Expert Systems With Applications 189 (2022) 116097

Contents lists available at ScienceDirect

Expert Systems With Applications

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

A differential evolution algorithm for the customer order scheduling problem with sequence-dependent setup times

Bruno de Athayde Prata [a,*], Carlos Diego Rodrigues [b], Jose Manuel Framinan [c]

[a] Department of Industrial Engineering, Federal University of Ceara, Ceara, Brazil
[b] Department of Statistics and Applied Mathematics, Federal University of Ceara, Ceara, Brazil
[c] Industrial Management/Laboratory of Engineering for Environmental Sustainability, University of Seville, Seville, Spain

ARTICLE INFO

Keywords:
Customer order scheduling
Production sequencing
Assembly scheduling problems
Total completion time
Metaheuristics

ABSTRACT

Although the customer order scheduling problem to minimize the total completion time has received a lot of attention from researchers, the literature has not considered so far the case where there are sequence-dependent setups between jobs belonging to different orders, a case that may occurs in real-life scenarios. For this NP-hard problem we develop a novel efficient approximate solution procedure. More specifically, we develop an innovative discrete differential evolution algorithm where differential mutations are performed directly in the permutation space and that uses a novel, parameter-free, restart procedure. The so-obtained solutions are improved by two proposed local search mechanisms that employ problem-specific, heuristic dominance relations. We carry out an extensive computational experience with randomly generated test instances to compare our proposal with existing algorithms from related problems. In these experiments, the proposed algorithm obtains the best results in terms of their average relative percentage deviation and success rate. Furthermore, an analysis of variance test, followed by a Tukey's test, confirms the excellent performance of the algorithm proposed.

1. Introduction

A lot of attention has been paid to assembly scheduling problems in the last few years (Framinan et al., 2019). Amongst the different assembly scheduling environments, the customer order scheduling arises in several real-world applications, such as the paper industry, the pharmaceutical industry, as well as assembly operations (Leung et al., 2005). In the customer order scheduling environment, a given number of customer orders, each of them composed of different products or services (jobs in the following), has to be processed in a set of dedicated parallel machines. The order is completed once its corresponding jobs have been completed, thus it can be seen as an special case of assembly scheduling with zero assembly processing time.

In most real-life settings, the machines in the shop cannot process different types of jobs without tool changes, adjustments in their settings, etc. More specifically, our work is inspired by the case of a laboratory for quality control in the pharmaceutical industry. The routine of the laboratory is to analyse raw materials, products in process, and finished products. Each order is usually composed of several products that require a specific type of chemical analysis. Depending on the sequence of the products to be analysed, the corresponding equipment requires a different setup, hence the need of explicitly considering the setup times, which in general constitutes an important topic in the scheduling literature (Abreu et al., 2020; Allahverdi et al., 2008; Moccellin et al., 2018). However, studies tackling the customer order scheduling problems including setup times are scarce, being (Prata et al., 2021a) the only contribution addressing the problem, in that case for makespan minimization. Therefore, to the best of our knowledge, the order scheduling problem with sequence-dependent setup times with the objective of minimizing the total completion time has not been previously researched.

Therefore, our paper addresses the customer order scheduling with sequence-dependent setup times to minimize total completion time. The main contributions of this paper are threefold. First, we present a mixed-integer linear formulation to tackle this problem. Second, we develop a discrete differential evolution (DDE) metaheuristic to provide high-quality solutions within feasible computational times. The proposed algorithm applies a new parameter-free restart procedure, a chaotic mechanism to self-adjust the crossover rate, and it incorporates two novel elements i.e. (1) two local search procedures based on dominance relations and (2) a novel restart procedure based solely on the fitness of the solutions. The extensive computational experiments carried out show that the proposed DDE algorithm yields excellent

* Corresponding author.
E-mail addresses: baprata@ufc.br (B.A. Prata), diego@lia.ufc.br (C.D. Rodrigues), framina@us.es (J.M. Framinan).

https://doi.org/10.1016/j.eswa.2021.116097
Received 26 March 2021; Received in revised form 21 August 2021; Accepted 13 October 2021
Available online 24 October 2021

results for the problem and that the novel elements embedded in the DDE substantially improves its performance.

The remainder of the paper is organized as follows: in Section 2, we discuss the problem background. In Section 3, we present the Mixed-Integer Linear Programming (MILP) model for the problem, while in Section 4 we describe the proposed algorithm. In Section 5, the results from the computational experiments are presented and discussed and, finally, in Section 6 we draw some conclusions and suggestions for future works.

2. Literature review

As discussed in Section 1, to the best of our knowledge, the problem under consideration has not been addressed in the literature, even if the customer order scheduling problem is becoming a major topic for production scheduling researchers (Wu, Yang et al., 2019). Therefore, we here discuss related problem in order to investigate whether their solution procedures can be adapted to our problem. More specifically, we review the customer order scheduling problem with total completion time minimization and no setups, and the customer order scheduling problem with set ups and makespan minimization as objective.

Regarding the customer order scheduling problem with total completion time as objective, Wagneur and Sriskandarajah (1993) were the first to discuss this problem as a particular case of the open-shop with job overlapping. They proved that, for the total completion time objective, it is unary NP-hard for $m \geq 2$. Later, Roemer and Ahmadi (2001) showed that this problem is NP-hard in the strong sense. For the objective of minimizing the total weighted completion time, Sung and Yoon (1998) proposed two constructive heuristics, namely the Shortest Total Processing Time first (STPT) and the Shortest Maximum Processing Time first (SMPT). Other constructive heuristics for the problem are due to Wang and Cheng (2007) (Smallest Maximum Completion Time or SMCT heuristic), and to Leung et al. (2005) (the SMCT heuristic, the Earliest Completion Time or ECT dispatching rule, and a Tabu Search metaheuristic). Shi et al. (2017) proposed two mathematical models for the weighted case, being the first one a quadratic formulation, and the second one a linearization of the first model. Furthermore, they proposed a hybrid nested partitions algorithm. Framinan and Perez-Gonzalez (2017) proposed a constructive heuristic for the unweighted case with a look-ahead mechanism that estimates the contribution to the objective function of the non-scheduled orders. Furthermore, a Greedy Search Algorithm (GSA) metaheuristic is proposed. Finally, Riahi et al. (2019) extended the look-ahead mechanism presented in Framinan and Perez-Gonzalez (2017) to all positions in the sequence, called Permutation Construction and Exploration (PCE) algorithm. Besides, they propose a Perturbative Search Algorithm (PSA) metaheuristic. According to their computational experience, their PSA can be considered the state-of-the-art algorithm for this problem and hence its adaptation to our problem will be used in the subsequent experimentation in Section 5. Finally, some special cases of the customer order scheduling problem with stochastic considerations or including a learning effect can be found in Wu et al. (2021) and Wu, Lin et al. (2019), respectively.

Regarding the customer order scheduling problem with setups and makespan as objective, this problem is first introduced by Prata et al. (2021a). Two mixed-integer linear programming models are developed for this problem, along with two matheuristics that reduce the number of decision variables. Their so-called Fixed Variable List Algorithm (FVLA) is shown to outperform all other methods under comparison, being the state-of-the-art solution procedure for the problem. This procedure will be adapted for the total completion time as objective in the experiments in Section 5. Note that the corresponding problem without setups (the customer order scheduling problem with makespan objective) is a trivial decision problem and therefore there are no procedures that can be adapted to our case.

As a summary, despite the relevance of the problem and the existence of solution procedures for related problems, we are not aware of contributions dealing with the customer order scheduling problem with setups and total completion time minimization as objective. This problem is formalized in the next section, together with the proposal of a MILP model.

