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

Jose M. Framinan[1]†, Paz Perez-Gonzalez[1]

[1] Industrial Management, School of Engineering, University of Seville,

Ave. Descubrimientos s/n, E41092 Seville, Spain, {framinan,pazperez}@us.es

September 18, 2016

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. (2005b) 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.

Keywords: Order Scheduling, Completion Times, Heuristics

*Preprint submitted to Computers & Operations Research. DOI: http://dx.doi.org/10.1016/j.cor.2016.09.010

†Corresponding author. Tel.: +34-954487214.

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 (Ahmadi et al., 2005). 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 (Blocher and Chhajed, 1996). 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 (Ahmadi et al., 2005), and the pre eminence of Make-To-Order production environments where it happens (Leung et al., 2005b).

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 (1990), and several practical applications of this model have been described, including finishing operations in the paper industry (Leung et al., 2005b), manufacturing of semi finished lenses (Ahmadi et al., 2005), the pharmaceutical industry (Leung et al., 2005a), or the assembly of operations (Leung et al., 2005b). Other specific applications can be seen in Blocher and Chhajed (1996), Yang and Posner (2005), and Yang (2005).

Among the different objectives that can be set for the scheduling problem, perhaps the most

treated is the minimization of the sum of the completion times of the orders, which is related to 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 Leung et al., 2005b), and its NP-hardness was first established by Wagneur and Sriskandarajah (1993), although their proof contained a flaw (see Leung et al., 2005b), 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 (1997). Other objectives also considered in the literature are the total tardiness (Lee, 2013), or weighted sum of completion time (Leung et al., 2007b; Wang and Cheng, 2007).

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 (1998); Ahmadi et al. (2005); Leung et al. (2005b), and Wang and Cheng (2007). Among then, the so-called ECT (Earliest Completion Time) heuristic by Leung et al. (2005b), also described in Ahmadi et al. (2005), 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. (2005b). 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 Leung et al., 2007a; 2008).

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. (2005b) 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-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 $C_{\pi_0,j}(\Pi) := 0 \quad \forall j$. $C_{\pi_i}(\Pi)$ the completion time of order scheduled in position i-th 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 (Sung and Yoon, 1998). 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. (2005b).

- Shortest Maximum Processing Time (SMPT) heuristic (Sung and Yoon, 1998). 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 (Wang and Cheng, 2007). 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 $I_k := \max_{1\leq j\leq m}\{C_{\pi_r^j,j}(\Pi^j) : \pi_r^j = k\}$ 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 $O(n^2m)$.

- Earliest Completion Time (ECT) heuristic (Ahmadi et al., 2005; Leung et al., 2005b). 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 $O(n^2m)$ (Leung et al., 2005b), since it has n iterations and, for each iteration, $O(n)$ candidates have to be evaluated according to equation (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 heuristic, an extensive computational evaluation is carried out by Leung et al. (2005b), where all four heuristics above are compared. In their experimentation, it turns out that ECT is the most efficient heuristic in terms of the quality of the solutions, clearly outperforming the rest of heuristics. Such good performance is remarkable taking into account that it is basically a greedy heuristic. This fact may be indicating a special structure of the solution space, and opens some opportunities for improvement by carefully designing heuristics taking into account such structure. In the next section, we will discuss our proposal, which is based on properly assessing not only the contribution to the completion time done by the candidate order but also the estimated contribution of the unscheduled orders.

3 New approximate algorithms for the problem

In this section, we present new algorithms for the problem under consideration. The first algorithm, discussed in Section 3.1, is a constructive heuristic for the problem based in the ideas presented at the end of the previous section. The aim is to obtain a fast heuristic that can yield better results than the ECT heuristic with the same complexity, thus being able to provide solutions in negligible CPU time.

The second algorithm, discussed in Section 3.2, adopts a greedy search strategy and it is aimed at improving the solutions provided by constructive heuristics using the computation time in an efficient manner, thus being able to outperform the Tabu Search algorithm devised for the problem

by Leung et al. (2005b). To do so, the algorithm uses specific local search mechanisms based on the ECT heuristic. Furthermore, these mechanisms can be eventually used as standalone heuristics.

3.1 A new constructive heuristic

As mentioned earlier, our proposal is based on incorporating a look-ahead mechanism in order to be able to assess, not only the potential contribution of the candidate orders to the objective function, but also an estimation of the contribution to the objective function of the non scheduled orders. More specifically, the proposed heuristic constructs a solution $\Pi := (\pi_1, \ldots, \pi_n)$ from a set $\mathcal{U}$, representing the set of unscheduled orders. Initially, no order has been scheduled and consequently, $\mathcal{U}$ contains all orders. Then the heuristic starts an iterative process of n steps: in step k ($k = 1, \ldots, n$), each order in $\mathcal{U}$ is selected as candidate to be appended at the end of Π. To select the chosen order among the candidates, for each order $\omega_l \in \mathcal{U}$, a (partial) sequence S_l is formed by appending ω_l at the end of Π, i.e. $S_l = (\pi_1, \ldots, \pi_{k-1}, \omega_l)$. Also, $\mathcal{R}_l$ a set of remaining orders is obtained by removing ω_l from $\mathcal{U}$, i.e. $\mathcal{R}_l := \mathcal{U} - \{\omega_l\}$. Then, the following indicator ψ_l is computed as follows:

$$\psi_l = C(S_l) + \frac{1}{n-k+1} C^*(S_l, \mathcal{R}_l) \tag{3}$$

where $C^*(S_l, \mathcal{R}_l)$ is an estimate of the contribution to the completion times of the remaining unscheduled orders if ω_l is to be appended at the end of Π to form S_l. Leaving aside for the moment how an estimation of the unscheduled orders can be computed, note that ψ_l can be seen as a proxy for the objective function if order ω_l is selected to be appended at the end of Π. More specifically, ψ_l is composed of two terms: the first one is the *real* contribution to the objective function caused by the completion times of orders in the partial sequence Π plus that of ω_l when scheduled in position k, and the second term is the *estimated* contribution to the objective function caused by the orders not yet scheduled. The second term is weighted depending on the current iteration of the algorithm, i.e. for the first iterations there are many remaining orders which can be sorted in different ways, therefore the estimation of their contribution to the total completion time is more diffuse and should be smaller than in the last iterations, where there are less remaining orders and the estimation of their contribution would be rather similar to that of the final completion time.

It is clear then that the order with the lowest value of ψ would have the smallest contribution to the total completion times, therefore, in our proposal, the order with the lowest value of ψ is removed from $\mathcal{U}$ and appended at the end of Π. The procedure is repeated until $\mathcal{U} = \varnothing$.

Regarding the computation of $C^*(S_l, \mathcal{R}_l)$, note that the contribution of the orders in $\mathcal{R}_l$ to the completion times will depend on the specific manner in which these orders are scheduled, as different ways to schedule these orders would yield different completion times. One option would be to schedule the jobs in $\mathcal{R}_l$ after S_l in the best possible manner, i.e. with the minimum contribution to the total completion times. However, it is clear that this would imply solving the original problem.

Since the objective is to obtain an estimate of a sort of *average* contribution, we will evaluate such contribution assuming that the orders in $\mathcal{R}_l$ are scheduled following an specific sequence. Note that, in principle, since the orders in set $\mathcal{U}$ depend on the current iteration of the algorithm, such sequence would have to be obtained for each iteration. Since this would increase the computation times, our proposal is to establish such sequence before the algorithm performs the iterations, so it does not have to be calculated for each $\mathcal{R}_l$. More specifically, we will provide Ω a given sequence of all orders before starting the iterative procedure, and, at each iteration, we will compute $C^*(S_l, \mathcal{R}_l)$ as if the orders in $\mathcal{R}_l$ are scheduled in the relative ordering given by Ω. After selecting one of the orders in Ω to be scheduled, the order is appended at the end of the partial sequence and removed from Ω, so in the next iteration the sequence of the remaining orders does not have to be re-computed.

