Computers & Operations Research 53 (2015) 68–80

Contents lists available at ScienceDirect

Computers & Operations Research

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

A new set of high-performing heuristics to minimise flowtime in permutation flowshops

Victor Fernandez-Viagas, Jose M. Framinan *

Industrial Management, School of Engineering, University of Seville, Ave. Descubrimientos s/n, E41092 Seville, Spain

ARTICLE INFO

Available online 15 August 2014

Keywords:
Scheduling
Flowshop
Heuristics
Flowtime

ABSTRACT

This paper addresses the problem of scheduling jobs in a permutation flowshop with the objective of total completion time minimisation. Since this problem is known to be NP-hard, most research has focussed on obtaining procedures – heuristics – able to provide good, but not necessarily optimal, solutions with a reasonable computational effort. Therefore, a full set of heuristics efficiently balancing both aspects (quality of solutions and computational effort) has been developed. 12 out of these 14 efficient procedures are composite heuristics based on the *LR* heuristic by Liu and Reeves (2001), which is of complexity n^3m. In our paper, we propose a new heuristic of complexity n^2m for the problem, which turns out to produce better results than *LR*. Furthermore, by replacing the heuristic *LR* by our proposal in the aforementioned composite heuristics, we obtain a new set of 17 efficient heuristics for the problem, with 15 of them incorporating our proposal. Additionally, we also discuss some issues related to the evaluation of efficient heuristics for the problem and propose an alternative indicator.

1. Introduction

A flowshop is a common layout in many manufacturing scenarios (see e.g. [17,5,13]) where n jobs must be processed on m machines in the same order. The so-called flowshop scheduling problem consists in finding a sequence of jobs for each machine so certain performance measure(s) is(are) minimised. Additionally, it is customary to assume that the job sequences will be the same on every machine (permutation flowshops), along with other hypotheses such as the simultaneous availability of all jobs and of all machines, and deterministic processing times (for a complete list of these assumptions, see e.g. [1]).

Among the objectives considered in the flowshop scheduling problem, the minimisation of the sum of the completion times of the jobs (or equivalently mean completion time) has been consistently pointed out both as relevant and meaningful for today's dynamic production environment [8]. Under the assumption of a zero release time for the jobs, the minimisation of total (average) completion time is equivalent to total (average) flowtime minimisation, which leads to stable or even use of resources, a rapid turn-around of jobs and the minimisation of in-process inventory [12]. The flowshop scheduling problem with flowtime objective (denoted as $F|prmu|\sum C_j$, according to the notation by Graham et al. [4]) is known to be NP-hard, therefore most of the research on this topic is devoted to developing approaches yielding good (but not necessarily optimal) solutions in reasonable computation time. Obviously, in such approximate methods – or heuristics – one may expect a trade-off between quality of solution and computation time so better solutions are obtained by heuristics requiring longer CPU times. Recently, Pan and Ruiz [10] present an exhaustive evaluation of the different heuristics proposed for the problem in the literature taking into account the quality of the solutions (measured as the average relative percentage deviation over the best known solution) and the CPU time (in seconds). Using these two indicators as in a bicriteria decision problem, they derive a set of non-dominated (i.e. approximation of a Pareto set) heuristics. This 14-heuristics set can thus be used as a benchmark to propose new efficient heuristics for the problem.

A detailed analysis of this Pareto set reveals that 12 out of the 14 heuristics employ a mechanism for constructing the solutions based on the heuristic by Liu and Reeves [8]. In this paper, we propose a new heuristic that improves the results with respect to that by Liu and Reeves both in terms of quality of the solutions and in CPU time. By embedding this new heuristic in several heuristics in the Pareto set, we obtain a completely new efficient Pareto set. Additionally, since the indicators used in the Pareto set by Pan and Ruiz [10] penalise a certain type of heuristics, and we propose an alternative way to measure their efficiency.

The rest of the paper is organised as follows: in Section 2, the formal problem statement and the state-of-the-art heuristics are given. Section 3 analyses some issues related with the performance evaluation of the different heuristics for the problem, and propose

* Corresponding author. Tel.: +34 954487214; fax: +34 954487329.
E-mail addresses: vfernandezviagas@us.es (V. Fernandez-Viagas), framinan@us.es (J.M. Framinan).

http://dx.doi.org/10.1016/j.cor.2014.08.004
0305-0548/

some alternative indicators. In Section 4, a new set of heuristic is presented for the problem. The computational evaluations are carried out in Section 5. Finally, conclusions are discussed in Section 6.

2. Problem statement and state-of-the-art

The problem under consideration can be stated as follows: n jobs have to be scheduled in a flowshop consisting of m machines. On each machine i, each job j has a processing time denoted as p_{ij}. The completion time of job j on machine i is denoted as C_{ij}, whereas $C_{i[j]}$ indicates the completion time on machine i of job scheduled in position j. C_{mj} represents the completion time of job j.

As mentioned in the previous section, a great number of heuristics have been proposed for the problem. For a detailed presentation and evaluation of all these heuristics, we refer the interested reader to Pan and Ruiz [10], and we will describe here only a sub-set which is found to be state-of-the-art and consequently is the one used in this paper for comparison.

According to Framinan et al. [2], heuristics can include one or several of the following phases: index development, solution construction and solution improvement. A heuristic is deemed *composite* if it employs another heuristic for one or more of the three above-mentioned phases. Otherwise, it is regarded as *simple*.