3. Problem statement and MILP formulation

The problem under consideration can be described as follows: There are n customer orders, each one composed of m jobs, that must be completed in a shop. Each job i ($i = 1, \ldots, m$) in the order j ($j = 1, \ldots, n$) has to be processed on a dedicated machine requiring p_{ij} times unit to be completed. In addition, job k requires a setup time s_{ijk} if processed after job j, which is sequence-dependent (since these are dedicated machines, we can also say that s_{ijk} is the setup time of order k in machine i if processed after order j). An order can be considered completed whenever all the jobs in the order have been processed in the dedicated machines. The objective is to minimize the total completion times of the orders.

It is clear that, for each machine i, a permutation $\Pi_i := (\pi_1^i, \ldots, \pi_n^i)$ represents a feasible sequence in which the customer order π_j^i (or, equivalently, the ith job in the customer order j) is processed in machine i in position j. The decision problem can be then formulated as finding a sequence for each dedicated machine so the sum of the completion times of the orders is minimized. This problem is NP-hard problem since it can be reduced, if all setup times are equal to zero, to the canonical customer order scheduling problem to minimize total completion time, which is known to be an NP-hard problem (Wagneur & Sriskandarajah, 1993).

Given $\Pi_1, \ldots, \Pi_m$ a set of m permutations indicating the precedence of the processing of each order on a given machine, the completion time of each order in the dedicated machines can be computed as follows:

$$C^i_{\pi^i_j} = C^i_{\pi^i_{j-1}} + s_{i,\pi^i_{j-1},\pi^i_j} + p_{i,\pi^i_j} \quad j = 1, \ldots, n \quad i = 1, \ldots, m \tag{1}$$

Then, C_j the completion time of order j is:

$$C_j = \max_{i=1,\ldots,m} \left\{ C^i_j \right\} \tag{2}$$

and the total completion time is simply $\sum_j C_j$.

In order to better understand the problem under consideration and to be able to solve small instances to compare the efficiency of the proposed approximate methods, we develop a Mixed Integer Linear Programming (MILP) model of the problem. This model extends the formulation presented by Framinan and Perez-Gonzalez (2018) for the corresponding problem without setups. Note that the extension is not trivial as in the problem without setups it can be assumed that exists an optimal solution where the sequence is the same in all dedicated machines, which is not true, in general, if there are non-zero setup times.

Let x_{ijk} be a binary decision variable equal to 1 if order k is scheduled in position j of machine i. Taking into account the sequence-dependent setup times s_{ilk}, the model determines the setup time of position j on each machine i (D_{ij}), and consequently the total completion time of all orders. Hereafter, the notation used for the model is presented.

Indices and sets

k: index for orders: $k \in K := \{1, 2, \ldots, n\}$.
j: index for positions: $j \in J := \{1, 2, \ldots, n\}$.
i: index for machines: $i \in I := \{1, 2, \ldots, m\}$..

Parameters

p_{ik}: processing time of the job corresponding to order k in machine i.

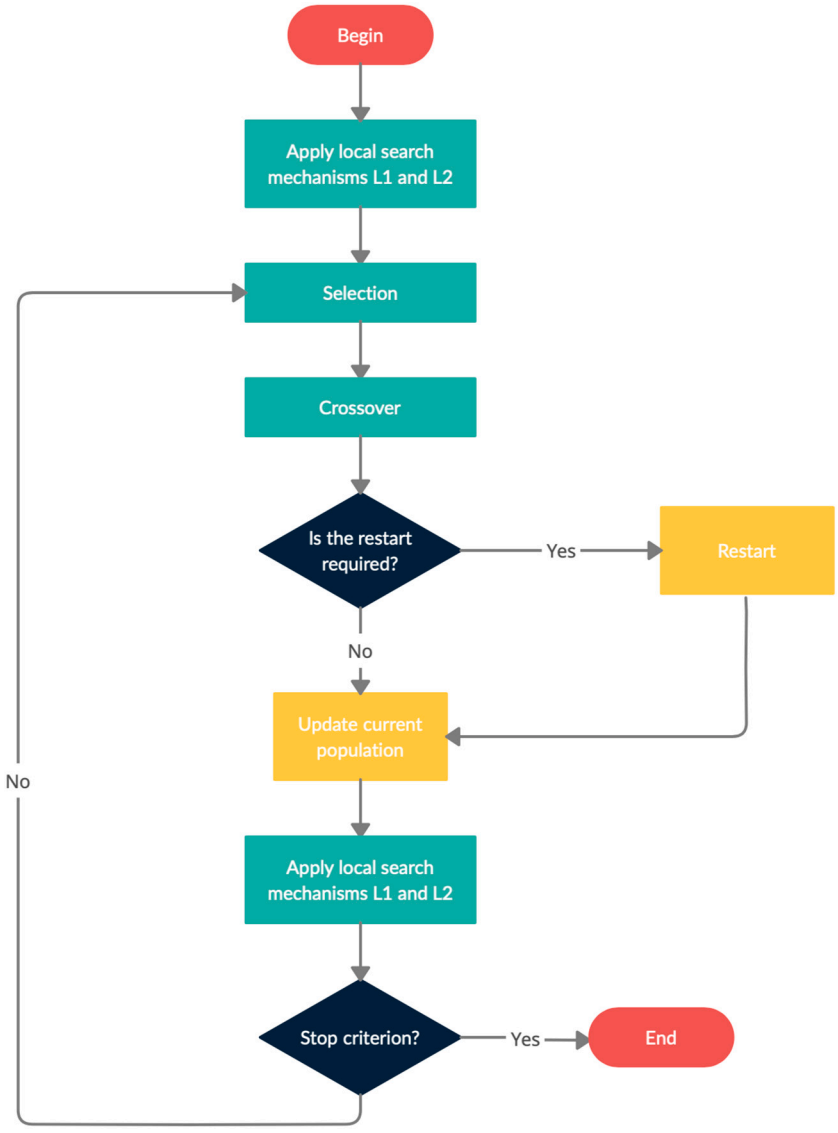

Fig. 1. Flowchart of the proposed Discrete Differential Evolution algorithm.

s_{ijk}: setup time of order k in machine i if processed after order j.

Decision variables

C_j: completion time of the order processed in position j.

C_{ij}: completion time of the (job corresponding to the) order processed in position j of machine i.

D_{ij}: setup time of order processed in position j in machine i.

$$x_{ijk} = \begin{cases} 1, & \text{if order k is processed in machine i in position j.} \\ 0, & \text{otherwise} \end{cases}$$

The resulting MILP model is the following:

minimize

$$\sum_{j=1}^{n} C_j \tag{3}$$

subject to

$$\sum_{k=1}^{n} x_{ijk} = 1, \qquad \forall i \in I, j \in J \tag{4}$$

$$\sum_{j=1}^{n} x_{ijk} = 1, \qquad \forall i \in I, k \in K \tag{5}$$

$$D_{ij} \geq (x_{i,j-1,k} + x_{ijl} - 1)s_{ikl}, \qquad \forall i \in I, \forall j > 1, \forall k, l \in K \tag{6}$$

$$\sum_{r=1}^{j} D_{ir} + \sum_{k=1}^{n} \sum_{r=1}^{j} p_{ik} x_{ikr} \leq C_{ij}, \qquad \forall i \in I, j \in J \tag{7}$$

$$C_j \geq C_{ij}, \qquad \forall i \in I, j \in J \tag{8}$$

$$C_j \geq 0, \qquad \forall j \in J \tag{9}$$

$$C_{ij} \geq 0, \qquad \forall i \in I, j \in J \tag{10}$$

$$D_{ij} \geq 0, \qquad \forall i \in I, j \in J \tag{11}$$

$$x_{ijk} \in \{0, 1\}, \qquad \forall i \in I, j \in J, k \in K \tag{12}$$

The objective function (3) is the total completion time minimization. Constraint set (4) ensures that an order k is scheduled only in a position j of the machine i. Constraint sets (5) enforces that a position j receives only a order k in machine i. Constraint sets (6) calculates the setup time of machine i in position j. We are assuming that all the machines are prepared in the first position i.e the setups times are only computed if j is greater than 1. Constraint sets (7) and (8) calculate the

total completion times. Finally, constraint sets (9), (10), (11), and (12) define the scope of the model variables.