Naturally, one may think that using as Ω a sequence yielding a very low completion time would provide a close estimation of the contribution of these orders and therefore may be tempted to obtain Ω by using e.g. the ECT heuristic. However, it has to be noted that, at each iteration of the proposed heuristic, the selected order would be extracted from Ω, and then there is no guarantee that the remaining unscheduled orders would make a good estimation. Therefore, rather than focusing on providing a sequence yielding a low completion time, we must focus on a robust ordering that ensures a good estimation even if an order is extracted in each iteration. To do so, we turn our attention to the SPT (Shortest Processing Time) rule, which is known to be optimal for job scheduling in a single machine with total completion time objective. This ordering has the advantage that, when certain job is extracted of an SPT sequence, the remaining jobs still comply

Procedure *SPT-B*
  for $j = 1$ **to** m **do**
    Compute the workload of each order in machine j, i.e. $W^j := (p_{1j}, p_{2j}, \ldots, p_{nj})$;
    Obtain Π^j the sequence of orders sorted in ascending (non descending) order of W^j;
  end
  $r := \text{argmin}_j\, C(\Pi^j)$;
  Selects Π^r as the incumbent sequence;
end

Figure 1: Pseudo-code of the SPT-B procedure

the SPT rule, therefore providing a consistent estimation no matter which jobs remain in the partial sequence.

Since SPT cannot be applied to our problem in an straightforward manner as there are orders (not jobs) processed on m machines in parallel, we propose a relatively simple procedure, named SPT-B in the following, inspired in the SPT: we consider each machine j and sort the processing times of each order on this machine according to the SPT rule. By doing so, we obtain m different sequences, each one representing the application of the SPT in each of the machines of the facility. Among these m sequences, we select the one yielding the lowest completion time. The pseudo code of the SPT-B procedure is presented in Figure 1.

Using the above elements, the design and rationale of the heuristic have been discussed: an initial sequence Ω is obtained by sorting the orders according to SPT-B. Then, a solution Π is constructed by iteratively selecting one order in Ω that is removed from Ω and appended as the current last order of Π. At each step, the unscheduled order yielding the lowest value of ψ computed as described in equation (3) is selected to be appended. The detailed pseudo-code of the heuristic is presented in Figure 2. Assuming, as usual, that $n > m$, the complexity of the heuristic is determined by the main iteration loop and not by the SPT-B procedure, as the complexity of the latter is $O(m(n \log n + nm))$ whereas that of the loop is $O(n^2 m)$. As it can be seen, the complexity of the heuristic proposed is the same than that of ECT.

3.2 Greedy Search Algorithm

In view of the good results obtained by the ECT heuristic, specific local search mechanisms based on this heuristic can be defined for the problem. The first of these mechanisms is labelled SHIFT-k. Its

Procedure *Proposed Heuristic*

$\Pi := \varnothing$;

Obtain a sequence $\Omega := (\omega_1, \ldots, \omega_n)$ by sorting orders according to procedure SPT-B;

for $i = 1$ **to** n **do**

for each $\omega_l \in \Omega$ **do**

Form sequence S by appending ω_l after Π and then append the remaining orders in Ω, i.e.: $S := (\pi_1, \ldots, \pi_{i-1}, \omega_l, \omega_1, \ldots, \omega_{l-1}, \omega_{l+1}, \ldots, \omega_{n-i+1})$;

Compute ψ_l:

$$\psi_l = \sum_{k=1}^{i-1} C_{\pi_k}(S) + C_{\omega_l}(S) + \frac{1}{n-i+1} \sum_{k=1;k\neq l}^{n-i+1} C_{\omega_k}(S)$$

end

$r := \text{argmin}_{k=1,\ldots,n-i+1}\, \psi_k$;

Append ω_r at the end of Π, i.e. $\Pi := (\pi_1, \ldots, \pi_{i-1}, \omega_r)$;

Extract ω_r from Ω, i.e. $\Omega := (\omega_1, \ldots, \omega_{r-1}, \omega_{r+1}, \ldots, \omega_{n-i+1})$;

end

end

Figure 2: Pseudo-code of the proposed heuristic

rationale is the following: At iteration k, the ECT heuristic selects the order from the unscheduled orders such as the partial completion time is the lowest one. However, it might be possible to further improve the completion time of the partial sequence by finding the most suitable neighbour for the newly appended order, potentially leading to a better sequence. Using this idea, we design a procedure that starts from an empty sequence Π and a set of unscheduled orders $\mathcal{U}$ and performs $i = 1, \ldots, n$ iterations to append at the end of Π the order $u_r \in \mathcal{U}$ so the partial total completion time of the resulting sequence is the lowest. Then, with the so-obtained $\Pi := (\pi_1, \ldots, \pi_{i-1}, u_r)$, order π_k $(k = 1, 2, \ldots, i-2)$ is removed from its position and inserted it in position $i-1$. This phase of the algorithm is denoted as reinsertion phase. If the so-obtained partial sequence yields a lower total completion time, then it replaces Π as the incumbent partial sequence for the next iteration. The algorithm finishes when all orders have been scheduled and $\mathcal{U} = \emptyset$. The pseudocode of SHIFT-k is presented in Figure 3.

A variant of SHIFT-k –denoted SHIFT-k-OPT in the following– can be designed if, during the reinsertion phase in iteration i, a better sequence is found, then the reinsertion phase is restarted, i.e. all orders in positions $k = 1, 2, \ldots, i-2$ are extracted and inserted in position $i-1$ to see whether further improvements in the objective function are possible. The process is repeated until

Procedure *SHIFT-k heuristic*

$\Pi := \varnothing$;
$\mathcal{U} := \{1, 2, \ldots, n\}$;
for $i = 1$ **to** n **do**
  for each order $u_l \in \mathcal{U}$ **do**
    Form partial sequence S_l by appending u_l after Π, i.e.: $S_l := (\pi_1, \ldots, \pi_{i-1}, u_l)$;
    Compute $C(S_l)$ the total completion time of S_l;
  end
  $r := \text{argmin}_{k=1,\ldots,n-i+1} C(S_k)$;
  Append u_r at the end of Π, i.e. $\Pi := (\pi_1, \ldots, \pi_{i-1}, u_r)$;
  Extract u_r from $\mathcal{U}$, i.e. $\mathcal{U} := \mathcal{U} - \{u_r\}$;
  // *Reinsertion phase*
  for $k = i - 2$ **to** 1 **do**
    Obtain (partial) sequence Ω_k by extracting order π_k from Π and inserting it in position $i - 1$, i.e.: $\Omega_k := (\pi_1, \ldots, \pi_{k-1}, \pi_{k+1}, \ldots, \pi_{i-1}, \pi_k, u_r)$;
    Compute $C(\Omega_k)$;
  end
  $s := \text{argmin}_{k=1,\ldots,i-2} C(\Omega_k)$;
  if $C(\Omega_s) < C(\Pi)$ **then**
    Replace Π as the incumbent solution for the next iteration, i.e.: $\Pi = \Omega_s$
  end
end
end