The heuristics that we will considered in our paper are the following:

- Heuristic $LR(x)$ [8]: This heuristic constructs a solution for the problem by appending, one by one, the unscheduled jobs (jobs in set U in the following) at the end of a sequence S of already scheduled jobs. To do so, ξ_{jk} an indicator of the suitability for job j ($j \in U$) to be scheduled in the last position (position $k+1$ where k indicates the amount of scheduled jobs in each iteration) is calculated according to

  $$\xi_{jk} = (n-k-2)\cdot IT_{jk} + AT_{jk}$$

  where IT_{jk} estimates the weighted idle time induced when scheduling job j in position $k+1$, i.e.:

  $$IT_{jk} = \sum_{i=2}^{m} \frac{m \cdot \max\{C_{i-1,j} - C_{i,[k]}, 0\}}{i + k \cdot (m-i)/(n-2)}$$

  and AT_{jk} is the so-called artificial flowtime and it is defined as the sum of the completion time of job j plus the completion time of job p, an artificial job with processing times equal to the average processing time of the other jobs in U (excluding job j), and can be computed as follows:

  $$AT_{jk} = C_{mj} + C_{mp}$$

  More specifically, the $LR(x)$ heuristic operates as follows:
  1. Sort all jobs in ascending order of indicator ξ_{j0}. (Let us U denote such ordered set.) Ties are broken in favor of jobs with higher $IT_{j\,0}$.
  2. Use each of the first x ranked jobs in U as the first job in S, and then constructs a solution by appending the rest of the jobs one by one using indicator ξ_{jk}.
  3. Out of the x solutions so obtained, select the one with the minimum flowtime.
- *Heuristic* $LR(x)$–$FPE(y)$ [8]: This is a composite heuristic where a local search method (denoted $FPE(y)$) is applied to the solution of $LR(x)$. $FPE(y)$ consists of the following steps: for each job j in a sequence, this job is exchanged with the next y jobs in the sequence, and the flowtimes of the so-obtained solutions are evaluated. If any of the solutions has improved the flowtime, then the local search procedure is repeated. Otherwise, the local search stops.
- *Heuristic NEH* [9]: Originally conceived for minimising the makespan in a permutation flowshop, this well-known algorithm has been used as a reference method for many problems in the literature. Its application to the flowtime minimisation problem was discussed by Framinan et al. [3], and it was found that the best option is to first sort the jobs in ascending sum of their processing times. Then, a job sequence is constructed in the following manner: assuming a sequence already determined for the first $k-1$ jobs, k candidate (sub)sequences are obtained by inserting job k in the k possible slots of the current sequence. Out of these k (sub)sequences, the one yielding the minimum flowtime is kept as relative (sub)sequence for these first k jobs given by phase one. Then, job $k+1$ from the first phase is considered analogously, and so on until all n jobs have been sequenced.
- *Heuristic Raj* [11]: This heuristic can be seen as a version of the *NEH*, but here job k is inserted only in slots $\lfloor k/2 \rfloor$ to k, thus reducing the computation time. Additionally, jobs are initially

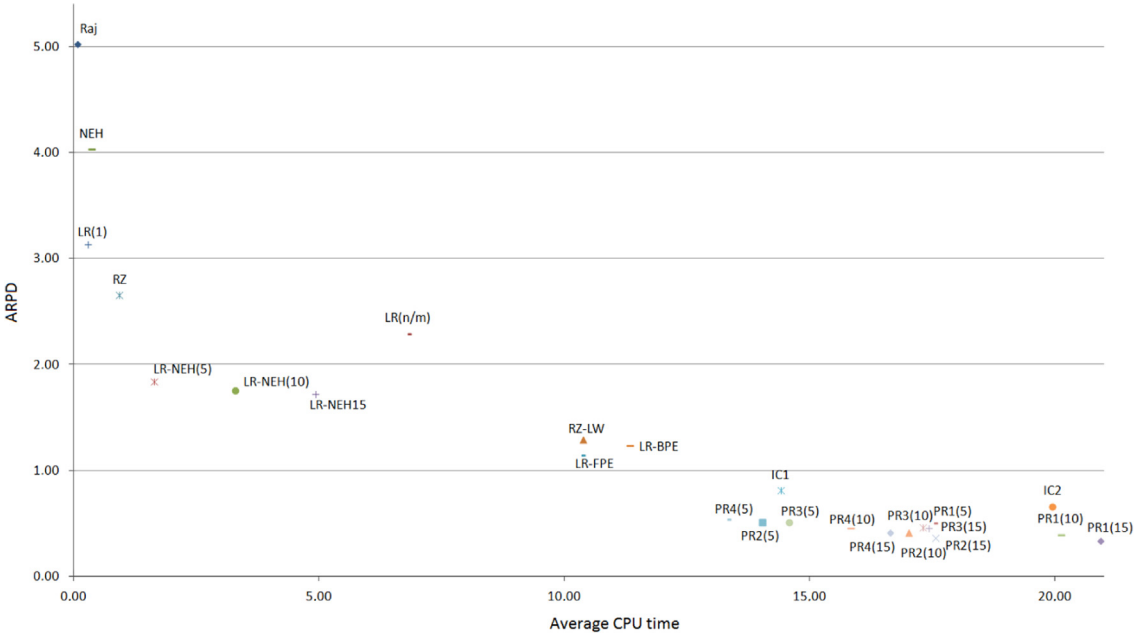

Fig. 1. Pareto set using the average computational time [10].