4. A new discrete differential evolution algorithm

Differential Evolution (DE) is a well-known metaheuristic originally proposed by Storn and Price (1997) for continuous optimization problems. In the last few years, several extensions of the standard DE algorithm have been proposed for discrete optimization problems, including manufacturing scheduling problems (Bai et al., 2017; Liu et al., 2020; Pan et al., 2008; Wang et al., 2010; Zhao et al., 2020; Zhou et al., 2021). DE algorithms are usually classified as evolutionary algorithms since they use a population of solutions, even if they are not bio-inspired algorithms, as their behaviour is not based on natural systems such as the evolution of species, or the behaviour of social animals.

In view of the successful application of DE to other scheduling problems, we decided to analyse this metaheuristic to solve the problem under study. The main characteristics of this algorithm that led us to select it are the low dependency of parameters and the facility of hybridization with other solution procedures.

We present here a variant of the DE algorithm (hereinafter referred to as Discrete Differential or DDE), which, according to the classification in Storn and Price (1997) is a DE/rand/1/bin algorithm. The pseudocode outlining the main components of DDE are shown in Algorithm 1, whereas in Fig. 1 we present a flowchart of the algorithm. Firstly, we generate an initial population with a random permutation for each machine, and these sequences are improved using the well-known NEH algorithm (Nawaz et al., 1983). The NEH algorithm is a constructive heuristic based in the iterative insertion of jobs (orders in our case) in a partial solution, and it yields excellent results for a wide range of scheduling problems, such as e.g. the permutation flowshop (Nawaz et al., 1983). Then, the value of the objective function for each element in the current population (hereinafter referred to as *fitness*) is calculated. Next, we improve the entire population using two innovative local search mechanisms – denoted as L1 and L2 –, which employ specific heuristic dominance relations for the problem to reduce the computational effort.

After this initial build up of a population of size pop_{iv} (described in detail in Section 4.1), a number of iterations are carried out until a given time limit is reached. More specifically, in each iteration, three elements from the current population are randomly selected and a crossover operator is applied to them. Furthermore, mutant solutions in the current population are generated if a number randomly generated for each position j is less than or equal to the crossover rate (both crossover and mutation mechanisms are described in Section 4.2). Then, we compute the fitness of each solution in the so-obtained population. If the best and worst fitness are equal, then the entire population is composed of solutions with the same fitness and we perform a restart procedure to introduce diversity of solutions in the population (the restart procedure is described in Section 4.4). Otherwise (if the best and worst fitness are different), we apply the local search mechanisms L1 and L2 to all solutions, updating the entire population. In Section 4.5 we describe these mechanisms in detail.

Since the proposed metaheuristic runs until a specified time limit, it is difficult to determine the overall complexity of the algorithm. Nevertheless, we can emphasize the complexity of some of its parts, including the main loop. The first step presents the complexity $\mathcal{O}(NEH)$, i.e. the complexity of the NEH heuristic for the problem under study. Furthermore, assuming that the local search procedures are applied in all elements of the population, we have the complexity $\mathcal{O}(NPmn^2)$, since L1 presents complexity $\mathcal{O}(NPmn)$, and L2 presents complexity $\mathcal{O}(NPmn^2)$ (that dominates L1). The complexity of the main loop is dominated by the case where the local search procedures are executed. Thus, its complexity is $\mathcal{O}(NPmn^2)$.

	J1	J2	J3	J4	J5
M1	2	1	5	4	3
M2	5	3	1	2	4

Fig. 2. Encoding for the proposed Discrete Differential Evolution algorithm.

Based on the above considerations, the overall complexity of the proposed DDE algorithm is given by: $\mathcal{O}(NEH) + \mathcal{O}(NPmn^2) + \mathcal{O}(Tmn^2)$, where T is the number of iterations until the time limit is reached.

Algorithm 1 Discrete Differential Evolution algorithm

Result: Return a permutation matrix

```
Generate an initial population pop_iv with the NEH algorithm.
Apply local search mechanisms L1 and L2.
while current_time ≤ time_limit do
    Randomly select r1,r2,r3 ∈ {1,2,...,NP}.
    Evaluate the following test for each position j
    if (U)[0,1] ≤ CR then
        ω_v^i ← π_{r1}^i ⊕ (π_{r2}^i ⊖ π_{r3}^i)
    else
        ω_v^i ← π_v^i
    end
    Evaluate each solution according to the fitness function.
    if fitness_MAX = fitness_MIN then
        Perform restart procedure
    else
        Apply local search mechanisms L1 and L2.
        Update current population
    end
end
```

In the following sections we describe the main elements of DDE in detail.

4.1. Problem encoding and generation of the initial population

To represent a feasible solution, we use a permutation list of the orders in each machine. In this manner, all the solutions generated are feasible. Fig. 2 illustrates the problem encoding, for an element of the population with m=2 and n=3. For the fitness evaluation, we calculate the total completion time using the recursive Eqs. (1)–(2).

Since we are not aware of an existing constructive heuristic for the problem, we tried to adapt some algorithms available to related problems. For each individual of the population, we generate a random permutation for each machine. Then we apply the well-know NEH heuristic on each machine to improve the total completion time on this machine, without considering the rest of the machines.

4.2. Selection, crossover and update of the population

Usually, in the discrete differential evolution algorithms, the population of solutions is modelled as a set of float numbers, in which the standard operators are performed. Next, the float numbers are decoded into permutations aiming to calculate the fitness of the current population (Onwubolu & Davendra, 2009). After preliminary computational experiments, we could observe that this approach did not lead to good results. Thus, we decided to perform the standard operators directly in the permutation space, as also suggested in Santucci et al. (2016).

As the selection operator, we adopt the standard selection procedure where three distinct solutions are randomly chosen from the current population (Storn & Price, 1997). The crossover operator combines the

three selected permutations $\pi^i_{r_1}$, $\pi^i_{r_2}$, and $\pi^i_{r_3}$ to create a mutant solution ω^i_v as in Eq. (13):

$$\omega^i_v = \begin{cases} \pi^i_{r_1} \oplus (\pi^i_{r_2} \ominus \pi^i_{r_3}) & \text{if } \mathcal{U}[0,1] \leq CR \\ \pi^i_v & \text{otherwise} \end{cases} \tag{13}$$

Let π_1 and π_2 be two permutations, the analogous operators for sum and difference of permutations can be defined as follows.

$$\pi_1 \oplus \pi_2 = \pi_1 \circ \pi_2 \tag{14}$$

$$\pi_1 \ominus \pi_2 = \pi_2^{-1} \circ \pi_1 \tag{15}$$

Aiming to illustrate the operations with permutations, consider the following permutations $\pi_1 = \{3,2,1,5,4\}$, $\pi_2 = \{3,4,2,1,5\}$, and $\pi_3 = \{4,3,5,2,1\}$. The inverse permutation π_3^{-1} can be obtained as follows:

$$\begin{pmatrix} 4 & 3 & 5 & 2 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 4 & 2 & 1 & 3 \end{pmatrix}$$

In positions 1, 2, 3, 4, and 5 of π_3, we have the values 4, 3, 5, 2, and 1, respectively. If we consider each initial value as a position and each initial position as a value, we have the values 5, 4, 2, 1, and 3 in the positions 1, 2, 3, 4, and 5. Thereby, the inverse permutation is $\pi_3^{-1} = \{5,4,2,1,3\}$. Two permutations can be composed by tracing the destination of each element. Taking into consideration π_3^{-1} and π_2, $\pi_3^{-1} \circ \pi_2$ can be found as follows: $\pi_3^{-1}(\pi_2(1)) = \pi_3^{-1}(3) = 2$, $\pi_3^{-1}(\pi_2(2)) = \pi_3^{-1}(4) = 1$, $\pi_3^{-1}(\pi_2(3)) = \pi_3^{-1}(2) = 4$, $\pi_3^{-1}(\pi_2(4)) = \pi_3^{-1}(5) = 3$, and $\pi_3^{-1}(\pi_2(5)) = \pi_3^{-1}(5) = 3$. Finally, the composition between π_1 and the difference of π_2 and π_3 can be found as follows: $\pi_1(2) = 2$, $\pi_1(1) = 3$, $\pi_1(4) = 5$, $\pi_1(5) = 4$, and $\pi_1(3) = 1$. Hence, $\pi_1 \oplus (\pi_2 \ominus \pi_3) = \{2,3,5,4,1\}$.

