Swarm and Evolutionary Computation 51 (2019) 100601

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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

An adaptive multi-objective algorithm based on decomposition and large neighborhood search for a green machine scheduling problem

Luciano P. Cota [a,*], Frederico G. Guimarães [b], Roberto G. Ribeiro [a], Ivan R. Meneghini [b], Fernando B. de Oliveira [c], Marcone J.F. Souza [d], Patrick Siarry [e]

[a] Instituto Tecnológico Vale, Ouro Preto, MG, Brazil
[b] Machine Intelligence and Data Science (MINDS) Laboratory, Department of Electrical Engineering, Universidade Federal de Minas Gerais, Belo Horizonte, MG, 31270-010, Brazil
[c] Department of Computer and Systems, Universidade Federal de Ouro Preto, João Monlevade, MG, 35931-008, Brazil
[d] Department of Computer Science, Universidade Federal de Ouro Preto, Ouro Preto, MG, 35400-000, Brazil
[e] Université Paris-Est Créteil, LiSSi, 61 Avenue du Général de Gaulle 94010 Créteil, Cedex, France

ARTICLE INFO

Keywords:
Green scheduling
Multi-objective optimization
Parallel machines
Adaptive large neighborhood search
Learning automata
Decomposition and aggregation

ABSTRACT

Green machine scheduling consists in the allocation of jobs in order to maximize production, in view of the sustainable use of energy. This work addresses the unrelated parallel machine scheduling problem with setup times, with the minimization of the makespan and the total energy consumption. The latter takes into account the power consumption of each machine in different operation modes. We propose multi-objective extensions of the Adaptive Large Neighborhood Search (ALNS) metaheuristic with Learning Automata (LA) to improve the search process and to solve the large scale instances efficiently. ALNS combines ad-hoc destroy and repair (also named removal and insertion) operators and a local search procedure. The LA is used to adapt the selection of insertion and removal operators within the framework of ALNS. Two new algorithms are developed: the MO-ALNS and the MO-ALNS/D. The first algorithm is a direct extension of single objective ALNS by using multi-objective local search. As this method does not offer much control of the diversification of the Pareto front approximation, a second strategy employs the decomposition approach similar to MOEA/D algorithm. The results show that the MO-ALNS/D algorithm has better performance than MO-ALNS and MOEA/D in all indicators. These findings show that the decomposition strategy is beneficial not only for evolutionary algorithms, but it is indeed an efficient way to extend ALNS to multi-objective problems.

1. Introduction

Sustainable use of energy is essential into the next industrial revolution, so-called Industry 4.0. Reference [1] links the thoughts of this new revolution with the incorporation of advancements in Information Technology, including computational optimization techniques, into manufacturing technology and systems. According to Ref. [2], the concepts behind sustainable manufacturing are conservation of energy, material and value-added products, waste prevention and environmental protection. Regarding saving energy in the context of advanced manufacturing systems, one kind of problem which needs attention is the green machine scheduling. The mentioned problem consists of balancing the makespan and the total energy consumption of the machines. It is a challenging multi-objective optimization problem with combinatorial nature. Therefore, for large instances, heuristic-based approaches are required given the practical limits of exact methods.

Machine scheduling problems have great relevance for the industries. These problems involve the maximization of the production by allocating the jobs to the available resources in an optimal way. According to Ref. [3], this type of problem addresses the allocation of resources for tasks or services by periods to optimize one or more objectives. Single objective scheduling seeking to minimize the *makespan*, defined by the notation $R_M|S_{ijk}|C_{\max}$, is treated in various works in the literature. This kind of scheduling problem appears in many manufacturing industries, such as textile, chemical, semiconductor and ink [4]. Besides, this problem belongs to the $\mathcal{NP}$-hard class. It is a generalization of the *par-*

* Corresponding author.
E-mail addresses: luciano.p.cota@itv.org (L.P. Cota), fredericoguimaraes@ufmg.br (F.G. Guimarães), rogorib@ufmg.br (R.G. Ribeiro), ivan.reinaldo@ifmg.edu.br (I.R. Meneghini), fboliveira@ufop.edu.br (F.B. de Oliveira), marcone@ufop.edu.br (M.J.F. Souza), siarry@u-pec.fr (P. Siarry).

https://doi.org/10.1016/j.swevo.2019.100601
Received 11 February 2019; Received in revised form 17 July 2019; Accepted 22 October 2019
Available online XXX

allel machine scheduling problem with identical machines and without setup times, which has been proven to belong to that class in Refs. [5,6]. Reference [7] presents a comprehensive up to date survey which addresses the topic, classifying and comparing 500 papers of scheduling problems. We review some of these works as follows. In Ref. [4] the authors proposed a metaheuristic for randomized priority search and a mathematical model. Genetic algorithms and a mathematical model have been implemented in Ref. [8]. In the works [9–11], the authors have proposed several algorithms that combine the Iterated Local Search and Variable Neighborhood Descent [12] metaheuristics. Two hybrid models based on logic-based Benders decomposition and Branch-and-check have been defined in Ref. [13]. An Adaptive Large Neighborhood Search metaheuristic with Learning Automata has been presented in Ref. [14].