Figure 3: Pseudo-code of SHIFT-k heuristic

Procedure *SHIFT-k-OPT heuristic*

$\Pi := \varnothing$;
$\mathcal{U} := \{1, 2, \ldots, n\}$;
for $i = 1$ **to** n **do**
  for each order $u_l \in \mathcal{U}$ **do**
    Form partial sequence S_l by appending u_l after Π, i.e.: $S_l := (\pi_1, \ldots, \pi_{i-1}, u_l)$;
    Compute $C(S_l)$ the total completion time of S_l;
  end
  $r := \text{argmin}_{k=1,\ldots,n-i+1} C(S_k)$;
  Append u_r at the end of Π, i.e. $\Pi := (\pi_1, \ldots, \pi_{i-1}, u_r)$;
  Extract u_r from $\mathcal{U}$, i.e. $\mathcal{U} := \mathcal{U} - \{u_r\}$;
  // *Reinsertion phase*
  restart: for $k = i - 2$ **to** 1 **do**
    Obtain (partial) sequence Ω_k by extracting order π_k from Π and inserting it in position $i - 1$, i.e.: $\Omega_k := (\pi_1, \ldots, \pi_{k-1}, \pi_{k+1}, \ldots, \pi_{i-1}, \pi_k, u_r)$;
    Compute $C(\Omega_k)$;
    if $C(\Omega_k) < C(\Pi)$ **then**
      Replace Π as the incumbent solution, i.e.: $\Pi := \Omega_s$
      go to **restart**;
    end
  end
end
end

Figure 4: Pseudo-code of SHIFT-k-OPT heuristic

no improvement is found. Figure 4 presents the pseudocode of SHIFT-k-OPT.

Finally, note that the reinsertion phase of SHIFT-k-OPT can be employed as a local search mechanism: given a complete sequence $\Pi := (\pi_1, \ldots, \pi_n)$, order π_r ($r = 1, \ldots, n-1$) can be removed and inserted in position n, thus obtaining sequence $\Pi' := (\pi_1, \ldots, \pi_{r-1}, \pi_{r+1}, \ldots, \pi_n, \pi_r)$. Then, the most suitable neighbour for π_r can be found by removing orders in positions k ($k = n - 1, \ldots, 1$) in Π', and if a best total completion time is found, then the so-obtained solution becomes the incumbent solution for reinsertion.

In order to obtain very high-quality solutions for the problem, we combine the excellent results (presented in Section 4.2) obtained by the constructive heuristic proposed in Section 3.1 with the local search mechanism above. The result is a Greedy Search Algorithm (GSA) that, starting from the solution provided by the new constructive heuristic proposed performs a number of iterations where the SHIFT-k-OPT local search is combined by pairwise interchange in a greedy search schema.

The pseudocode of this algorithm is summarised in Figure 5 .

4 Computational experience

In this section, we carry out different computational experiments in order to assess the contribution of the heuristics proposed in Section 3. The testbeds employed for the experiments are discussed in Section 4.1, whereas in Section 4.2 we compare the constructive heuristics proposed in Section 3.1. Finally, in Section 4.3, the GSA proposed in Section 3.2 is compared with different variants of the tabu search procedure by Leung et al. (2005b).

4.1 Testbed Design

To the best of our knowledge, the most complete experimentation setting is that by Leung et al. (2005b), where various values of a factor called *order diversity* are tested. This factor refers to the number of product types that the orders may contain. More specifically, for a given number of orders n and a given number of machines m, each order may contain all product types, each one amounting a different processing time requirements for each machine, or just certain types. Taking this factor into account, two different types of instances have been generated:

- Instances type #1, for which each order requests all product types, namely m.
- Instances type #2, for which each order requests r product types ($r < m$).

Different number of customer orders and machines can be set for each testbed type. Both factors, particularly the number of customer orders, determine whether an instance can be optimally solved in reasonable time by e.g. exhaustive enumeration, or not. Therefore, for each testbed type, we build two different sizes of instances:

- Small instances, composed of instances with $n \in \{5, 10, 12\}$ and $m \in \{2, 5, 10, 20\}$. The goal of this testbed is to test the quality of the solutions obtained by the approximate methods against the optimal solution.

Procedure *Greedy Search Algorithm (GSA)*

An initial solution $\Pi := (\pi_1, \ldots, \pi_n)$ is obtained using heuristic NEW;
Set Π as Π^* the best solution so far, i.e. $\Pi^* := \Pi$;
for $i = 1$ **to** *iterations* **do**
Obtain r a random number between 1 and $n-1$;
Obtain sequence Π' by removing order π_r in Π and inserting it in position n, i.e.:

$$\Pi' := (\pi_1, \ldots, \pi_{r-1}, \pi_{r+1}, \ldots, \pi_n, \pi_r)$$

// Reinsertion phase
restart: for $k = n-2$ **to** 1 **do**
Obtain sequence Ω_k by extracting order π_k from Π' and inserting it in position $i-1$;
Compute $C(\Omega_k)$;
if $C(\Omega_k) < C(\Pi')$ **then**
Replace Π' as the incumbent solution, i.e.: $\Pi' = \Omega_k$;
go to **restart**;
end
end
// Pairwise interchange
for $k = 1$ **to** $n-1$ **do**
for $j = k+1$ **to** n **do**
Obtain Π'' by exchanging positions k and j in Π';
if $C(\Pi'') < C(\Pi')$ **then**
$\Pi := \Pi''$
end
end
end
// Update best-so-far if a lower sum of completion times is found
if $C(\Pi) < C(\Pi^*)$ **then**
$\Pi^* := \Pi$
end
end
end

Figure 5: Pseudo-code of GSA

- Big instances, composed of instances with $n \in \{20, 50, 100, 200\}$ and $m \in \{2, 5, 10, 20\}$. This testbed has the same size as in Leung et al. (2005b), thus providing a exhaustive benchmark for the evaluation of the heuristics.

30 instances of each combination of orders and machines have been generated. For the type #1 instances, we follow the testbed design in Leung et al. (2005b) and, for each product type j in customer order i, a processing time p_{ij} has been generated according to a uniform $[1, 99]$ distribution. For type #2 instances, for each customer order i, r requested product types have been obtained, being r drawn from a uniform $[1, m]$ distribution. Also following Leung et al. (2005b), for each one of these product types, the corresponding processing time has been generated according to a uniform $[1, 99]$ distribution.

In summary, the following testbeds will be used in the computational experience:

- $TEST - 1 - S$, composed of 360 type #1 instances of small size.
- $TEST - 2 - S$, composed of 360 type #2 instances of small size.
- $TEST - 1 - B$, composed of 480 type #1 instances of big size.
- $TEST - 2 - B$, composed of 480 type #2 instances of big size.

4.2 Comparison of constructive heuristics

In this section, we assess the effectiveness of existing constructive heuristics for the problem, including the new constructive heuristics presented in Section 3. More specifically, the following heuristics are compared:

- The ECT heuristic by Leung et al. (2005b) described in Section 2 as the best-so-far heuristic for the problem.
- The STPT heuristic by (Sung and Yoon, 1998) described in Section 2 as the second best heuristic for the problem.
- The heuristic proposed in Section 3.1, denoted as NEW in the following.

- The SPT-B sorting order proposed in Section 3. This heuristic is included to check if the behaviour of NEW is mainly due to that of the SPT-B and therefore, the iterative mechanism in NEW is not worth the computational effort.

- The heuristic proposed in Section 3.1 using the solution given by ECT as sorting order for the jobs in $\mathcal{U}$, denoted as NEW-ECT in the following. This heuristic is included to verify the suitability of the SPT-B procedure as initial ordering for NEW as compared to that of ECT.

- The heuristic proposed in Section 3.1 using the solution given by STPT as sorting order for the jobs in $\mathcal{U}$, denoted as NEW-STPT in the following. This heuristic is included to verify the suitability of the SPT-B procedure as initial ordering for NEW as compared to that of STPT.