A mutant solution Ω_{iv} will replace its basic solution pop_{iv} if the fitness of the mutant solution is less than or equal to the fitness of its basic solution. This selective strategy incurs in an evolutionary pressure, leading to high-quality solutions. However, with this selective strategy, a premature convergence of the population can be observed. To mitigate the population convergence, we used a restart procedure to guarantee diversity in the search process.

4.3. Self-adjustment of CR values

In our proposal, we adopt a dynamic parameter control mechanism for the self-adjustment of CR values, based on a chaotic system (Lu et al., 2011). Tavazoei and Haeri (2007) evaluated 10 one-dimensional maps in chaos optimization algorithms and concluded that the Tent (triangular) function presented better results. The dynamic adjustment of parameter CR is determined as in Eq. (16):

$$CR^{g+1} = \begin{cases} 2CR^g, & 0 < CR^g < 0.5 \\ 2(1 - CR^g), & 0.5 < CR^g < 1 \end{cases} \quad g = 1,2,\ldots,g_{max}. \tag{16}$$

where g is the current generation and g_{max} is the maximum number of generations. In the first generation, the initial value CR^0 is generated using a uniform distribution excluding the values 1/4, 1/2, 2/3, and 3/4, as suggested by Tavazoei and Haeri (2007).

4.4. Restart procedure

Traditionally, the restart procedure is based on a parameter that reflects the number of iterations without improvement in the best solution found (Abreu et al., 2020; Alcaraz et al., 2003; Prata, 2015; Ruiz et al., 2006; Sales et al., 2018). In our view, this approach presents two main disadvantages: the first one is the necessity to calibrate a new parameter, and the second one is the uncertainty of the moment in which the convergence of the current population occurs.

Here we propose a restart procedure based only on the fitness of the solutions. Let $fitness_{MAX}$ be the worst fitness in the current population, and $fitness_{MIN}$ be the best fitness in the current population, we restart the current population if the maximum fitness of the population equals its minimum fitness, i.e. if $fitness_{MAX} = fitness_{MAX}$. Thereby, we eliminate the necessity of a parameter for the restart procedure, and we restart the population in the exact iteration in which the current population lost all diversity. We can emphasize that we randomly select an element from the current population to be maintained after the restart, such as in the elitism practice. In other words, all solutions in the population are replaced by random solutions with the exception of one, which is maintained.

4.5. Local search mechanisms

We face a non-permutation scheduling problem with a large search space. Thus, a local search procedure would present a high computational effort since we must investigate the permutations for all machines and solutions in the current population. Using a well-known 2-opt local search that presents a complexity $O(n^2)$, for m machines and NP solutions during a specified number of generations or a given time limit could be prohibitive to large-sized instances.

With the aim of reducing the number of evaluated solutions, we develop two local search mechanisms that employ concepts of dominance relations heuristically. The first one explores a neighbourhood with adjacent orders and the second one explores a neighbourhood with non-adjacent orders. We can observe that there is no guarantee that the dominance relations are verified. However, this hybrid procedure reduces the computational effort as well as improved substantially the quality of the solutions found.

The local search L1 can be described as follows. Let a and b be two adjacent orders in $\Pi_i := (\pi^i_1, \ldots, z, a, b, \ldots, \pi^i_n)$. If $s_{iza} + p_{ia} + s_{iab} < s_{izb} + p_{ib} + s_{iba}$, we swap the above-mentioned orders so order a precedes order b in Π_i. The rationale of this heuristic dominance condition is that, if two sequences for a given machine i differ only with respect to two adjacent jobs a and b, then it is more likely that we obtain a better solution by placing these jobs so the additional operating time (processing times plus setup times) induced in this machine is lower. Note however that this is just a heuristic rule, as it is not true that this local (i.e. in machine i) reduction of the completion time necessarily translates in a reduction of the order completion time. In Algorithm 2, we summarize the proposed local search procedure L1, in which condition 1 is given by $p_{i,\pi^i_k} + s_{i,\pi^i_k,\pi^i_{k+1}} > p_{i,\pi^i_{k+1}} + s_{i,\pi^i_{k+1},\pi^i_k}$, and condition 2 is given by $s_{i,\pi^i_k,\pi^i_{k+1}} + p_{i,\pi^i_k} + s_{i,\pi^i k-1,\pi^i_k} > s_{i,\pi^i_{k+1},\pi^i_k} + p_{i,\pi^i_{k+1}} + s_{i,\pi^i_{k-1},\pi^i_{k+1}}$.

Algorithm 2: Local Search L1

```
for i := 1 to m do
  for k := 1 to n-1 do
    if k = 1 then
      if condition 1 is true then
        Swap orders k and k+1
      end
    else
      if condition 2 is true then
        Swap orders k and k+1
      end
    end
  end
end
```

In the local search L2, if the summation of setup times before and after a and b are smaller than the summation of setup times before and after b and a, after pairwise interchange for a given machine i, we swap the orders a and b in Π_i. In Algorithm 2, we summarize the proposed local search procedure L1, in which the condition 3 is given by $s_{i,\pi^i_k,\pi^i_{k+1}} + s_{i,\pi^i_{j-1},\pi^i_j} + s_{i,\pi^i_j,\pi^i_{j+1}} < s_{i,\pi^i_j,\pi^i_{k+1}} + s_{i,\pi^i_{j-1},\pi^i_k} + s_{i,\pi^i_k,\pi^i_{j+1}}$, condition 4 is given by $s_{i,\pi^i_k,\pi^i_{k+1}} + s_{i,\pi^i_{j-1},\pi^i_j} < s_{i,\pi^i_j,\pi^i_{k+1}} + s_{i,\pi^i_{j-1},\pi^i_k}$, condition 5 is given by $s_{i,\pi^i_{k-1},\pi^i_k} + s_{i,\pi^i_k,\pi^i_{k+1}} + s_{i,\pi^i_{j-1},\pi^i_j} + s_{i,\pi^i_j,\pi^i_{j+1}} < s_{i,\pi^i_{k-1},\pi^i_j} + s_{i,\pi^i_j,\pi^i_{k+1}} + s_{i,\pi^i_{j-1},\pi^i_k} +$

$s_{i,\pi^i_k,\pi^i_{j+1}}$, and condition 6 is given by: $s_{i,\pi^i_{k-1},\pi^i_k} + s_{i,\pi^i_k,\pi^i_{k+1}} + s_{i,\pi^i_{j-1},\pi^i_j} < s_{i,\pi^i_{k-1},\pi^i_j} + s_{i,\pi^i_j,\pi^i_{k+1}} + s_{i,\pi^i_{j-1},\pi^i_k}$. Conditions 3 and 4 are applied if the order to be swapped is in the first position of the sequence, and conditions 5 and 6 are applied otherwise.

