Computers & Operations Research 78 (2017) 181–192

Contents lists available at ScienceDirect

Computers & Operations Research

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

New approximate algorithms for the customer order scheduling problem with total completion time objective

Jose M. Framinan*, Paz Perez-Gonzalez

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

ARTICLE INFO

Article history:
Received 24 July 2015
Received in revised form
21 August 2016
Accepted 15 September 2016
Available online 16 September 2016

Keywords:
Order scheduling
Completion times
Heuristics

ABSTRACT

In this paper, we study a customer order scheduling problem where a number of orders, composed of several product types, have to be scheduled on a set of parallel machines, each one capable to process a single product type. The objective is to minimise the sum of the completion times of the orders, which is related to the lead time perceived by the customer, and also to the minimisation of the work-in-process. This problem has been previously studied in the literature, and it is known to be NP-hard even for two product types. As a consequence, the interest lies on devising approximate procedures to obtain fast, good performing schedules. Among the different heuristics proposed for the problem, the ECT (Earliest Completion Time) heuristic by Leung et al. [6] has turned to be the most efficient constructive heuristic, yielding excellent results in a wide variety of settings. These authors also propose a tabu search procedure that constitutes the state-of-the-art metaheuristic for the problem. We propose a new constructive heuristic based on a look-ahead mechanism. The computational experience conducted shows that it clearly outperforms ECT, while having both heuristics the same computational complexity. Furthermore, we propose a greedy search algorithm using a specific neighbourhood that outperforms the existing tabu search procedure for different stopping criteria, both in terms of quality of solutions and of required CPU effort.

1. Introduction

In classic scheduling literature, jobs to be processed are treated as individual entities possibly belonging to different customers, and hence the objectives sought are related to the completion times of the individual jobs, or to the differences between the completion times and their due dates or deadlines. However, in many real-life situations, a customer order is composed of different products that have to be processed in the shop, and therefore it may be sensible to pursue objectives related to the completion of the order as a whole rather than to the individual jobs in the order. This is caused by the fact that many customers require to receive the complete order and therefore, from their viewpoint, only the completion time of the full order is relevant [2]. Additional reasons for this assumption include the fact that shipping partial orders results in additional cost of transport, aside than causing a proliferation of documentation and extra management time [3]. Furthermore, if a final assembly of the jobs that compose the order has to be carried out, there is no interest in having just a fraction of these components, which indeed represents an extra-cost for the customer due to the corresponding inventory holding costs. In view of the frequency of real-life settings where this situation arises, a branch of scheduling labelled *customer order scheduling* has emerged as a new research area in order to respond to the ever increasing importance of customer satisfaction and short delivery times [2], and the pre eminence of Make-To-Order production environments where it happens [6].

In this paper, we consider a case of customer order scheduling in which we have a facility with several machines in parallel. Each machine can produce one (and only one) particular product type, i.e. they are considered to be dedicated machines. On this productive environment, customer orders composed of some/all the product types that can be manufactured have to be scheduled. This problem was first formulated by Ahmadi and Bagchi [1], and several practical applications of this model have been described, including finishing operations in the paper industry [6], manufacturing of semi finished lenses [2], the pharmaceutical industry [5], or the assembly of operations [6]. Other specific applications can be seen in Blocher and Chhajed [3], Yang and Posner [15], and Yang [14].

Among the different objectives that can be set for the scheduling problem, perhaps the most treated is the minimisation of the sum of the completion times of the orders, which is related to

* Corresponding author.
E-mail addresses: framinan@us.es (J.M. Framinan), pazperez@us.es (P. Perez-Gonzalez).

http://dx.doi.org/10.1016/j.cor.2016.09.010
0305-0548/© 2016 Elsevier Ltd. All rights reserved.

the delivery times perceived by the customer. Note that minimising the sum (or average) of completion times of the order not only enables a time-based competition, but also aims at lowering the average work in process according to Little's law. The resulting problem is usually denoted in the literature as $PD \,||\, \sum C_j$ (see [6]), and its NP-hardness was first established by Roemer and Ahmadi [12], although their proof contained a flaw (see [6]), so its complexity remained uncertain until the problem was shown to be NP-hard in the strong sense even for two machines by Roemer and Ahmadi [10]. Other objectives also considered in the literature are the total tardiness [4], or weighted sum of completion time [9,13].

Given the NP-hard nature of the problem, it is understandable that the focus of the researchers has been mainly on devising approximate procedures or heuristics to obtain good solutions for the problem in a reasonable CPU time, even if there is no optimality guarantee for these procedures. More specifically, several heuristics for the problem have been proposed by Sung and Yoon [11], Ahmadi et al. [2], Leung et al. [6], and Wang and Cheng [13]. Among then, the so-called ECT (Earliest Completion Time) heuristic by Leung et al. [6], also described in Ahmadi et al. [2], has been shown to clearly outperforms the rest. Indeed, the performance of the ECT heuristic is exceptional according to the results obtained by Leung et al. [6]. These authors use the ECT procedure as a starting solution of a tabu search procedure specifically designed for the problem, which is denoted in the following as TS (ECT). Therefore, both ECT and TS(ECT) constitute the best constructive and improvement heuristics for the problem, respectively. Furthermore, the effectiveness of the ECT heuristic is such that its adaptation (WECT – Weighted ECT) is also among the best heuristics for the related problem of minimising the weighted sum of the completion times (see [7,8]).