- The SHIFT-k heuristic presented in Section 3.2.

- The SHIFT-k-OPT heuristic presented in Section 3.2.

The quality of the solutions provided by these heuristics is measured in terms of the Relative Percentage Deviation (RPD) obtained by each heuristic. The RPD is defined for heuristic h in the instance i as follows:

$$RPD_{ih} = \frac{C^{ih} - C^{i*}}{C^{i*}} \cdot 100 \tag{4}$$

where C^{ih} is the completion time obtained by heuristic h on instance i and C^{i*} is the best-so-far total completion time available for instance i. For instances in TEST-1-S and TEST-2-S, C^{i*} is the optimal solution obtained by exhaustive enumeration whereas for instances in TEST-1-B and TEST-2-B, C^{i*} is the best total completion time obtained in the computational experiments by all heuristics under comparison, including those obtained by the lengthy TS and GSA presented in Section 4.3. Note that, in this section, we do not measure the computation effort of the heuristics under comparison as it is negligible.

Tables 1 and 2 show the Average RPD (ARPD) values obtained for each problem size together with the standard deviation of the RPD for the small and big testbeds respectively. In view of these tables, the following comments can be done:

- As it can be seen in both tables, the heuristic proposed clearly outperforms the ECT heuristic, which was considered to be the best-so-far for the problem.

- The quality of the proposal can be assessed not only in relative terms when compared with that obtained by other approximate procedures, but also with respect to the optimal solutions obtained for the small testbeds, being less than 0.8% on average for all testbeds.

- Both for small and big testbeds, it can be seen that the superiority of the proposed heuristic does not depend on the problem size, or on the type of testbed, obtaining the best results for all cases. Additionally, the heuristic seems to be quite robust: it does not only has the lowest average standard deviation, but also its standard deviation is the lowest among the rest of the heuristics for all problem sizes in the two testbeds. This speaks for an exceptional performance as compared with the existing heuristics for the problem. This can be graphically seen in Figure 6, where 95% confidence intervals of the ARPD values for the two big testbeds are shown. In Figure 6 the worst-performing heuristic –SPT-B– has been excluded to make a more homogeneous scaling.

- The sorting order SPT-B is, as expected, a rather bad heuristic itself, although this sorting order was not designed to be used in isolation. Indeed, SPT-B turns to be crucial for establishing the estimation mechanism for the proposed heuristic, which can be seen more clearly by comparing the results obtained by NEW with those found by NEW-ECT and NEW-STPT. From these results it can be seen that the strategy of focusing on the robust sorting ordering for the estimation is critical for the proposed algorithm. This fact can also be reinforced in an indirect manner by checking that STPT is able to outperform ECT as a starting solution.

- The differences in the heuristics are statistically significant for the big testbeds, as shown in Table 3. The table show that, for both testbeds, the NEW heuristic constitutes a separate group, indicating statistically significant differences with the rest of the heuristics. A second group is formed by the SHIFT-k-OPT procedure, followed by SHIFT-k. All of them perform significantly better than ECT and that its experimental variant.

TEST-1-S

		ECT		STPT		NEW		SPT-B		NEW-ECT		NEW-STPT		SHIFT-k		SHIFT-k-OPT	
m	n	Aver.	St. Dev.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
5	2	0.317	1.133	3.542	3.226	0.111	0.330	4.120	4.796	0.317	1.133	0.333	0.922	0.151	0.535	0.084	0.391
5	5	0.863	1.808	4.028	3.676	0.610	1.324	2.987	2.745	0.863	1.808	0.323	0.832	0.321	0.724	0.216	0.477
5	10	0.832	1.596	3.218	2.112	0.479	0.869	1.589	1.819	0.832	1.596	0.599	1.104	0.728	1.340	0.728	1.340
5	20	1.276	1.895	3.849	2.971	0.375	0.689	1.159	4.711	1.276	1.895	0.683	0.978	0.800	1.561	0.736	1.267
10	2	0.402	0.575	4.600	3.882	0.302	0.478	9.356	7.548	0.402	0.575	0.484	0.665	0.218	0.409	0.172	0.386
10	5	0.804	0.996	7.227	4.113	0.726	0.697	7.510	3.677	0.804	0.996	0.482	0.555	0.490	0.839	0.411	0.773
10	10	1.188	1.216	6.444	2.952	0.773	0.612	6.009	2.128	1.188	1.216	0.898	1.064	0.617	0.645	0.542	0.545
10	20	1.542	1.190	6.526	2.719	1.034	0.801	4.522	2.881	1.542	1.190	1.054	0.859	1.102	0.821	1.036	0.821
12	2	0.697	0.779	4.078	3.078	0.535	0.638	10.092	6.401	0.697	0.779	0.480	0.588	0.300	0.462	0.181	0.332
12	5	1.161	1.241	7.524	3.191	0.851	0.734	10.180	3.827	1.161	1.241	0.784	0.752	0.871	1.037	0.827	0.946
12	10	1.789	1.534	7.098	3.151	1.241	1.219	7.600	2.040	1.789	1.534	1.438	1.213	1.142	0.889	1.035	0.847
12	20	1.866	1.347	7.155	2.766	1.142	1.179	5.494	1.453	1.866	1.347	1.180	1.064	1.184	1.116	1.092	1.096
Avg.		1.061	1.276	5.441	3.153	0.682	0.798	5.885	3.669	1.061	1.276	0.728	0.883	0.660	0.865	0.588	0.769

TEST-2-S

		ECT		STPT		NEW		SPT-B		NEW-ECT		NEW-STPT		SHIFT-k		SHIFT-k-OPT	
m	n	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
5	2	0.213	0.621	3.108	4.421	0.236	0.614	7.549	9.916	0.213	0.621	0.270	0.680	0.087	0.444	0.087	0.444
5	5	0.249	0.701	5.157	5.666	0.379	0.767	5.726	4.613	0.249	0.701	0.460	0.930	0.183	0.617	0.183	0.617
5	10	1.207	3.397	6.865	6.727	0.236	0.836	4.046	4.790	1.207	3.397	0.262	0.544	1.144	3.138	1.144	3.138
5	20	0.405	1.823	5.766	4.389	0.113	0.551	2.133	6.753	0.405	1.823	0.120	0.581	0.083	0.490	0.083	0.490
10	2	1.104	1.542	6.197	5.026	0.621	0.999	22.531	13.148	1.104	1.542	0.766	1.003	0.413	0.928	0.359	0.917
10	5	1.306	1.742	9.859	6.613	1.234	2.172	16.565	8.879	1.306	1.742	0.786	0.958	0.691	1.279	0.691	1.279
10	10	0.944	1.814	11.317	8.265	0.757	1.156	13.848	5.765	0.944	1.814	0.610	0.682	0.534	0.875	0.354	0.537
10	20	1.101	1.687	11.773	7.846	0.973	1.100	8.778	2.607	1.101	1.687	1.203	1.506	0.811	1.395	0.805	1.392
12	2	1.081	1.510	7.231	5.432	0.727	0.850	23.489	14.240	1.081	1.510	0.913	1.484	0.493	0.862	0.257	0.412
12	5	1.143	1.394	10.447	8.736	1.016	1.144	18.374	6.390	1.143	1.394	1.020	1.177	0.591	0.976	0.449	0.753
12	10	1.544	1.721	15.354	8.344	1.131	1.129	15.501	5.351	1.544	1.721	1.301	1.249	1.031	1.288	1.012	1.260
12	20	1.722	1.674	10.771	4.984	1.237	1.530	10.395	3.221	1.722	1.674	0.797	0.783	1.294	1.317	1.200	1.337
Avg.		1.002	1.636	8.654	6.371	0.722	1.071	12.411	7.139	1.002	1.636	0.709	0.965	0.613	1.134	0.552	1.048