Algorithm 3: Local Search L2

```
for i := 1 to m
  for k := 1 to n−2
    for j := k + 2 to n
      if k == 1 then
        if j < n then
          if condition 3 is true then
            Swap orders j and k.
          end
        else
          if condition 4 is true then
            Swap orders j and k.
          end
        end
          elseif j < n
            if condition 5 is true then
            Swap orders j and k.
          end
        end
          if condition 6 is true then
            Swap orders j and k.
          end
        end
      end
    end
  end
```

5. Computational experiments

In order to establish the efficiency of the proposed algorithm, we conduct a comprehensive computational experience. In Section 5.1 we discuss the design of the experimentation, including the generation of the testbed instances and the indicators employed to evaluate the algorithms. Section 5.2 is devoted to the calibration of the proposed DDE, analysing its parameters and the contribution to its efficiency of the different local search mechanisms proposed. Next, we present in Section 5.3 the algorithms that we will use as a benchmark. Finally, in Section 5.4 we show the results obtained along with the corresponding discussion.

5.1. Experimental design

Regarding the generation of test instances of the problem, we use m = {5, 10, 20}, n = {20, 30, 40}, and $s \in \{[1,25],[1,75],[1,125]\}$ red for the number of machines, orders and jobs respectively, as suggested by Prata et al. (2021a). For combination of orders and machines we have generated 10 instances, making a total of 270 test instances. The processing times were generated according to a uniform [1, 99] distribution, as suggested by Framinan and Perez-Gonzalez (2017). The test instances proposed by Prata et al. (2021b) are available here.

The Relative Percentage Deviation (RPD) and the Success Rate (SR) have been used as performance measures. The RPD of the heuristic METHOD for a given instance is calculated as in Eq. (17):

$$\text{RPD} = \frac{v_{\text{METHOD}} - v_{\text{BEST}}}{v_{\text{BEST}}} \times 100 \tag{17}$$

where v_{METHOD} denotes the best objective function value obtained by METHOD while v_{BEST} denotes the best value of the objective function obtained. To summarize the computational results, we calculate the average RPD (ARPD) for a given method by grouping the RPD obtained across a given set of instances.

SR is calculated as the number of times that a given method finds the best solution (with or without a draw) divided by the number of test instances in a given instance set, as expressed as in Eq. (18):

$$\text{SR} = \frac{n_{\text{BEST}}}{n_{\text{INST}}} \times 100 \tag{18}$$

where n_{BEST} is the number of instances in which a given method achieved the best solution and n_{INST} is the number of instances in the given instance set.

5.2. Calibration of the proposed DDE

A standard DE algorithm basically has three parameters: NP: population size; F: scaling factor; and CR: crossover rate. Pan et al. (2011) highlighted that the NP parameter highly relies on the complexity of a given optimization problem and should be a user-specific parameter. Since we are working directly with permutations, we can suppress the scaling factor F, reducing the number of parameters of the proposed DDE.

As it can be seen, the only parameter in our algorithm is the population size. In order to study the influence of this parameter, we evaluate three levels, i.e.: $NP \in \{1/2 \times n, n, 2 \times n\}$, where n is the number of jobs. We run the DDE algorithm for these three levels and compare the ARPD and the SR values. Regarding ARPD, the values for the above-mentioned population sizes were 2.0%, 1.0%, and 0.1% respectively. Considering the SR performance indicator, the values obtained were respectively 3.0%, 15.1%, and 81.9%. Thus, we adopt the population size $NP = 2 \times n$ in the subsequent computational experiments.

In order to analyse the effectiveness of the local search mechanisms presented in Section 4.5, we have compared four variants of DDE: (i) DDE1 (using local search strategies L1 and L2); (ii) DDE2 (without L1 and with L2); (iii) DDE3 (with L2 and without L1); and (iv) DDE4 (without L1 and L2). More specifically, we performed an ANOVA test to evaluate if the difference in the ARPD values was statistically significant, see the results in Fig. 3. Furthermore, we performed a Tukey test (Montgomery, 2017) to evaluate the statistical significance of the differences among the variants, and show the confidence intervals in Fig. 4. As it can be seen in both Figs. 4 and 3, it is the addition of the specific local search mechanisms that produces a significant improvement in the performance of the DDE, particularly when both mechanisms are combined. In view of these results, we adopt DDE1 (the DDE variant including both local search mechanisms) in the subsequent computational experiments.

Besides, we also evaluate with experiments the benefits of our proposed restart procedure as compared to a standard restart operator. More specifically, we use the procedure employed by Ruiz et al. (2006), where the population is restarted after G_r generations without improvement in the best solution. After a full-factorial design taking into account three values of G_r (25, 50, and 75), the value of 25 returned the best results. Thereby, we compare our parameter-free restart procedure with a standard restart procedure with $G_r = 25$. More specifically, we compare the best DDE obtained in the calibration process with our proposed restart procedure (DDE1) with the same algorithm using the standard restart procedure, as presented by Ruiz et al. (2006) (DDE5). Fig. 5 shows the boxplot in terms of ARPD values for both DDE variants. The ANOVA test shows that there are statistically significant differences between the two algorithms. More specifically, DDE1 returns an ARPD value of 0.0% and SR equal to 100%, whereas DDE5 returns an ARPD value of 13.3%, and SR equals 0.0%. It is then clear that our proposal yields significant benefits in terms of the quality of the solutions found as compared with a standard restart procedure.

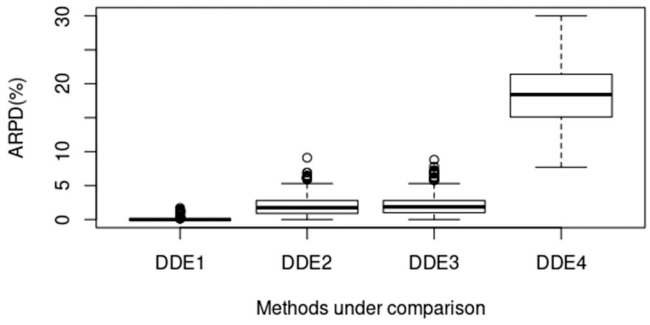

Fig. 3. Boxplot of average percent relative deviation for each DDE variant.

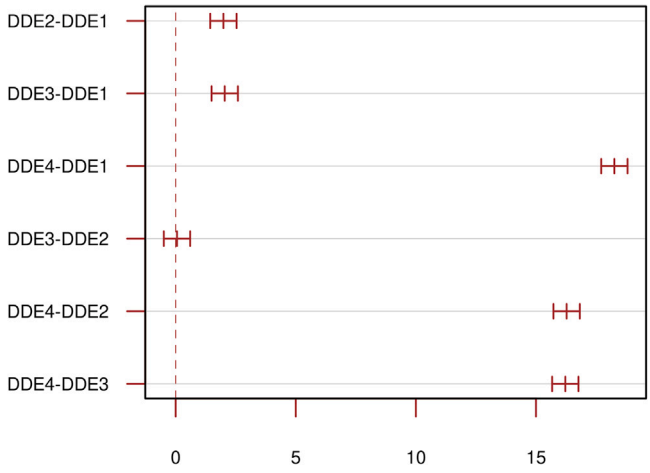

Fig. 4. Tukey confidence intervals for ARPD.