As reported by the US Energy Information Administration, the industry sector is responsible for the consumption of 54% of the total energy generated in the world. The predominant energy sources are natural gas and electricity. Since the machines are responsible for most of the electricity consumption in the industries, machine scheduling problems have great relevance in this scenario. Scheduling problems have been studied since the years before World War II [15]. Nevertheless, the sustainable use of energy has become relevant only more recently. Research in this area is defined as green scheduling, see for instance Ref. [16].

Green scheduling in machine scheduling problems is still underexplored in the literature. Many studies in the literature deal with flow shop scheduling problems, as follows. In Ref. [17], the authors have constructed a mathematical model for solving a problem with the objective of minimizing the electricity costs and carbon emissions. In Ref. [18] the authors have implemented a multi-objective Iterated Greedy algorithm for solving a problem seeking to minimize the *makespan* and total carbon emissions. In Ref. [2] a mathematical model and a constructive heuristic have been proposed to solve a problem attempting to minimize the *makespan* and total energy consumption. In Ref. [19] the authors have implemented a mathematical model to minimize electricity consumption and total weighted tardiness. In Ref. [20] a genetic algorithm has been defined to solve a problem seeking to minimize the *makespan* and the energy consumption.

With regard to energy optimization aspects, production scheduling with electrical energy constraint has been addressed in Ref. [21]. Total energy cost in a single-machine manufacturing environment has been studied in Ref. [22], and the complexity analysis of those problems on energy states of machines has been reviewed in Ref. [23]. The objective is to minimize the total energy consumption costs. Besides, the trade-off between makespan and energy consumption has been investigated in Ref. [24] for two-machine permutation flowshop scheduling problem with sequence dependent setup times. Those machines have a variable speed and this feature was incorporated in the mathematical model. Pareto fronts were used to present that trade-off in this problem.

Bi-objective optimization in green scheduling problems are performed in different context. The single machine batch scheduling is proposed by Ref. [25] and the objectives are to minimize the makespan and the total energy costs. The scheduling problem with identical parallel machine has been studied in Ref. [26]. The objectives are to minimize total energy consumption and makespan. The problem with parallel machine is also addressed in Ref. [27]. Besides, those authors consider machines with different processing cost rates. The makespan and the total cost (namely total green cost) are minimized. A memetic differential evolution algorithm has been proposed in Ref. [28] to solve an unrelated parallel machine scheduling problem. The objectives are to minimize total energy consumption and makespan.

Metaheuristics based on local search methods, such as Iterated Local Search [29] and Adaptive Large Neighborhood Search (ALNS) [30], have been successfully applied to solve single objective combinatorial optimization problems, including scheduling problems, in a number of studies [9,14,31–33]. In the literature, there are several adaptations of these metaheuristics to solve multi-objective problems [34–36]. However, many of those adaptations do not have the proper structure to deal with the essential requirements of the multi-objective problems, such as the diversity control in the Pareto front approximation. Population-based metaheuristics have been very effective for approaching multi-objective problems, with the advantage of evolving a set of solutions simultaneously and having specific operations to enforce diversity on the Pareto front approximation [37]. Evolutionary algorithms such as MOEA/D [38], NSGA-II [39], NSGA-III [40], and SPEAII [41], that are based on genetic operators to explore the search space, are among the most well-known algorithms. Nevertheless, there is enough evidence that embedding local search methods in these evolutionary algorithms brings a great impact on their performance in combinatorial optimization problems [42–44]. Accordingly, the combination of metaheuristics based on extensions of local search methods and pure population-based multi-objective metaheuristics might generate effective and powerful algorithms for solving multi-objective scheduling problems. This is the main motivation for the present study.