Table 1: Average and Standard Deviation of RPD for the constructive heuristics on the small testbeds

TEST-1-B

		ECT		STPT		NEW		SPT-B		NEW-ECT		NEW-STPT		SHIFT-k		SHIFT-k-OPT	
m	n	Aver.	St. Dev.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
2	20	1.193	0.746	4.712	2.723	0.545	0.350	15.060	7.615	1.193	0.746	0.909	0.503	0.654	0.398	0.466	0.367
2	50	1.637	0.684	3.724	2.118	0.429	0.178	18.925	5.832	1.637	0.684	1.420	0.485	0.891	0.395	0.691	0.333
2	100	2.519	0.643	3.353	2.017	0.395	0.127	24.545	4.700	2.519	0.643	2.051	0.374	1.395	0.382	1.087	0.280
2	200	2.946	0.418	2.749	1.084	0.340	0.105	25.873	3.077	2.946	0.418	2.388	0.212	1.680	0.250	1.339	0.171
5	20	1.295	0.823	6.342	2.386	0.959	0.697	11.966	3.530	1.295	0.823	1.068	0.893	0.853	0.586	0.743	0.539
5	50	1.943	0.602	6.907	2.117	1.060	0.436	16.675	2.683	1.943	0.602	1.493	0.397	1.457	0.384	1.379	0.369
5	100	1.761	0.326	6.545	1.906	1.034	0.225	16.718	2.181	1.761	0.326	1.610	0.367	1.350	0.248	1.284	0.242
5	200	1.965	0.284	4.974	1.520	1.111	0.191	17.887	1.777	1.965	0.284	1.829	0.350	1.594	0.234	1.513	0.220
10	20	1.793	0.896	8.511	2.561	1.241	0.743	9.912	1.960	1.793	0.896	1.293	0.797	1.424	0.778	1.368	0.767
10	50	1.925	0.673	8.762	2.332	1.422	0.585	13.076	1.653	1.925	0.673	1.544	0.480	1.642	0.529	1.548	0.485
10	100	1.731	0.433	7.420	1.967	1.292	0.367	13.879	1.393	1.731	0.433	1.526	0.284	1.574	0.406	1.537	0.400
10	200	1.693	0.273	5.550	1.368	1.084	0.190	14.023	1.152	1.693	0.273	1.428	0.229	1.593	0.230	1.576	0.225
20	20	1.866	0.934	7.800	2.769	1.082	0.678	7.985	1.533	1.866	0.934	1.768	1.021	1.577	0.794	1.491	0.758
20	50	1.799	0.664	8.646	2.249	1.382	0.659	10.577	1.210	1.799	0.664	1.550	0.556	1.566	0.462	1.513	0.416
20	100	1.628	0.330	8.141	2.072	1.312	0.329	11.468	0.961	1.628	0.330	1.477	0.438	1.522	0.299	1.481	0.293
20	200	1.598	0.218	6.196	1.390	1.146	0.228	11.388	0.859	1.598	0.218	1.338	0.237	1.545	0.207	1.533	0.204
Avg.		1.831	0.719	6.271	2.789	0.990	0.555	14.997	5.789	1.831	0.719	1.543	0.624	1.395	0.534	1.284	0.533

TEST-2-B

		ECT		STPT		NEW		SPT-B		NEW-ECT		NEW-STPT		SHIFT-k		SHIFT-k-OPT	
m	n	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
2	20	1.595	1.120	7.245	3.705	1.065	0.893	32.654	14.073	1.595	1.120	1.407	0.961	0.570	0.512	0.348	0.407
2	50	2.437	0.943	7.222	3.758	0.757	0.302	41.252	9.695	2.437	0.943	2.187	0.818	1.142	0.488	0.829	0.393
2	100	2.989	0.916	6.151	3.209	0.706	0.295	47.241	12.398	2.989	0.916	2.347	0.615	1.404	0.508	1.051	0.416
2	200	3.859	0.760	3.919	2.286	0.552	0.190	53.304	7.700	3.859	0.760	2.820	0.346	1.795	0.433	1.346	0.283
5	20	2.025	1.359	12.810	5.814	1.620	0.935	29.334	8.900	2.025	1.359	1.409	0.828	1.401	1.224	1.166	0.917
5	50	3.472	1.101	12.846	4.238	1.833	0.649	33.836	6.901	3.472	1.101	2.602	0.683	2.617	0.897	2.373	0.759
5	100	3.862	0.804	11.004	3.360	2.107	0.639	39.776	4.907	3.862	0.804	3.155	0.807	3.010	0.711	2.882	0.692
5	200	4.570	0.863	10.314	3.396	2.437	0.626	40.022	3.623	4.570	0.863	3.942	0.806	3.599	0.577	3.393	0.544
10	20	2.432	1.790	14.899	4.974	1.820	1.153	23.119	6.404	2.432	1.790	1.910	1.311	1.905	1.319	1.788	1.249
10	50	3.008	1.314	16.168	5.470	2.121	1.062	31.109	4.777	3.008	1.314	2.611	0.788	2.536	1.114	2.454	1.058
10	100	3.339	0.964	14.096	3.536	2.277	0.689	34.445	2.634	3.339	0.964	2.778	0.548	2.908	0.834	2.842	0.785
10	200	3.428	0.580	11.914	2.692	2.307	0.370	35.602	2.787	3.428	0.580	3.012	0.550	3.118	0.516	3.029	0.526
20	20	2.695	1.772	13.318	4.291	1.697	1.486	18.737	4.317	2.695	1.772	2.063	1.173	2.106	1.358	1.915	1.262
20	50	2.804	0.920	15.891	4.236	2.278	1.107	26.465	3.641	2.804	0.920	2.534	0.780	2.432	0.895	2.334	0.883
20	100	2.717	0.554	14.654	3.784	2.156	0.670	30.790	3.692	2.717	0.554	2.354	0.611	2.475	0.483	2.419	0.479
20	200	2.935	0.527	12.485	2.943	2.067	0.340	32.691	2.284	2.935	0.527	2.411	0.567	2.728	0.490	2.701	0.508
Avg.		3.010	1.286	11.558	5.277	1.738	0.992	34.399	10.892	3.010	1.286	2.471	0.997	2.234	1.139	2.054	1.137

Table 2: Average and Standard Deviation of RPD for the constructive heuristics on the big testbeds