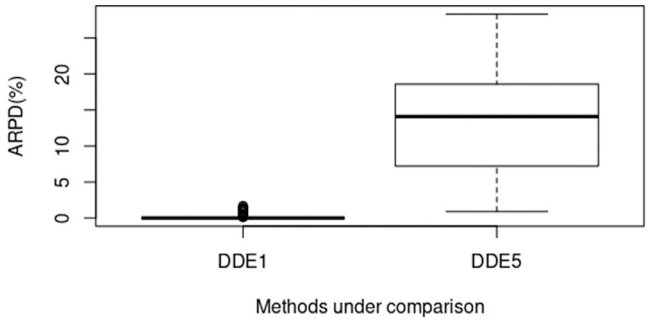

Fig. 5. Boxplot of average percent relative deviation for evaluation of the restart procedure.

5.3. Algorithms under comparison

In the computational experiments we considered the following methods:

- MILP model, defined as in Eqs. (3)–(12). In the MILP model, a different permutation is considered for each machine.
- An adaptation to the problem of the Fixed Variation List Algorithm (FVLA) (Prata et al., 2021a), the state-of-the-art algorithm for the customer order scheduling problem with makespan objective and non-zero setup times. In the FVLA matheuristic, distinct permutations are considered for each machine.
- An adaptation to the problem of the Perturbative Search Algorithm (PSA) (Riahi et al., 2019), the state-of-the-art algorithm for the customer order scheduling problem with total completion time as objective and zero setup times. In the PSA metaheuristic, a single permutation is considered for the machines.
- An adaptation to the problem of the Variable Neighbourhood Search (VNS) (Kuo et al., 2020). Since the problem under study may be regarded as related to the single-machine environment, it would be interesting to adapt algorithms for single-machine scheduling with related objectives and constraints. However, to the best of our knowledge, there are no contributions to the single-machine scheduling problem with sequence-dependent setup times to minimize the total completion time. Thus, we have considered a similar variant, as addressed by Kuo et al. (2020). Here also a single permutation is considered for the machines.
- Discrete Differential Evolution (DDE) (our proposal).

Regarding the manner in which the above algorithms are adapted, we note the following. Concerning the FVLA, we adopt $\alpha = 0.4$ since this value conducted to better results in Prata et al. (2021a). Concerning the PSA, to construct the list $\mathcal{L}$, we adopt the Total Weighted Processing Times (TWPT) used for the Permutation Construction and Explorations (PCE) heuristic to build the initial solution, following Riahi et al. (2019). We consider the same original parameters of PSA, as reported by Riahi et al. (2019): $D = 6$, $R = 0.005$, where D controls the exclusion of customer orders from the input solution, and R controls the acceptance of solutions with worse objective function values.

Regarding VNS, we adapt the objective function and the problem constraints to our variant. The initial solution is also generated randomly, and we reproduce the parameters considered by Kuo et al. (2020), i.e.: $T_s = -(Z_{max} - Z_{min})/\log_e 0.5$, $T_e = -(Z_{max} - Z_{min}) \times 0.0001/\log_e 0.05$, $\alpha_{TEMP} = 0.99$, $K = 1,000,000$, and $M = 50$. We adapt the values of Z_{max} and Z_{min} to the problem under study: $Z_{max} = n \times p_{max} + (n-1) \times s_{max}$ and $Z_{min} = n \times p_{min} + (n-1) \times s_{min}$.

5.4. Results and discussion

We implemented all the methods using Julia with Atom IDE (https://atom.io/). For the mathematical programming model as well as the matheuristic, we used the commercial solver Gurobi (https://www.gurobi.com/) version 9.0.2 with JuMP library (Lubin & Dunning, 2015). We perform the computational experience on a PC with AMD Ryzen 3 3200U APU 3.5 GHz Dual-Core and 8 GB memory, with the Windows 10 LTS operating system. For all methods under comparison, we adopt a time limit $t_{limit} = mn/2$, expressed in seconds. The computational times were not reported since all evaluated methods used the specified time limit.

Fig. 6 and Table 1 illustrate the ARPD values obtained for the methods under comparison for each instance category. Furthermore, Table 2 describes SR values for each method. Regarding Fig. 6, it can be noted that the methods with fixed permutations in the machines (i.e. PSA and VNS where the same permutation for all machines is used) presented the worst results, as they yield the highest ARPD values. It can be then concluded that, although these methods presented good solutions for the makespan criterion in Prata et al. (2021a), their adaptation to the total completion time does not seem to be equally competitive.

We can also emphasize that the method with a controlled fixing of decision variables (FVLA) presents better results than the methods with fixed permutations. Finally, it is to note that the proposed DDE metaheuristic presented better values of ARPD and SR than the rest of the other methods under comparison. We can observe that our proposal obtains the lowest ARPD values in 23 (out of 27) sets of instances. Besides, the proposed DDE yields ARPD values equal to zero in 12 of these sets, which means that it finds the best solution in all instances in these sets.

Aiming to achieve a pairwise comparison between the evaluated methods, we perform an ANOVA procedure, followed by a Tukey's

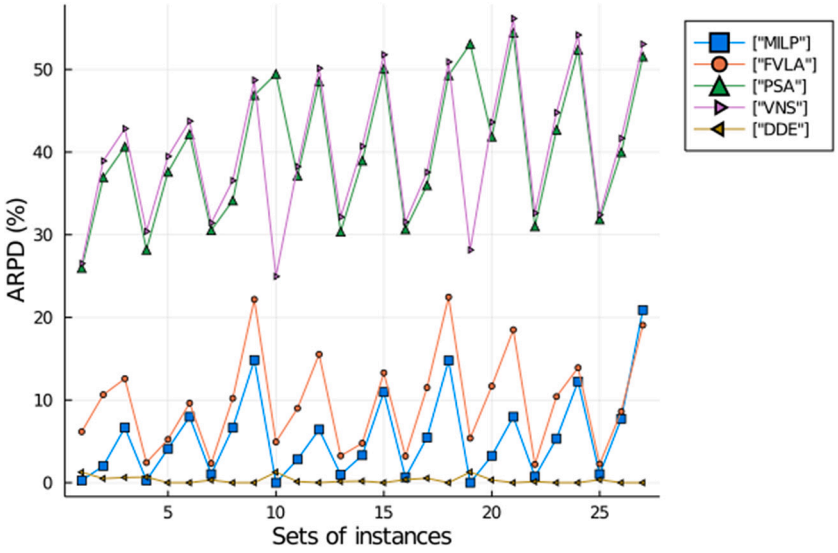

Fig. 6. Comparison of average percent relative deviation for each method by instance category.

Table 1
Average percent relative deviation and standard deviation for each method by instance category.