More recently, in Ref. [45], the authors presented a new formulation for the unrelated parallel machine scheduling problem with setup times, attempting to minimize the *makespan* and the total energy consumption. The energy consumption is calculated considering the power consumption of each machine in different operation modes. In Ref. [45], the authors developed a Mixed Integer Linear Programming (MILP) model for the problem and show that the objectives *makespan* and total energy consumption are conflicting. In addition, the authors have shown the importance of considering energy consumption in the problem. Two exact methods to solve the problem $R_M|S_{ijk}|(C_{\max}, TEC)$ have been presented. Due to the limitation of the exact methods in solving large problems in a restricted time, only small and medium-sized instances were solved. The aim of our work is to propose multi-objective algorithms to solve the large-sized instances efficiently. The main objective is to develop multi-objective algorithms with learning techniques to improve the search process. This strategy is a significant extension on previous work proposed in Ref. [14]. That algorithm is an ALNS with Learning Automata that is able to learn the best momentary choice of insertion and removal methods. The ALNS with Learning Automata obtained a good performance to solve the single objective problem $R_M|S_{ijk}|C_{\max}$. In this paper we extend this ALNS framework to deal with the problem $R_M|S_{ijk}|(C_{\max}, TEC)$. This paper extends the previous work by presenting: (i) multi-objective algorithms based on decomposition features of the MOEA/D and the Learning Automata strategy used; (ii) a solution for large-sized instances of a green scheduling machine problem. Although the fact that this problem is quite relevant in the industry 4.0 context, the literature did not show until nowadays a sufficient scientific effort to tackle it. Therefore, we seek to generate advanced metaheuristics to address this green manufacturing problem.

Regarding such metaheuristics, we create two multi-objective algorithms for solving large-sized instances of $R_M|S_{ijk}|(C_{\max}, TEC)$ problems. The first algorithm is a multi-objective version of the mono-objective ALNS with Learning Automata, which is called MO-ALNS, employing Pareto local search. A local search is added to explore the changes in modes of operation. This algorithm does not control the diversification of the Pareto front approximation. This characteristic might be a disadvantage. The second algorithm is a combination of the MOEA/D algorithm [38] and the single-objective ALNS with Learning Automata. This method is called MO-ALNS/D. In the classical MOEA/D, a multi-objective problem is decomposed into S single objective optimization subproblems using aggregation methods. The subproblems are generated using weight vectors uniformly distributed; the exploration of the subproblems is performed by means of genetic operators. The MO-ALNS/D has the decomposition structure of classical MOEA/D, but it uses the ALNS with Learning Automata to explore the scalar subproblems instead of the genetic operators. Unlike the MO-ALNS, the MO-ALNS/D has better control of the diversification of the Pareto front approximation. This characteristic is inherited from the MOEA/D.

The rest of this paper is organized as follows. The addressed problem is detailed in Section 2. The proposed multi-objective algorithms are introduced in Section 3. In Section 4, the results on computational experiments are reported and discussed. Finally, Section 5 presents the conclusions and future works.

2. Problem definition

The problem has the following characteristics. There are a set of non-preemptive jobs, a set of parallel and independent machines and a set of different operation modes. Each operation mode is related to a corresponding speed of processing the job and power consumption. Therefore, there is a processing time to perform a job on a given machine. There is a setup time to prepare the machine for processing the jobs, and this setup time depends on the order of allocation of the jobs. Each machine has a power consumption with normal speed of operation.

The objectives are to allocate all the jobs on the machines using the operation modes, seeking to minimize the *makespan* and the total energy consumption. This problem can be described as $R_M|S_{ijk}|(C_{\max}, TEC)$, according with the classical notation for machine scheduling problems [46]. R_M represents the unrelated parallel machines, S_{ijk} the setup times, $C_{\max}$ the *makespan* and TEC the total energy consumption. These two objectives have great importance in the problem. The minimization of the *makespan* usually implies an efficient use of the machines [47]. Meanwhile, the minimization of total energy consumption implies reducing costs for industries and the conscious use of environmental resources. Furthermore, the objectives have conflicting nature because there is a trade-off between maximizing production with respect to completion time and minimizing power consumption by allocating jobs to those machines with low power consumption and processing jobs at lower speeds.

The mathematical model for the problem $R_M|S_{ijk}|(C_{\max}, TEC)$ has been defined initially in Ref. [45] and we reproduce it here for the sake of completeness. Let:

- $M = \{1, \ldots, m\}$: set of machines with m being the number of machines;
- $N = \{1, \ldots, n\}$: set of jobs with n being the number of jobs;
- $L = \{1, \ldots, o\}$: set of o different modes of operation with o being the number of modes of operation. Each mode is related to a corresponding speed of operation and power consumption;
- p_{ik}: processing time of job k in machine i [minutes];
- S_{ijk}: setup time necessary for processing the job k in machine i after job j [minutes];
- π_i: power consumption of machine i at normal speed of operation [kW];
- v_l: multiplying factor of speed at normal operation, with $l \in L$;
- λ_l: multiplying factor of power at normal speed, with $l \in L$;
- B: big constant.