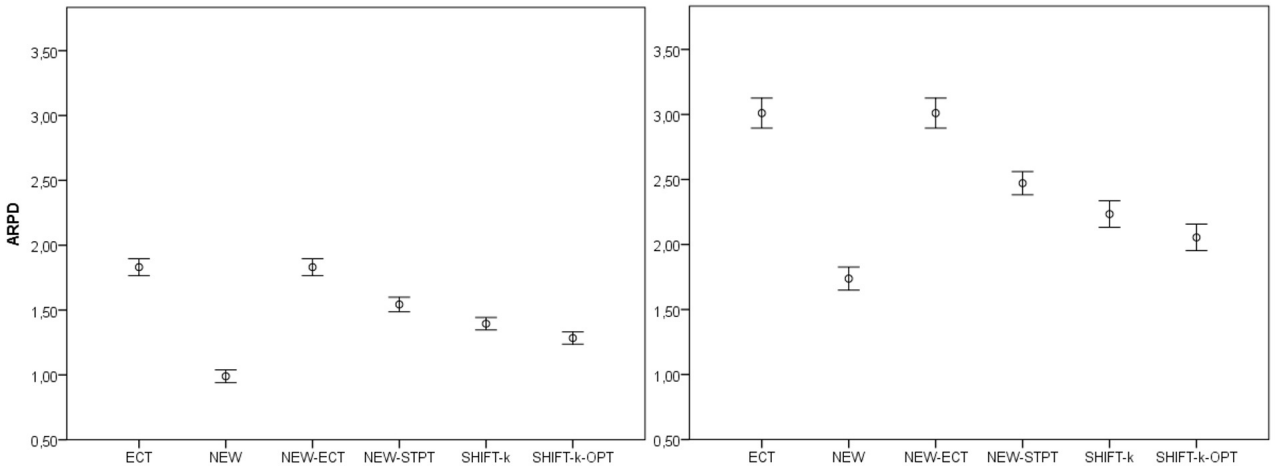
TEST-1-B

		Subset				
Constructive Heuristic	Observations	1	2	3	4	5
NEW	480	0.990				
SHIFT-K-OPT	480		1.284			
SHIFT-k	480			1.395		
NEW-STPT	480				1.543	
ECT	480					1.831
NEW-ECT	480					1.831
Sig.		1.000	1.000	1.000	1.000	1.000

TEST-2-B

		Subset				
Constructive Heuristic	Observations	1	2	3	4	5
NEW	480	1.738				
SHIFT-K-OPT	480		2.054			
SHIFT-k	480			2.234		
NEW-STPT	480				2.471	
ECT	480					3.010
NEW-ECT	480					3.010
Sig.		1.000	1.000	1.000	1.000	1.000

Table 3: HSD Tukey for the testbeds with big instances. Homogeneous subsets of ARPD values for the factor Constructive Heuristic



Constructive Heuristics: TEST-1-B Constructive Heuristics: TEST-2-B

Figure 6: 95% CI RPD for all constructive heuristics (except SPT-B) on the testbeds with big instances

4.3 Comparison of improvement heuristics

In this section, we compare several improvement heuristics for the problem, including the GSA proposed in Section 3.2. More specifically, the following algorithms are compared:

- The tabu search by Leung et al. (2005b) described in Section 2 as the best-so-far metaheuristic for the problem. This algorithm is denoted in the following TS(ECT), as it uses the ECT heuristic as starting solution.
- The TS using NEW as initial solution. This algorithm is included in order to assess the beneficial effect of using NEW as initial solution as compared to using ECT. This algorithm is denoted in the following TS(NEW).
- The GSA presented in Section 3.2.

Different stopping criteria are tested for the algorithms, which are measured in terms of their ARPD and the CPU time effort. More specifically, for the two TS procedures, the number of iterations allowed without improvement is set to 10, 50, and 100. Similarly, the number of iterations allowed in GSA is 10, 50, and 100. Note however that the effort required in the iterations of GSA is different than that of TS, therefore the CPU time has to be computed as well.

The results are shown in terms of RPD in Table 4, and in terms of CPU time in Table 6. According to these results, the following comments can be done:

- Using the proposed heuristic (NEW) as a starting solution of the TS procedure by Leung et al. (2005b) improves the quality of solutions obtained by the tabu search regardless the stopping criteria.
- The results obtained by the proposed GSA are better than those obtained by the TS(NEW) procedure and, consequently, than those obtained by the original TS for the three tested stopping criteria. On average, the best results obtained by the GSA procedure (GSA 100) are around 7 times lower than those obtained by the best TS procedure (TS(NEW) 100). The difference is more striking by taking into account that GSA 100 requires, on average, around 16 times less computational effort in the less favourable case. We think that these

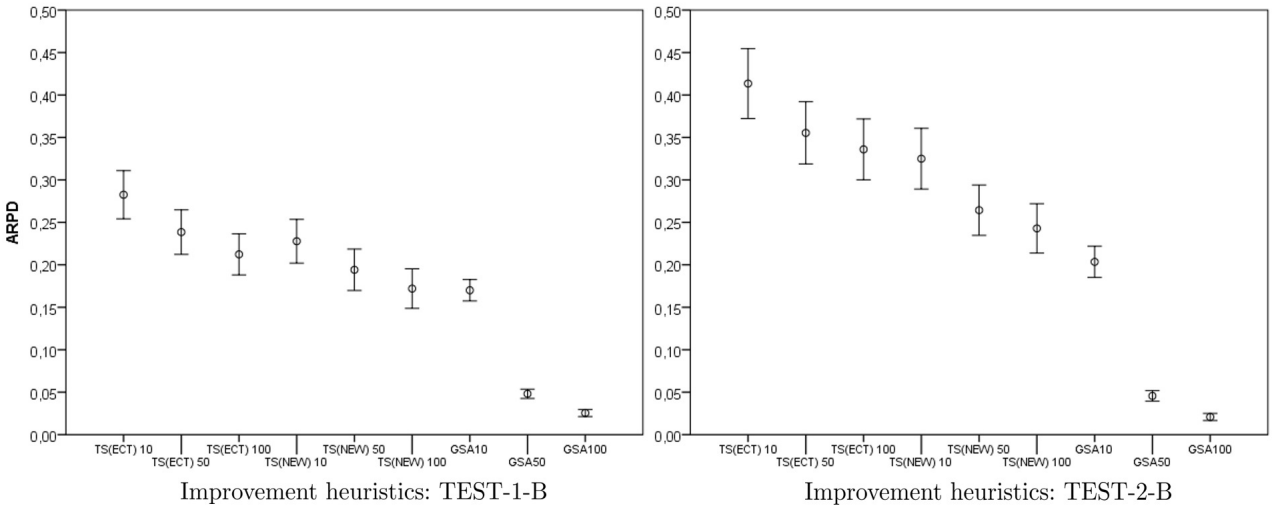


Figure 7: 95% CI RPD for the improvement heuristics on the testbeds with big instances

facts speak for the efficiency of the specific local search mechanism devised for the problem (SHIFT-k-OPT).

- Tukey HSD for the three procedures under comparison are shown in Table 5. From these results, it can be seen that there are statistically significant differences among most procedures (see also Figure 7 where the 95% confidence intervals of ARPD are shown). As it can be seen, GSA stands out as the best procedure, being superior to all versions of TS, with the exception of GSA 10, which is not different than TS(NEW) 100. It has to be taken into account, however, that TS(NEW) 100 is, on average, around 100 times slower than GSA 10.

- For the GSA proposed, the decrease in the ARPD as a function of the number of allowed iterations shows the scalability of the proposed procedure, particularly if compared with that of the TS.