Set	m	n	s	MILP		FVLA		PSA		VNS		DDE	
				ARPD	σ	ARPD	σ	ARPD	σ	ARPD	σ	ARPD	σ
1	5	20	[1,25]	0.3	0.5	6.2	1.6	26	6.2	26.5	5.4	1.3	1.1
2	5	20	[1,75]	2	2.4	10.6	4.9	36.9	5.8	38.9	5.1	0.5	0.8
3	5	20	[1,125]	6.7	4.4	12.6	4.8	40.7	8.2	42.8	6.9	0.6	1.9
4	5	30	[1,25]	0.3	2.8	2.4	2.6	28.2	4.4	30.4	3.4	0.7	2.7
5	5	30	[1,75]	4.1	2.0	5.2	3.5	37.6	3.1	39.5	1.7	0	0.0
6	5	30	[1,125]	8	4.6	9.6	5.9	42.2	6.4	43.7	7.0	0	0.0
7	5	40	[1,25]	1	1.5	2.3	1.5	30.5	5.7	31.4	5.2	0.3	0.8
8	5	40	[1,75]	6.6	2.3	10.2	4.5	34.1	6.5	36.6	5.2	0	0.0
9	5	40	[1,125]	14.8	3.5	22.2	5.6	46.9	6.8	48.7	4.7	0	0.0
10	10	20	[1,25]	0	0.0	4.9	1.8	49.5	50.3	25	4.9	1.3	0.9
11	10	20	[1,75]	2.8	2.6	9	3.7	37.2	10.4	38.2	8.3	0.1	0.4
12	10	20	[1,125]	6.4	4.6	15.5	4.6	48.5	7.0	50.1	6.3	0	0.0
13	10	30	[1,25]	1	1.0	3.3	1.8	30.4	6.4	32.1	5.8	0.1	0.3
14	10	30	[1,75]	3.3	2.2	4.8	3.2	39	7.5	40.7	5.4	0.2	0.4
15	10	30	[1,125]	11	3.1	13.3	4.1	50.1	5.3	51.8	4.3	0	0.0
16	10	40	[1,25]	0.7	1.2	3.2	1.4	30.6	5.0	31.5	4.3	0.4	0.5
17	10	40	[1,75]	5.5	2.2	11.5	4.4	36	6.6	37.6	5.0	0.5	1.6
18	10	40	[1,125]	14.8	4.4	22.5	4.4	49.3	4.6	50.9	3.5	0	0.0
19	20	20	[1,25]	0	0.0	5.4	1.9	53.1	50.7	28.2	4.6	1.3	0.7
20	20	20	[1,75]	3.3	2.2	11.7	3.3	41.9	5.2	43.6	5.1	0.3	0.8
21	20	20	[1,125]	8	3.5	18.5	4.3	54.4	7.9	56.2	6.7	0	0.0
22	20	30	[1,25]	0.7	0.8	2.2	0.9	31	6.2	32.6	4.7	0.1	0.2
23	20	30	[1,75]	5.3	1.6	10.4	3.5	42.7	1.8	44.8	2.8	0	0.0
24	20	30	[1,125]	12.2	1.8	13.9	4.7	52.4	6.4	54.2	4.9	0	0.0
25	20	40	[1,25]	1	0.9	2.3	1.4	31.9	4.3	32.4	4.0	0.4	0.6
26	20	40	[1,75]	7.7	3.0	8.6	2.8	40	3.3	41.7	2.9	0	0.0
27	20	40	[1,125]	20.9	8.4	19.1	2.9	51.6	4.1	53.1	3.4	0	0.0

test. Fig. 7 illustrates the boxplots for ARPD values, and Fig. 8 illustrates the Tukey multiple comparisons of means with 95% family-wise confidence level. From these results, it can be seen that there are statistically significant differences among most procedures (see Fig. 7 where the 95% confidence intervals of ARPD are illustrated). Based on the results obtained, we can conclude that the proposed DDE algorithm outperforms all the other methods under comparison with statistical significance.

Fig. 9 illustrates an example of convergence graph for the proposed DDE, considering the test instance 270 ($m = 20$, $n = 40$, $s \in 1, 125$). Taking this figure into account, we can observe that the DDE algorithm was able to improve the best solution found during the generations. These improvements were more substantial in the first generations, and more gradual in the next ones. Furthermore, it can be observed that the DDE algorithm has not presented an asymptotic convergence. Thereby, with a greater time limit, the algorithm could potentially find better solutions for large-sized test instances.

Table 2
Success rate for each method.

Method	SR (%)
MILP	20.4
FVLA	9.6
PSA	0.0
VNS	0.0
DDE	70.0

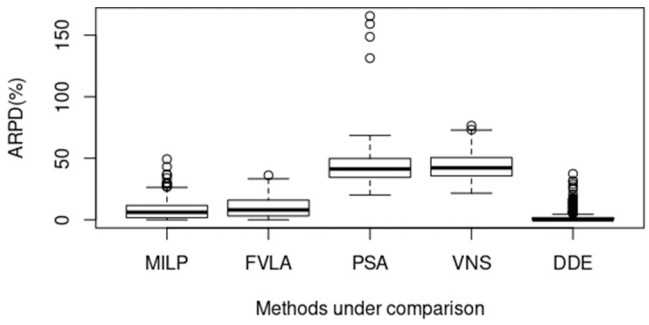

Fig. 7. Boxplot of average percent relative deviation for each method.

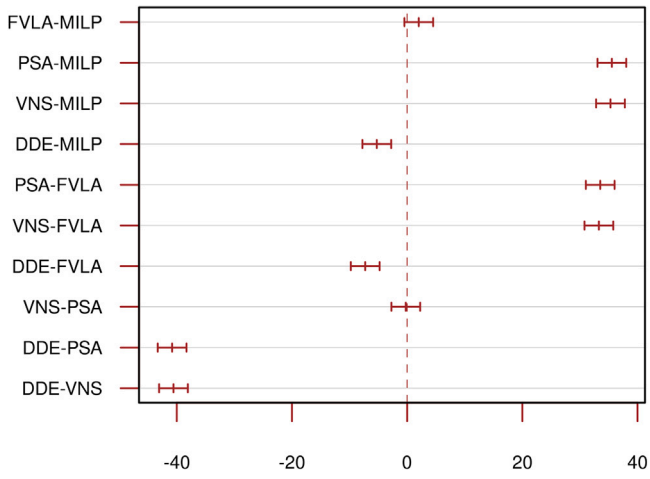

Fig. 8. Tukey confidence intervals for ARPD.

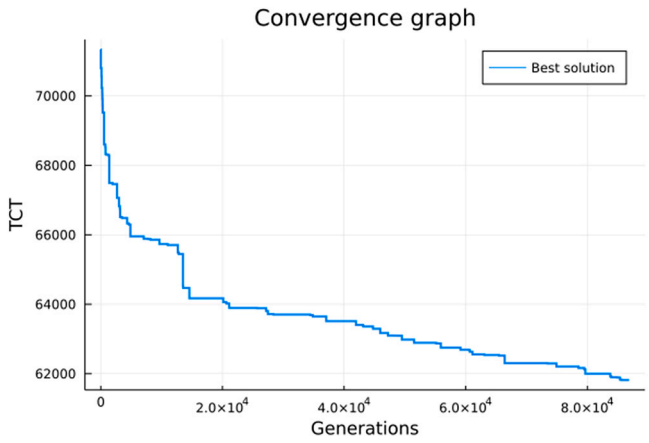

Fig. 9. Convergence graph for the test instance p270.

6. Final remarks and perspectives

In this paper, we have addressed the customer order scheduling problem with sequence-dependent setup times using the total completion time as objective function. To the best of our knowledge, this problem has not been dealt with in the previous literature.

Since the problem considered is NP-hard, we have developed a Discrete Differential Evolution (DDE) algorithm to solve it. The operators work directly with the permutation space so the operations with float numbers – characteristic in Differential Evolution algorithms – are not necessary. A new, parameter-free, restart procedure is introduced, and we propose two novel local search mechanisms based on heuristic dominance relations aiming to reduce the search space. Furthermore, the calibration of the DDE shows the effectiveness of these local search mechanisms. The extensive computational experimentation carried out indicates that the proposed DDE clearly outperforms all other methods under comparison.

As suggestion for future works, we are interested in expanding our research along different lines. One of them is the consideration within our customer order scheduling problem with sequence-dependent setups of other performance measures. Particularly, we are interested in due-date related criteria widely used in industry, such as total tardiness minimization. Other line refers to improving the proposed DDE so the only remaining parameter (i.e. the population size in each generation), can be removed, thus making the algorithm more robust. Finally, it would be interesting to further analyse the similarities between the problem under consideration and other scheduling layouts where there are numerous contributions considering sequence-dependent setups, so further problem-specific properties (similar to the ones successfully employed here to derive the local search mechanisms) could be potentially derived.

