necessary imply the best decision in an iteration of the algorithm (see e.g. Dong, Huang, & Chen, 2008; Fernandez-Viagas & Framinan, 2015b, where tie-breaking mechanisms based on idle times are included in the evaluation of partial sequences to improve the solutions in related problems); On the other hand, Fernandez-Viagas and Framinan (2017b) found that, under certain conditions, some stages could be ignored in the traditional flowshop, being approximately equivalent to a single-machine scheduling problem. Note that both reasonings could be also applied to the problem under consideration.

To tackle these challenges, our contribution to the problem is twofold: Firstly, an exhaustive computational evaluation of the heuristics available for the problem is performed. Secondly, we propose four new efficient (memory-based and Johnson-based) constructive heuristics that take into account the aforementioned ideas and that our experiments show that they outperform the existing ones. The first two heuristics construct a solution step by step in a greedy manner, but also taking into consideration the most promising partial solutions obtained in the previous iteration. The last two heuristics reduce the problem to different two-machine flowshop scheduling problems and use the Johnson's algorithm (Johnson, 1954) to solve them exactly. The remainder of the paper is organised as follows: In Section 2, the problem under consideration is formally described and its background is discussed. The constructive heuristics proposed are described in Section 3. The computational evaluation of both existing and new heuristics is presented in Section 4. Finally, the conclusions are discussed in Section 5.

2. Problem description and background

The problem under study can be defined as follows. There is a set $\mathcal{N}$ of n jobs that have to be processed on a set $\mathcal{M}$ of m stages. Each stage i ($\forall i \in \{1, \ldots, m\}$) is composed of m_i identical machines. Each job has to be processed on only one machine in each stage, all jobs following the same order of stages. The processing time of job j in stage i is denoted by p_{ij}. The problem then consists in determining, for each stage, both the machines where each job is to be processed and the order of jobs to process for each machine in order to minimise the maximum completion time or makespan ($C_{\max}$). In addition, the following hypotheses are also adopted: each machine processes at most one job at the same time, and each job is available at initial time; setup times are considered as sequence-independent and non-anticipatory, and they can hence be included in the processing times of the jobs; finally, unlimited inventory is considered between stages.

Note that many approximated algorithms have been proposed to solve the problem in the existing literature, as already mentioned in Section 1 (see Ribas et al., 2010; Ruiz & Vázquez-Rodríguez, 2010 for a more detailed review and explanation of all these approaches). Approximate algorithms can be classified in heuristics and metaheuristics (Framinan, Leisten, & Ruiz-Usano, 2005 and Ruiz & Maroto, 2005). While heuristics (constructive and improvement) typically obtain a fast solution using a fixed number of iterations, metaheuristics are typically forced to stop after a fixed CPU time or number of iterations. In this section, we focus in studies proposing constructive or improvement heuristics for the $HFm||C_{\max}$ problem. In addition, metaheuristics typically require initial solutions obtained using constructive/improvement heuristics. Therefore, we also review the existing metaheuristics for the problem in order to identify additional constructive/improvement heuristics.

Lee and Vairaktarakis (1994) solve the two-stage HFS problem to minimise makespan by using a simple heuristic that assigns jobs to the first stage with the First Available Machine rule (FAM), i.e. each job in a sequence is assigned to the first machine which becomes available. In the second stage, a mirror image of the FAM rule, named Last Busy Machine rule (LBM), is developed to assign the jobs. Koulamas and Kyparisis (2000) propose three linear time heuristics to solve the two- and three-stage case. More specifically, the H_L heuristic solves the two-stage case and two heuristics (denoted as H_0 and H_S) solve the three-stage case. Several heuristics are also proposed by Soewandi and Elmaghraby (2001) to solve the three-stage problem.

For the m-stage case, Santos, Hunsucker, and Deal (1996) adapt four heuristics by Campbell, Dudek, and M.L. (1970), Palmer (1965),Gupta (1971), and Dannenbring (1977), originally developed for the permutation flowshop to minimise makespan. In their experiments, the proposals by Campbell et al. (1970) (denoted as CDS1) and Dannenbring (1977) (denoted as DNN) outperform the heuristics by Palmer (1965), and Gupta (1971). Brah and Loo (1999) compare the CDS1 heuristic against four other heuristics originally proposed for the permutation flowshop problem by Nawaz et al. (1983), Hundal and Rajgopal (1988), Park, Pegden, and Enscore (1984), and Ho (1995). All heuristics were adapted to the hybrid flowshop to minimise makespan and other objectives. The most promising heuristic regarding the makespan are the heuristic by Nawaz et al., 1983 (denoted as NEH), the CDS1 heuristic, and the adaptation of CDS2 (originally proposed by Park et al., 1984). Acero-Dominguez and Paternina-Arboleda (2004) and Paternina-Arboleda, Montoya-Torres, Acero-Dominguez, and Herrera-Hernandez (2008) propose a heuristic, denoted as BH, based in the bottleneck concept according to the theory of constraints (Goldratt & Cox (1992)). They compare their proposal against the traditional shifting bottleneck heuristic (proposed by Adams, Balas, & Zawack, 1988 to solve a job shop layout) and against the hybrid shifting bottleneck-local search (proposed by Pinedo & Chao, 1999).

Regarding metaheuristics, Alaykýran, Engin, and Döyen (2007) and Engin and Döyen (2004) propose an Artificial Immune System and an Ant Colony Optimisation, respectively. With respect to the generation of initial solutions for these algorithms, the former uses a random population based on the idle times between the jobs, while in the latter the method to obtain the initial population is not described. Their algorithms outperform a B&B by Néron, Baptiste, and Gupta (2001) (using a maximum CPU time) on the set of instances proposed by Carlier and Néron (2000) (note that, although this benchmark only considers a maximum number of jobs equal to 15, it is the most used benchmark so far for the problem under consideration). Negenman (2001) adapts several local search algorithms from the flow shop and job shop literature, and compares them against a variable-depth search and a Simulated Annealing. No indication of the initial solution is detailed. Niu, Zhou, and Ma (2009) propose a quantum algorithm using an initial population randomly generated (see e.g. Kurz & Askin, 2004; Norman & Bean, 1999). Liao, Tjandradjaja, and Chung (2012) propose a hybrid Particle Swarm Optimization (PSO) using the BH heuristic to obtain the initial solution. Their results have been compared using the benchmark from Carlier and Néron (2000) against some existing metaheuristics (Alaykýran et al., 2007; Carlier & Néron, 2000; Engin & Döyen, 2004; Niu et al., 2009; Néron et al., 2001). A simple iterated greedy algorithm and two different constructive heuristics, denoted as WT1(x) and WT2(x), are proposed by Kizilay, Tasgetiren, Pan, and Wang (2015). Regarding WT1(x) and WT2(x), they found that both outperform NEH. However, the comparison is carried