The factor v_l is a non-decreasing function of λ_l, and these factors are the same on all machines. The relation between these factors is given below.

$$v_l \text{ and } \lambda_l = \begin{cases} v_l = 1 \text{ and } \lambda_l = 1, & \text{normal speed of machine operation} \\ 0 < v_l < 1 \text{ and } 0 < \lambda_l < 1, & \text{speed slower than normal, then the machine consumes less power} \\ v_l > 1 \text{ and } \lambda_l > 1, & \text{speed greater than normal, then the machine consumes more power} \end{cases}$$

Decision variables used in the mathematical model are:

$$x_{ijkl} = \begin{cases} 1, & \text{if job } k, \text{ with operation mode } l, \text{ is allocated immediately after job } j \text{ in machine } i \\ 0, & \text{otherwise} \end{cases}$$

Auxiliary variables used in the model are:

- C_j: completion time of job j;
- O_i: completion time of machine i;
- $C_{\max}$: maximum processing time of all machines (*makespan*);
- TEC: total energy consumption [kWh].

In the model, a fictitious job 0 is allocated at the beginning of each machine with processing time and setup time equal to 0 ($p_{i0} = 0 \forall i \in M$ and $S_{i0k} = 0 \forall i \in M, \forall k \in N$). The mathematical model is described in Eqs. (1)–(12):

$$\min C_{\max} \tag{1}$$

$$\min TEC \tag{2}$$

Subject to:

$$\sum_{i=1}^{m} \sum_{\substack{j=0 \\ j \neq k}}^{n} \sum_{l=1}^{o} x_{ijkl} = 1 \, \forall k \in N \tag{3}$$

$$\sum_{i=1}^{m} \sum_{\substack{k=1 \\ j \neq k}}^{n} \sum_{l=1}^{o} x_{ijkl} \leq 1 \, \forall j \in N \tag{4}$$

$$\sum_{k=1}^{n} \sum_{l=1}^{o} x_{i0kl} \leq 1 \forall \, i \in M \tag{5}$$

$$\sum_{\substack{k=0 \\ k \neq j}}^{n} \sum_{l=1}^{o} x_{ijkl} - \sum_{\substack{h=0 \\ h \neq j}}^{n} \sum_{l=1}^{o} x_{ihjl} = 0 \, \forall j \in N, \forall i \in M \tag{6}$$

$$C_k - C_j + B\left(1 - x_{ijkl}\right) \geq S_{ijk} + \frac{p_{ik}}{v_l} \qquad \forall j \in N_0, \forall k \in N, \; j \neq k, \forall l \in L, \; \forall i \in M \tag{7}$$

$$C_0 = 0 \tag{8}$$

$$\sum_{j=0}^{n} \sum_{\substack{k=1 \\ k \neq j}}^{n} \sum_{l=1}^{o} \left(S_{ijk} + \frac{p_{ik}}{v_l}\right) x_{ijkl} = O_i \forall i \in M \tag{9}$$

$$C_{\max} \geq O_i \forall i \in M \tag{10}$$

$$TEC \geq \sum_{i=1}^{m} \sum_{j=0}^{n} \sum_{\substack{k=1 \\ j \neq k}}^{n} \sum_{l=1}^{o} \left(\lambda_l \times \frac{\pi_i}{60} \times \frac{p_{ik}}{v_l}\right) x_{ijkl} \tag{11}$$

$$x_{ijkl} \in \{0, 1\} \qquad \forall j \in N_0, \forall k \in N, \; j \neq k, \forall i \in M, \; \forall l \in L \tag{12}$$

The objective function is to minimize the *makespan* (1) and the total energy consumption (2). Constraint (3) defines that each job will be allocated to only one machine and it has a predecessor. This constraint also defines that the allocation has a unique operation mode.

Table 1
Setup times of machines M_1 and M_2.

M_1	1	2	3	4	5	6	M_2	1	2	3	4	5	6
1	0	1	8	1	3	9	1	0	5	1	6	1	7
2	4	0	7	3	7	8	2	6	0	7	7	6	2
3	7	3	0	2	3	5	3	7	6	0	9	6	9
4	3	8	3	0	5	2	4	3	7	3	0	1	7
5	8	3	7	9	0	5	5	5	8	5	6	0	9
6	8	8	1	2	2	0	6	7	4	1	7	9	0