5 Conclusions

In this paper, we address the problem of scheduling orders composed of different product types in a parallel machine environment where each machine is able to manufacture a single product type. The objective is that of minimising the sum of the completion times of the orders. For this

TEST-1-B

		TS(ECT)10		TS(ECT)50		TS(ECT)100		TS(NEW)10		TS(NEW)50		TS(NEW)100		GSA10		GSA50		GSA100	
m	n	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
2	20	0.086	0.216	0.085	0.216	0.060	0.128	0.038	0.093	0.038	0.093	0.038	0.093	0.053	0.068	0.013	0.032	0.004	0.014
2	50	0.059	0.081	0.055	0.082	0.052	0.082	0.008	0.016	0.004	0.006	0.002	0.005	0.070	0.033	0.036	0.025	0.029	0.022
2	100	0.043	0.053	0.040	0.052	0.040	0.052	0.005	0.006	0.003	0.005	0.002	0.005	0.063	0.018	0.042	0.014	0.038	0.014
2	200	0.039	0.042	0.039	0.042	0.039	0.042	0.003	0.006	0.003	0.006	0.002	0.006	0.031	0.015	0.023	0.009	0.021	0.008
5	20	0.169	0.259	0.117	0.179	0.094	0.153	0.141	0.243	0.098	0.230	0.078	0.231	0.132	0.114	0.050	0.066	0.030	0.055
5	50	0.178	0.164	0.119	0.127	0.106	0.123	0.096	0.119	0.070	0.113	0.061	0.109	0.227	0.102	0.113	0.080	0.069	0.055
5	100	0.155	0.114	0.120	0.116	0.108	0.102	0.113	0.098	0.080	0.085	0.061	0.062	0.127	0.072	0.039	0.044	0.031	0.045
5	200	0.097	0.077	0.088	0.078	0.085	0.080	0.059	0.038	0.053	0.041	0.045	0.040	0.104	0.049	0.025	0.025	0.011	0.018
10	20	0.240	0.328	0.174	0.303	0.127	0.265	0.212	0.323	0.101	0.158	0.073	0.107	0.229	0.191	0.074	0.077	0.055	0.072
10	50	0.477	0.259	0.392	0.213	0.355	0.209	0.263	0.215	0.196	0.203	0.145	0.189	0.278	0.142	0.072	0.091	0.040	0.067
10	100	0.374	0.245	0.312	0.246	0.263	0.183	0.295	0.158	0.249	0.156	0.236	0.164	0.132	0.065	0.031	0.037	0.007	0.019
10	200	0.341	0.141	0.307	0.150	0.291	0.148	0.307	0.084	0.280	0.091	0.261	0.089	0.202	0.068	0.026	0.027	0.000	0.000
20	20	0.310	0.306	0.232	0.280	0.191	0.249	0.280	0.308	0.198	0.282	0.113	0.186	0.209	0.166	0.060	0.085	0.041	0.077
20	50	0.427	0.325	0.322	0.307	0.258	0.279	0.490	0.408	0.446	0.404	0.382	0.394	0.302	0.168	0.083	0.084	0.032	0.064
20	100	0.668	0.335	0.628	0.338	0.575	0.343	0.627	0.217	0.592	0.234	0.571	0.249	0.219	0.143	0.039	0.054	0.000	0.000
20	200	0.856	0.225	0.786	0.238	0.754	0.238	0.708	0.176	0.695	0.182	0.684	0.180	0.343	0.101	0.043	0.049	0.000	0.000
Avg.		0.283	0.316	0.238	0.292	0.212	0.271	0.228	0.288	0.194	0.272	0.172	0.260	0.170	0.141	0.048	0.061	0.025	0.047

TEST-2-B

		TS(ECT)10		TS(ECT)50		TS(ECT)100		TS(NEW)10		TS(NEW)50		TS(NEW)100		GSA10		GSA50		GSA100	
m	n	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
2	20	0.032	0.114	0.032	0.114	0.022	0.103	0.030	0.113	0.030	0.113	0.030	0.113	0.063	0.122	0.029	0.065	0.012	0.029
2	50	0.051	0.095	0.049	0.096	0.047	0.096	0.008	0.016	0.008	0.016	0.008	0.016	0.098	0.053	0.044	0.030	0.030	0.023
2	100	0.049	0.062	0.049	0.062	0.048	0.062	0.003	0.006	0.002	0.006	0.002	0.006	0.085	0.030	0.044	0.024	0.039	0.021
2	200	0.035	0.040	0.035	0.040	0.035	0.040	0.001	0.002	0.001	0.002	0.001	0.002	0.040	0.011	0.033	0.011	0.029	0.012
5	20	0.299	0.309	0.234	0.288	0.219	0.271	0.281	0.405	0.214	0.383	0.212	0.385	0.246	0.304	0.041	0.091	0.023	0.081
5	50	0.265	0.259	0.235	0.250	0.214	0.259	0.159	0.135	0.137	0.141	0.126	0.143	0.289	0.115	0.093	0.081	0.051	0.068
5	100	0.246	0.169	0.223	0.168	0.213	0.169	0.194	0.255	0.178	0.252	0.159	0.256	0.140	0.077	0.026	0.041	0.018	0.040
5	200	0.261	0.159	0.254	0.160	0.251	0.163	0.144	0.091	0.134	0.091	0.129	0.090	0.071	0.048	0.010	0.018	0.001	0.004
10	20	0.588	0.713	0.492	0.712	0.473	0.715	0.499	0.722	0.241	0.440	0.168	0.403	0.222	0.260	0.026	0.079	0.016	0.043
10	50	0.625	0.490	0.519	0.492	0.477	0.491	0.434	0.327	0.325	0.244	0.218	0.221	0.423	0.224	0.123	0.111	0.058	0.088
10	100	0.650	0.359	0.591	0.354	0.544	0.369	0.482	0.250	0.440	0.235	0.419	0.248	0.217	0.096	0.042	0.060	0.013	0.038
10	200	0.612	0.209	0.585	0.216	0.574	0.208	0.512	0.145	0.469	0.137	0.461	0.132	0.153	0.095	0.023	0.028	0.000	0.000
20	20	0.542	0.771	0.228	0.284	0.193	0.262	0.326	0.382	0.205	0.290	0.195	0.292	0.265	0.340	0.036	0.073	0.027	0.053
20	50	0.652	0.435	0.530	0.399	0.489	0.388	0.562	0.357	0.393	0.291	0.353	0.302	0.470	0.202	0.079	0.100	0.017	0.050
20	100	0.746	0.371	0.693	0.386	0.645	0.353	0.665	0.423	0.604	0.341	0.569	0.351	0.234	0.149	0.028	0.056	0.000	0.003
20	200	0.962	0.256	0.939	0.264	0.930	0.261	0.901	0.268	0.850	0.239	0.839	0.236	0.241	0.101	0.052	0.057	0.000	0.000
Avg.		0.413	0.460	0.355	0.408	0.336	0.400	0.325	0.398	0.264	0.331	0.243	0.325	0.204	0.205	0.046	0.070	0.021	0.047

Table 4: Average and Standard Deviation of RPD for the improvement heuristics on the big testbeds

TEST-1-B

		Subset				
Improvement Heuristic	Observations	1	2	3	4	5
GSA100	480	0.0255				
GSA50	480	0.0481				
GSA10	480		0.1700			
TS(NEW) 100	480		0.1720			
TS(NEW) 50	480		0.1941	0.1941		
TS(ECT) 100	480			0.2123	0.2123	
TS (NEW) 10	480				0.2278	
TS(ECT) 50	480				0.2385	
TS(ECT) 10	480					0.2825
Sig.		0.4497	0.3598	0.7358	0.2492	1.0000

TEST-2-B

		Subset				
Improvement Heuristic	Observations	1	2	3	4	5
GSA100	480	0.0208				
GSA50	480	0.0456				
GSA10	480		0.2035			
TS(NEW) 100	480		0.2429	0.2429		
TS(NEW) 50	480			0.2643		
TS (NEW) 10	480				0.3249	
TS(ECT) 100	480				0.3360	
TS(ECT) 50	480				0.3554	
TS(ECT) 10	480					0.4135
Sig.		0.8478	0.2771	0.9272	0.6381	1.0000

Table 5: HSD Tukey for big instances testbeds. Homogeneous subsets of ARPD values for the factor Improvement Heuristic