CRediT authorship contribution statement

Bruno de Athayde Prata: Conceptualization, Investigation, Software, Formal analysis, Writing – original draft, Visualization, Funding acquisition. **Carlos Diego Rodrigues:** Formal analysis, Methodology, Writing. **Jose Manuel Framinan:** Conceptualization, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This study was financed in part by the Coordination for the Improvement of Higher Education Personnel (CAPES), Brazil and the National Council for Scientific and Technological Development (CNPq), Brazil, through grant 303594/2018-7. The support of the Spanish Ministry of Science and Innovation, Spain via the ASSORT grant with reference PID2019-108756RB-I00, and the Andalusian Regional Government, Spain via grants DEMAND and EFECTOS with references P18-FR-1149 and US-1264511 respectively is acknowledged and appreciated.

References

Abreu, L. R., Cunha, J. O., Prata, B. A., & Framinan, J. M. (2020). A genetic algorithm for scheduling open shops with sequence-dependent setup times. *Computers & Operations Research*, *113*, Article 104793.

Alcaraz, J., Maroto, C., & Ruiz, R. (2003). Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. *Journal of the Operational Research Society*, *54*(6), 614–626.

Allahverdi, A., Ng, C., Cheng, T., & Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, *187*(3), 985–1032.

Bai, D., Zhang, Z., Zhang, Q., & Tang, M. (2017). Open shop scheduling problem to minimize total weighted completion time. *Engineering Optimization*, *49*(1), 98–112.

Framinan, J. M., & Perez-Gonzalez, P. (2017). New approximate algorithms for the customer order scheduling problem with total completion time objective. *Computers & Operations Research*, *78*, 181–192.

Framinan, J. M., & Perez-Gonzalez, P. (2018). Order scheduling with tardiness objective: Improved approximate solutions. *European Journal of Operational Research*, *266*(3), 840–850.

Framinan, J. M., Perez-Gonzalez, P., & Fernandez-Viagas, V. (2019). Deterministic assembly scheduling problems: A review and classification of concurrent-type scheduling models and solution procedures. *European Journal of Operational Research*, *273*(2), 401–417.

Kuo, Y., Chen, S.-I., & Yeh, Y.-H. (2020). Single machine scheduling with sequence-dependent setup times and delayed precedence constraints. *Operational Research*, *20*(2), 927–942.

Leung, J. Y.-T., Li, H., & Pinedo, M. (2005). Order scheduling in an environment with dedicated resources in parallel. *Journal of Scheduling*, *8*(5), 355–386.

Liu, W.-L., Gong, Y.-J., Chen, W.-N., Liu, Z., Wang, H., & Zhang, J. (2020). Coordinated charging scheduling of electric vehicles: A mixed-variable differential evolution approach. *IEEE Transactions on Intelligent Transportation Systems*, *21*(12), 5094–5109.

Lu, Y., Zhou, J., Qin, H., Wang, Y., & Zhang, Y. (2011). Chaotic differential evolution methods for dynamic economic dispatch with valve-point effects. *Engineering Applications of Artificial Intelligence*, *24*(2), 378–387.

Lubin, M., & Dunning, I. (2015). Computing in operations research using Julia. *INFORMS Journal on Computing*, *27*(2), 238–248.

Moccellin, J. V., Nagano, M. S., Pitombeira Neto, A. R., & Prata, B. A. (2018). Heuristic algorithms for scheduling hybrid flow shops with machine blocking and setup times. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, *40*(2), 40.

Montgomery, D. C. (2017). *Design and analysis of experiments*. John wiley & sons.

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

Onwubolu, G., & Davendra, D. (2009). Differential evolution for permutation—based combinatorial problems. In *Differential evolution: A handbook for global permutation-based combinatorial optimization* (pp. 13–34). Springer.

Pan, Q.-K., Suganthan, P., Wang, L., Gao, L., & Mallipeddi, R. (2011). A differential evolution algorithm with self-adapting strategy and control parameters. *Computers & Operations Research*, *38*(1), 394–408, Project Management and Scheduling.

Pan, Q.-K., Tasgetiren, M. F., & Liang, Y.-C. (2008). A discrete differential evolution algorithm for the permutation flowshop scheduling problem. *Computers & Industrial Engineering*, *55*(4), 795–816.

Prata, B. A. (2015). A hybrid genetic algorithm for the vehicle and crew scheduling in mass transit systems. *IEEE Latin America Transactions*, *13*(9), 3020–3025.

Prata, B. A., Rodrigues, C. D., & Framinan, J. M. (2021a). Customer order scheduling problem to minimize makespan with sequence-dependent setup times. *Computers & Industrial Engineering*, *151*, Article 106962.

Prata, B. A., Rodrigues, C. D., & Framinan, J. M. (2021b). Test instances - customer order scheduling with sequence-dependent setup times to minimize the total completion time objective.

Riahi, V., Newton, M. H., Polash, M., & Sattar, A. (2019). Tailoring customer order scheduling search algorithms. *Computers & Operations Research*, *108*, 155–165.

Roemer, T., & Ahmadi, R. (2001). The complexity of scheduling customer orders.

Ruiz, R., Maroto, C., & Alcaraz, J. (2006). Two new robust genetic algorithms for the flowshop scheduling problem. *Omega*, *34*(5), 461–476.

Sales, L. P. A., Melo, C. S., Bonates, T. O., & Prata, B. A. (2018). Memetic algorithm for the heterogeneous fleet school bus routing problem. *Journal of Urban Planning and Development*, *144*(2), 1–12.

Santucci, V., Baioletti, M., & Milani, A. (2016). Solving permutation flowshop scheduling problems with a discrete differential evolution algorithm. *AI Communications*, *29*(2), 269–286.

Shi, Z., Wang, L., Liu, P., & Shi, L. (2017). Minimizing completion time for order scheduling: Formulation and heuristic algorithm. *IEEE Transactions on Automation Science and Engineering*, *14*(4), 1558–1569.

Storn, R., & Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, *11*(4), 341–359.

Sung, C. S., & 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.

Tavazoei, M. S., & Haeri, M. (2007). Comparison of different one-dimensional maps as chaotic search pattern in chaos optimization algorithms. *Applied Mathematics and Computation*, *187*(2), 1076–1085.

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

Wang, G., & Cheng, T. E. (2007). Customer order scheduling to minimize total weighted completion time. *Omega*, *35*(5), 623–626.

Wang, L., Pan, Q.-K., Suganthan, P., Wang, W.-H., & Wang, Y.-M. (2010). A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems. *Computers & Operations Research*, *37*(3), 509–520, Hybrid Metaheuristics.

Wu, C.-C., Bai, D., Zhang, X., Cheng, S.-R., Lin, J.-C., Wu, Z.-L., & Lin, W.-C. (2021). A robust customer order scheduling problem along with scenario-dependent component processing times and due dates. *Journal of Manufacturing Systems*, *58*, 291–305.

Wu, C.-C., Lin, W.-C., Zhang, X., Chung, I.-H., Yang, T.-H., & Lai, K. (2019). Tardiness minimisation for a customer order scheduling problem with sum-of-processing-time-based learning effect. *Journal of the Operational Research Society*, *70*(3), 487–501.

Wu, C.-C., Yang, T.-H., Zhang, X., Kang, C.-C., Chung, I.-H., & Lin, W.-C. (2019). Using heuristic and iterative greedy algorithms for the total weighted completion time order scheduling with release times. *Swarm and Evolutionary Computation*, *44*, 913–926.

Zhao, F., Zhao, L., Wang, L., & Song, H. (2020). An ensemble discrete differential evolution for the distributed blocking flowshop scheduling with minimizing makespan criterion. *Expert Systems with Applications*, *160*, Article 113678.

Zhou, S., Xing, L., Zheng, X., Du, N., Wang, L., & Zhang, Q. (2021). A self-adaptive differential evolution algorithm for scheduling a single batch-processing machine with arbitrary job sizes and release times. *IEEE Transactions on Cybernetics*.