Constraints (4) and (5) ensure that each job will have, at most, one successor job. Constraint (6) defines the right order for allocating job. Constraint (7) calculates the accumulated time of each job. If $x_{ijkl} = 1$, the accumulated time of k is C_j plus S_{ijk} and $\left(\frac{p_{ik}}{v_l}\right)$. If $x_{ijkl} = 0$, the constant B will ensure the constraints are satisfied. Constraint (8) defines that the accumulated time of the fictitious job is equal to 0. Constraint (9) defines the calculation of the accumulated costs for each machine, given by the sum of the setup time and the processing time of all jobs allocated to a given machine. Constraint (10) defines the value of the *makespan* (or $C_{\max}$). Constraint (11) defines the calculation of total energy consumption (or *TEC*). This calculation uses the processing times (p_{ik}/v_l), the power input of each machine at normal speed of operation (π_i), and the multiplying factors (λ_l) and (v_l). The machine power is divided by 60 because the units of measurement. The power is given in kW, while the processing time is given in minutes, and the *TEC* is calculated in kWh. Finally, constraint (12) defines which variables are binary. N_0 is the set of jobs with the fictitious job 0. This mathematical model has n^2mq binary variables, $n + m + 2$ continuous variables and $2n + 3$ m $+ nm + 2n^2mq + 2$ constraints.

An instance with six jobs, two machines and one operation mode (normal speed of operation with $v_l = 1$ and $\lambda_l = 1$) was chosen to illustrate this problem. The processing times and the power consumption with normal speed of operation of the machines $M1$ and $M2$ are given below. Setup times of those machines are shown in Table 1.

$$M1 \ : \ p_{1j} = \{70, 87, 28, 32, 38, 9\}, \pi_1 = 70$$

$$M2 \ : \ p_{2j} = \{4, 21, 68, 17, 43, 48\}, \pi_2 = 179$$

The solver IBM ILOG CPLEX version 12.5 was used to solve this instance. Fig. 1 illustrates the optimal solution for the scheduling minimizing only the *makespan*. In this solution, the *makespan* is equal to 75 and the total energy consumption is equal to 283.36. Meanwhile, Fig. 2 illustrates the optimal solution for scheduling minimizing only the total energy consumption. The *makespan* of this allocation is equal to 119 and the total energy consumption is equal to 199.41.

It is possible to observe that the job allocations and objective function values are quite different in both figures. This example illustrates the conflicting nature of the two objectives in this machine scheduling problem, even with only one operation mode.

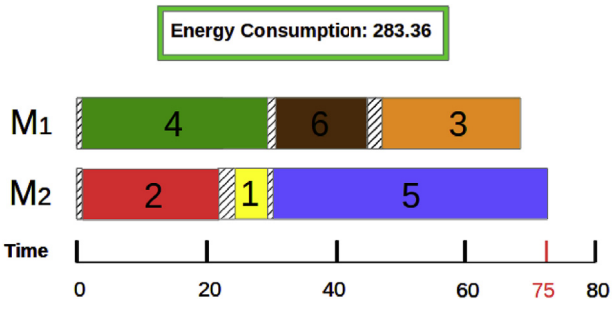

Fig. 1. Optimal solution for the minimization of the *makespan*.

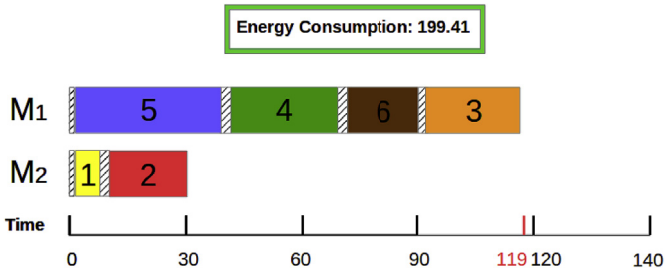

Fig. 2. Optimal solution for the minimization of the total energy consumption.

3. Proposed multi-objective algorithms

This section presents the two proposed multi-objective algorithms. At first, a solution representation and evaluation are shown. Then, algorithms are defined.

3.1. Solution representation and evaluation

A solution is represented by two data structures. One structure is applied to represent job allocations on machines and another one to represent the operation mode, in which each job is processed. Each operation mode is related to a corresponding speed of operation and power consumption.

In order to represent the allocation of jobs, a vector of integers with m positions is used, in which m is the total number of machines. A list is associated with each position of this vector and represents the jobs allocated to each machine. The operation modes are represented by a vector of integers. Fig. 3 shows a possible solution for an instance with two machines, seven jobs and three operation modes. Jobs 7, 3 and 4 are allocated on machine M_1, in that order. Jobs 2, 1, 6 and 5 are allocated on machine M_2, in that order. Operation modes are defined as follows: *i*) operation mode 1: jobs 3 and 4; *ii*) operation mode 2: jobs 1, 5 and 7; *iii*) and operation mode 3: jobs 2 and 6.

A solution is evaluated by means of *makespan* (as defined in Eq. (10), Section 2) and the total energy consumption (as defined in Eq. (11), Section 2).