TEST-1-B

n	m	TS(ECT) 10	TS(ECT) 50	TS(ECT) 100	TS(NEW) 10	TS(NEW) 50	TS(NEW) 100	GSA 10	GSA 50	GSA 100
20	2	0.02	0.07	0.10	0.02	0.08	0.16	0.00	0.01	0.02
20	5	0.03	0.13	0.20	0.04	0.13	0.31	0.01	0.02	0.04
20	10	0.07	0.20	0.25	0.06	0.19	0.40	0.01	0.04	0.07
20	20	0.10	0.33	0.48	0.11	0.34	0.60	0.01	0.06	0.12
50	2	0.41	0.81	1.53	0.44	0.82	1.50	0.03	0.12	0.22
50	5	0.84	1.79	3.25	0.83	1.75	3.21	0.06	0.24	0.41
50	10	1.23	2.81	8.22	1.23	2.80	5.28	0.10	0.40	0.66
50	20	2.37	5.04	13.46	2.49	5.06	10.44	0.17	0.63	1.01
100	2	4.47	7.97	27.64	4.90	8.48	13.53	0.18	0.63	1.01
100	5	10.17	20.28	88.12	10.36	20.06	37.47	0.35	1.11	2.06
100	10	15.92	34.76	84.71	15.92	35.04	63.51	0.59	1.92	3.70
100	20	28.63	55.73	121.00	28.97	56.44	106.89	0.97	3.56	6.91
200	2	77.79	100.66	297.40	87.81	112.61	157.36	0.91	3.40	6.48
200	5	190.00	260.46	595.65	190.26	261.07	354.12	2.30	7.94	15.03
200	10	350.09	500.25	970.69	351.50	503.24	727.41	4.46	14.45	28.57
200	20	492.60	862.04	1618.16	506.35	886.34	1366.59	8.15	28.41	56.65
Avg.		73.42	115.83	239.43	75.08	118.40	178.05	1.14	3.93	7.68

TEST-2-B

n	m	TS(ECT) 10	TS(ECT) 50	TS(ECT) 100	TS(NEW) 10	TS(NEW) 50	TS(NEW) 100	GSA 10	GSA 50	GSA 100
20	2	0.01	0.07	0.09	0.01	0.07	0.17	0.00	0.01	0.02
20	5	0.03	0.13	0.15	0.03	0.12	0.26	0.00	0.02	0.04
20	10	0.05	0.17	0.24	0.05	0.18	0.33	0.01	0.04	0.07
20	20	0.08	0.32	0.31	0.09	0.32	0.52	0.01	0.06	0.11
50	2	0.38	0.80	1.18	0.37	0.74	1.59	0.03	0.12	0.22
50	5	0.65	1.47	2.77	0.66	1.46	3.03	0.06	0.24	0.41
50	10	1.02	2.62	5.08	1.01	2.63	4.51	0.09	0.39	0.62
50	20	1.62	4.31	7.91	1.63	4.29	7.52	0.17	0.61	1.00
100	2	3.48	6.14	11.00	3.45	6.36	10.92	0.17	0.61	0.98
100	5	9.18	17.00	27.73	9.13	16.91	27.06	0.35	1.10	1.99
100	10	12.13	25.09	42.22	12.18	25.33	53.25	0.58	1.86	3.56
100	20	22.94	48.38	84.75	23.18	49.05	85.48	0.86	3.46	6.74
200	2	60.37	81.95	113.44	63.80	85.33	113.10	0.83	3.31	6.48
200	5	172.28	234.28	301.62	172.86	234.33	312.29	1.66	7.39	14.64
200	10	251.74	380.54	507.00	255.01	384.91	538.86	2.99	13.90	27.50
200	20	422.21	620.75	2758.04	436.15	638.97	879.57	6.24	28.09	55.82
Avg.		59.89	89.00	241.47	61.23	90.69	127.40	0.88	3.82	7.51

Table 6: Computational effort (seconds) required by the improvement heuristics on the testbeds with big instances

NP-hard problem, the state-of-art constructive heuristic is the so-called ECT (Earliest Completion Time), which selects as the next order to be scheduled the one with lowest completion time among the unscheduled orders. After an in-depth analysis of the problem, we propose a constructive heuristic with the same complexity than the ECT. The key element of this heuristic lies in its ability to estimate the expected contribution of the unscheduled orders, therefore mitigating the greedy behaviour of the ECT. In addition, we propose two improvement mechanisms that are further embedded into a greedy search procedure (GSA), so high-quality solutions can be obtained if a greater CPU effort is allowed.

An extensive computational experience has been carried out to test the performance of the proposed constructive heuristic, which turns to yield a much higher quality of solutions, both for the case where each order should contain all product types, or the case where the orders are just composed of a subset of the product types. The statistical analysis carried out confirms that the proposal outperforms ECT for all problem sizes, being also more robust within each problem size. Furthermore, the comparison of the GSA with an existing tabu search approach for the problem shows the efficiency of our proposal for different stopping criteria.

Acknowledgements

The authors are sincerely grateful to the anonymous referees, who provide very valuable comments on earlier versions of the paper. This research has been funded by the Spanish Ministry of Science and Innovation, under grant "ADDRESS" with reference DPI2013-44461-P.

References

Ahmadi, R. and Bagchi, U. (1990). Scheduling of mult-jobs customer orders in multi-machine enviroments. In *ORSA/TIMS, Philadelphia.*

Ahmadi, R., Bagchi, U., and Roemer, T. A. (2005). Coordinated scheduling of customer orders for quick response. *Naval Research Logistics*, 52(6):493–512.

Blocher, J. D. and Chhajed, D. (1996). The customer order lead-time problem on parallel machines. *Naval Research Logistics*, 43(5):629–654.

Lee, I. S. (2013). Minimizing total tardiness for the order scheduling problem. *International Journal of Production Economics*, 144(1):128–134.

Leung, J.-T., Li, H., and Pinedo, M. (2005a). *Multidisciplinary Scheduling: Theory and Applications*, chapter Order scheduling models: An overview, pages 37–53. Springer.

Leung, J. Y. T., Li, H., and Pinedo, M. (2005b). Order Scheduling in an Environment with Dedicated Resources in Parallel. *Journal of Scheduling*, 8(5):355–386.

Leung, J. Y. T., Li, H., and Pinedo, M. (2007a). Scheduling orders for multiple product types to minimize total weighted completion time. *Discrete Applied Mathematics*, 155(8):945–970.

Leung, J. Y. T., Li, H., and Pinedo, M. (2008). Scheduling orders on either dedicated or flexible machines in parallel to minimize total weighted completion time. *Annals of Operations Research*, 159(1):107–123.

Leung, J. Y.-T., Li, H., Pinedo, M., and Zhang, J. (2007b). Minimizing total weighted completion time when scheduling orders in a flexible environment with uniform machines. *Information Processing Letters*, 103(3):119–129.

Roemer, T. and Ahmadi, R. (1997). The complexity of scheduling customer orders. In *INFORMS Conference 1997, Dallas*.

Sung, C. S. and Yoon, S. H. (1998). Minimizing total weighted completion time at a pre-assembly stage composed of two feeding machines. *International Journal of Production Economics*, 54(3):247–255.

Wagneur, E. and Sriskandarajah, C. (1993). Openshops with jobs overlap. *European Journal of Operational Research*, 71(3):366–378.

Wang, G. and Cheng, T. (2007). Customer order scheduling to minimize total weighted completion time. In *Proceedings of the 1st Multidisciplinary Conference on Scheduling Theory and Applications*, pages 409–416.

Yang, J. (2005). The complexity of customer order scheduling problems on parallel machines. *Computers & Operations Research*, 32(7):1921–1939.

Yang, J. and Posner, M. E. (2005). Scheduling Parallel Machines for the Customer Order Problem. *Journal of Scheduling*, 8(1):49–74.