The ECT is a constructive heuristic that starts from an empty (partial) sequence and generates a sequence of orders one at a time, each time selecting as the next order to be appended at the end of the partial sequence the one that would be completed the earliest. Such exceptional performance of a pure greedy constructive heuristic suggests a peculiar structure of the solution space, and opens some opportunities for improvement by incorporating look-ahead elements that a greedy behaviour may overlook. Along this idea, in this paper we propose a new constructive heuristic for the problem that exploits some special features of the problem. More specifically, when selecting a new order to be scheduled, our proposed heuristic balances the contribution of the order to the sum of the completion times with the estimated contribution of the non-scheduled orders. By using such trade-off and providing a fast and relatively accurate estimation of the contribution of the non-scheduled orders, the proposed heuristic is shown to outperform ECT by a wide margin, being both heuristics of the same complexity. In addition, we design two specific local search mechanisms for the problem, and embed them into a Greedy Search Algorithm (GSA). The subsequent computational experience carried out shows that GSA outperforms the tabu search algorithm by Leung et al. [6] for different stopping criteria.

The rest of the paper is organised as follows: in Section 2 we formalise the problem under consideration, and discuss its state-of-the-art. Section 3 is devoted to presenting the proposed heuristic (Section 3.1) and the GSA (Section 3.2), while an exhaustive computational experience is carried out in Section 4. Finally, Section 5 is devoted to discuss the main conclusions of the research.

2. Background

The scheduling problem described in Section 1 can be formally stated as follows: there is a facility with m machines in parallel. Each machine can produce one particular product type. There are n customer orders composed of some/all the product types that can be manufactured on the m machines. The total amount of processing required by order i on machine j is p_{ij}, i.e. order i contains a number of units of the product type manufactured in machine j, so it requires p_{ij} time units of this machine. Note that this is equivalent to say that the number of units of a product type requested is different for each customer, which reflects the usual real-life situation.

A schedule or solution $\Pi := (\pi_1, \pi_2, \ldots, \pi_n)$ is given by a permutation of n components as it indicates the sequence in which the orders are processed. Let $C_{\pi_i,j}(\Pi)$ be the completion time of product type j in the order scheduled in the i position in sequence Π. In this setting, it is clear that $C_{\pi_i,j}(\Pi)$ is given by the following recursive equation:

$$C_{\pi_i,j}(\Pi) = C_{\pi_{i-1},j}(\Pi) + p_{\pi_i,j} \quad i = 1, \ldots, n, j = 1, \ldots, m \tag{1}$$

where $C_{\pi_0,j}(\Pi) := 0 \;\; \forall j$. $C_{\pi_i}(\Pi)$ the completion time of order scheduled in position ith is then:

$$C_{\pi_i}(\Pi) = \max_{j=1,\ldots,m} \{C_{\pi_i,j}(\Pi)\} \tag{2}$$

Similarly, $C(\Pi)$ is the sum of the completion times of the orders scheduled according to Π. Obviously, $C(\Pi) = \sum_{i=1}^{n} C_{\pi_i}(\Pi)$.

From the above definitions, it can be seen that a sequence can be evaluated anew in $O(nm)$, being $O(m)$ the computational cost of evaluating $C_{\pi_i}(\Pi)$ if all $C_{\pi_{i-1},j}(\Pi)$ have been already computed.

As mentioned in Section 1, several constructive heuristics have been proposed for the problem:

- *Shortest Total Processing Time (STPT) heuristic* [11]: This heuristic – originally proposed for the weighted total completion time case – constructs a sequence of orders by starting with an empty schedule and selecting as the next order to be scheduled the one with the smallest total amount of processing over all m machines among the non-scheduled jobs. Clearly, the complexity of this heuristic is $O(mn + n\log n)$, as pointed out in Leung et al. [6].
- *Shortest Maximum Processing Time* (SMPT) *heuristic* [11]: This heuristic selects the order with the smallest maximum amount of processing time on any of the m machines. Its complexity is also $O(mn + n\log n)$, and it was initially proposed for the weighted completion time case.
- Smallest Maximum Completion Time (SMCT) heuristic [13]. This heuristic, also applicable for the weighted completion time case, first sequences the orders in non decreasing order of the processing times on each machine j. Consequently, m solutions are obtained, each one denoted as $\Pi^j := (\pi_1^j, \ldots, \pi_n^j), j = 1, \ldots, m$. For each order k, an indicator $I_k := \max_{1 \le j \le m}\{C_{\pi_r^j,j}(\Pi^j) : \pi_r^j = k\}$ is computed. Then, a final solution S is obtained by sorting the orders in non decreasing order of I_k. The complexity of this heuristic is determined by the iteration loop, as the initial sorting order is $O(mn\log n)$ whereas that of the loop is $O(n^2 m)$.
- *Earliest Completion Time* (ECT) *heuristic* [2,6]: This heuristic generates a sequence of orders one at a time; each time it selects as the next order the one that would be completed the earliest. Note that this heuristic can be implemented in a natural way in $O(n^2 m)$ [6], since it has n iterations and, for each iteration, $O(n)$ candidates have to be evaluated according to Eq. (2), which can be computed in $O(m)$ since the completion times of the already scheduled jobs can be stored and the candidates do not need to be computed from scratch.

In order to evaluate the effectiveness of the aforementioned