3.2. MO-ALNS algorithm

Reference [14] addresses the Unrelated Parallel Machine Scheduling Problem with Setup Times with the single objective of minimizing the makespan. The authors propose an Adaptive Large Neighborhood Search [30] metaheuristic that uses Learning Automata to learn the best momentary choice of insertion and removal methods. The results show that the ALNS algorithm with Learning Automata is an attractive method to solve single objective machine scheduling problems. Taking this into account, the MO-ALNS algorithm is a multi-objective version of the ALNS with Learning Automata, by including a multi-objective local search procedure.

LA is an adaptive decision unit that improves its performance by means of repeated interactions in an unknown random environment [48–50], Narendra et al. [51] defines it as the following 6-tuple:

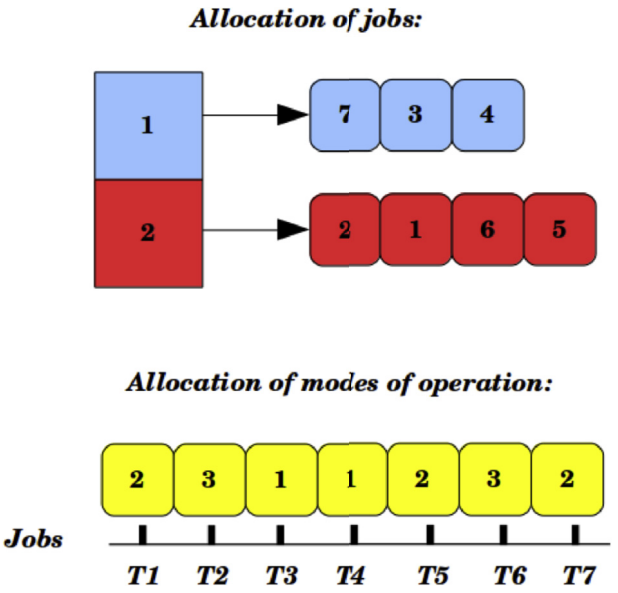

Fig. 3. Solution representation.

$(\varphi, \alpha, \beta, A, \pi, p)$. The terms φ, α and β denote sets of internal states, outputs or actions of learning automata and responses from the environment, respectively. The term A represents a Learning Automaton, while $\pi : \varphi \mapsto \alpha$ is a function that maps the current state into a current output, while p is a vector that defines a selection probability of an action on each stage.

The action is randomly chosen by means of a probability distribution over a set of actions. At each interaction, the selected action $j \in \alpha$ is used as input to the random environment for further learning. As a response, such environment returns to the LA the applied action with a noise signal. Let β_k be a response from the environment in step k with $\beta_k \in \{0, 1\}$, in which responses 0 and 1 mean "agreeable" and "disagreeable", respectively. If the response is "agreeable", the probability of each action $i \in \alpha$ might be updated with Equation (13). Otherwise, if the response is "disagreeable", the probability of each action $i \in \alpha$ is updated with Equation (14) [52].

$$p_i(k+1) = \begin{cases} p_i(k) + a(1 - p_i(k)) & \text{if } i = j \\ p_i(k)(1-a) & \text{if } i \neq j \end{cases} \tag{13}$$

$$p_i(k+1) = \begin{cases} p_i(k)(1-b) & \text{if } i = j \\ \frac{b}{r-1} + p_i(k)(1-b) & \text{if } i \neq j \end{cases} \tag{14}$$

The parameters a, b and r are the reward, the penalty, and the number of actions, respectively. The initial probabilities of all actions are equal. We define in the present work a Learning Automata (LA) for the removal methods (LA_{N^-}) and another one for the insertion methods (LA_{N^+}). A semi-greedy constructive heuristic is used to generate the initial solution. A multi-objective random variable neighborhood descent method (MO-RVND) is used to perform a local search. Algorithm 1 shows the pseudo-code of MO-ALNS.

The parameters a_1, a_2 and a_3 are rewards and the parameter b_1 is a penalty. The parameter q defines the number of jobs that are removed from and inserted in the current solution at each iteration of the algorithm. The parameter K defines the periodicity that the probabilities are updated. $t_{\max}$ is the maximum execution time and it is used as the stop criterion of the algorithm.

Initially, a solution s is constructed and a set of non-dominated solutions D is created. The set D is updated by means of the *addSolution* method, detailed in Ref. [53]. The initial temperature is calculated such that a current solution has 50% of chance of acceptance if it is 5% worse than the initial solution. It is considered the weighted sum of the objectives of the initial solution, in which each objective has a weight equal to 0.5.

The steps of the MO-ALNS at each iteration are:

1. A removal method $\alpha_i^- \in \alpha^-$ and an insertion method $\alpha_j^+ \in \alpha^+$ are selected by a roulette method, using the probabilities of those methods. The removal and insertion methods are the same used in the ALNS with Learning Automata;
2. q jobs are removed from the current solution with the α_i^- method, and re-inserted with the α_j^+ method. These two first steps are exactly the same of the ALNS with Learning Automata;
3. The multi-objective local search method MO-RVND is applied to the current solution s;
4. As in the ALNS with Learning Automata, the probabilities p^- and p^+ are updated. If the solution s' is accepted, Eq. (13) is used. Otherwise, if the solution s' is not accepted, Eq. (14) is applied. If the current solution s' dominates the solution s or if there is no dominance between them, an attempt to insert s' into D using *addSolution* method is performed. If the current solution is accepted according to the temperature criterion, a solution s'' receives the solution s';
5. A new solution s is randomly selected among all solutions of the set D and the solution s'';
6. As in the ALNS with the Learning Automata, at every K iterations the probability values are updated within each LA;
7. In the end, the set D of non-dominated solutions is returned.

A solution might be accepted in three different situations and for each of them a reward parameter (a_1, a_2 and a_3) is applied in the LA updating. These three situations are: *i*) a_1: if the solution s' dominates the solution s; *ii*) a_2: if there is no domination between solutions s' and s; *iii*) a_3: if the solution s' is dominated by the solution s, however it is accepted according to the temperature criterion. If the solution found is not accepted, the penalty parameter (b_1) is used to update the LA.

3.2.1. Constructive procedure

The constructive heuristic selects jobs to the machines and a random operation mode $l \in L$ for each allocation. The operation of this heuristic is given as follows. The heuristic has a time limit equals to 1% of the $t_{\max}$ (time limit for execution of the MO-ALNS). Initially, the method selects a random operation mode $l \in L$ for each allocation (vector of integers in Fig. 3). A new solution is generated at each iteration of the heuristic and only the best solution is stored. New solutions are generated by means of the semi-greedy constructive heuristic Adaptive Shortest Processing Time (ASPT) rule [54], which is guided by the *makespan*.

In the semi-greedy constructive heuristic, all jobs are inserted in a list of candidates LC. On each iteration, the insertion of each job $j \in LC$ at the end of each machine $i \in M$ is evaluated, considering the processing and setup times. A second list $best_{LC}$ stores the top 20% allocations that generated a lower cost for the corresponding machine. A job k is randomly selected in $best_{LC}$ and it is inserted into the corresponding machine. This process is repeated until all jobs in LC are allocated.

3.2.2. Insertion and removal heuristics

The removal and insertion methods used in the MO-ALNS are summarized in Tables 2 and 3. These heuristics use only the completion time in their evaluations. The removal methods are presented in Table 2. All methods remove q jobs. The insertion methods are shown in Table 3. The removed q jobs are shuffled at random before executing each insertion method.

3.2.3. Local search procedure

The multi-objective Random Variable Neighborhood Descent method is used as local search procedure. This algorithm is a multi-

Algorithm 1 MO-ALNS.

input : K, q, a_1, a_2, a_3, b_1
output: D
Define LA $(\phi^-, \alpha^-, \beta, A^-, \pi^-, p^-)$;
Define LA $(\phi^+, \alpha^+, \beta, A^+, \pi^+, p^+)$;
$s \leftarrow$ `constructive procedure` ();
$D \leftarrow$ `addSolution` (D,s,f) ; `/* Set of non-dominated solutions */`
$T \leftarrow (0.05 \times (\sum_{r=1}^{R} 0.5 \times f_r(s)))/\ln 2$;
while $currentTime \leq t_{\max}$ **do**
  Select a removal method α_i^- and an insertion method α_j^+ using roulette method;
  Remove q jobs from s using α_i^-;
  Insert the removed jobs on s using α_j^+;
  $s' \leftarrow$ `local search procedure` (s) ;
  if $(f(s') \prec f(s))$ **then**
    $D \leftarrow$ `addSolution` (D,s',f) ;
    $a \leftarrow a_1$; $b \leftarrow 0$;
    Update p^- and p^+ using Eq. (13)-(14);
  end
  else if $(f(s') \not\prec f(s)) \wedge (f(s) \not\prec f(s'))$ **then**
    $D \leftarrow$ `addSolution` (D,s',f) ;
    $a \leftarrow a_2$; $b \leftarrow 0$;
    Update p^- and p^+ using Eq. (13)-(14);
  end
  else if `random` $< \min\left\{1, \exp\left(\frac{\sum_{r=1}^{R} 0{,}5 \times f_r(s')}{T}\right)\right\}$ **then**
    $s'' \leftarrow s'$;
    $a \leftarrow a_3$; $b \leftarrow 0$;
    Update p^- and p^+ using Eq. (13)-(14);
  end
  else
    $a \leftarrow 0$; $b \leftarrow b_1$;
    Update p^- and p^+ using Eq. (13)-(14);
  end
  Random selection of s in (D, s'');
  if $k \mod K = 0$ **then**
    `updateLA`(A^-, A^+, p^-, p^+);
  end
  $k++$;
  $T \leftarrow 0.99 \times T$;
end
return D

objective version of the Random Variable Neighborhood Descent [55], in which the local searches are applied in random order. This method is more efficient than the classical Variable Neighborhood Descent [12], because the best order to apply the local searches may not only depend on the addressed problem, but also on the instances characteristics. Four neighborhood structures used in MO-RVND are described below.

1. **Multiple insertion – $NS^{MI}(s)$**: move of reallocating a job from a machine to another position in the same machine, or of reallocating a job in any position in another machine. Fig. 4a illustrates an example of this move, in which the job 4 on machine 2 is transferred to the second position on machine 1.
2. **Swap in the same machine – $NS^{SSM}(s)$**: move of swapping two jobs in the same machine. Fig. 4b illustrates an example of this move, in which the jobs 1 and 6 on machine 2 are swapped.
3. **Swap between different machines – $NS^{SDM}(s)$**: move of swapping two jobs from different machines. Fig. 4c illustrates an example of this move, in which the job 3 on machine 1 is swapped with job 1 on machine 2.
4. **Swap operation modes – $NS^{SMO}(s)$**: move of swapping operation mode of an allocation. Fig. 4d illustrates an example of this move, in which operation mode of job 3 is swapped from 1 to 3.

Four local searches are used to explore the search space using the neighborhood structures described previously. All local searches use the first improvement strategy. Three of those local searches were proposed in previous works, namely: *i)* FI_{MI} [9,14]: it uses the neighborhood structure NS^{MI}; *ii)* FI_{SDM} [10,14]: it uses the neighborhood structure NS^{SDM}; and *iii)* FI_{SSM} [14]: it uses the neighborhood structure NS^{SSM}. The only difference in this work is in the acceptance criterion. In the multi-objective version, a current solution s' is accepted if it dominates the solution s or if there is no dominance between them. That rule is also applied to the acceptance criteria of MO-RVND.

The fourth local search is called FI_{SMO}. It uses the neighborhood structure NS^{SMO}. The pseudo-code of this local search is given in Algorithm 2. Initially, all machines are sorted in descending order by the completion time. The machines are selected starting from longest to shortest completion time. For each machine, all changes between operation modes of its jobs are evaluated. The multi-objective evaluation

Table 2
Removal methods.

	Methods	Operation
1	Random removal	Removes jobs randomly.
2	Greedy expensive cost removal	Removes most expensive jobs considering the setup and processing time.
3	Semi-greedy expensive cost removal	It is a semi-greedy version of the previous method. At each iteration randomly removes one job from the 20% more expensive.
4	Random machine removal	Removes jobs from a randomly selected machine.
5	Highest cost machine removal	Removes jobs from machine with the highest cost.
6	Shaw removal	Removed jobs are considered similar. The similarity criterion is the sum of processing cost and setup time of allocated jobs.

Table 3
Insertion methods.

	Methods	Operation
1	Greedy insertion	Insert the jobs in the best position considering all positions of all machines.
2	Semi-greedy insertion	It is a semi-greedy version of the previous method. At each iteration, it inserts a job in a random position between the 20% best positions.
3	Lambda insertion	Insert 50% of the jobs using greedy insertion method and the other 50% of the jobs are randomly inserted.
4	ILS insertion	For each job a random machine is selected and the job is inserted in the best position of this machine.
5	Regretting insertion	Insert first the jobs with larger regretting cost. This cost is the difference between the best position and the second best solution considering all machines.
6	Hungarian insertion	Creates a square matrix where the lines are machines $m \in M$ and the columns are m first jobs of q. The data of the matrix are the best cost of allocation of each job on each machine. The Hungarian method is used to found the best allocation for each matrix. This process is repeated until all jobs of q are inserted.

function was detailed in Eqs: (1) and (2). If a neighbor $s' \in NS^{SSM}$ dominates the solution s or if there is no dominance between them, then the solution s' is accepted and the search ends; otherwise, it continues until all machines are analyzed.

3.2.4. Method for updating non-dominated solutions

The *addSolution* method is used to update the set of non-dominated solutions (D) [53]. This method uses simple Pareto dominance to evaluate solutions. The pseudo-code of this method is presented in Algorithm 3.

3.3. MO-ALNS/D algorithm

A limitation of the MO-ALNS algorithm (presented in the previous subsection) is that there is no control of diversification in the Pareto front approximation. As it can be verified in the line (30) of Algorithm 1, it just selects a random solution between solutions of the set D and the solution s''. Nonetheless, a distinguishing mark of the MOEA/D algo-

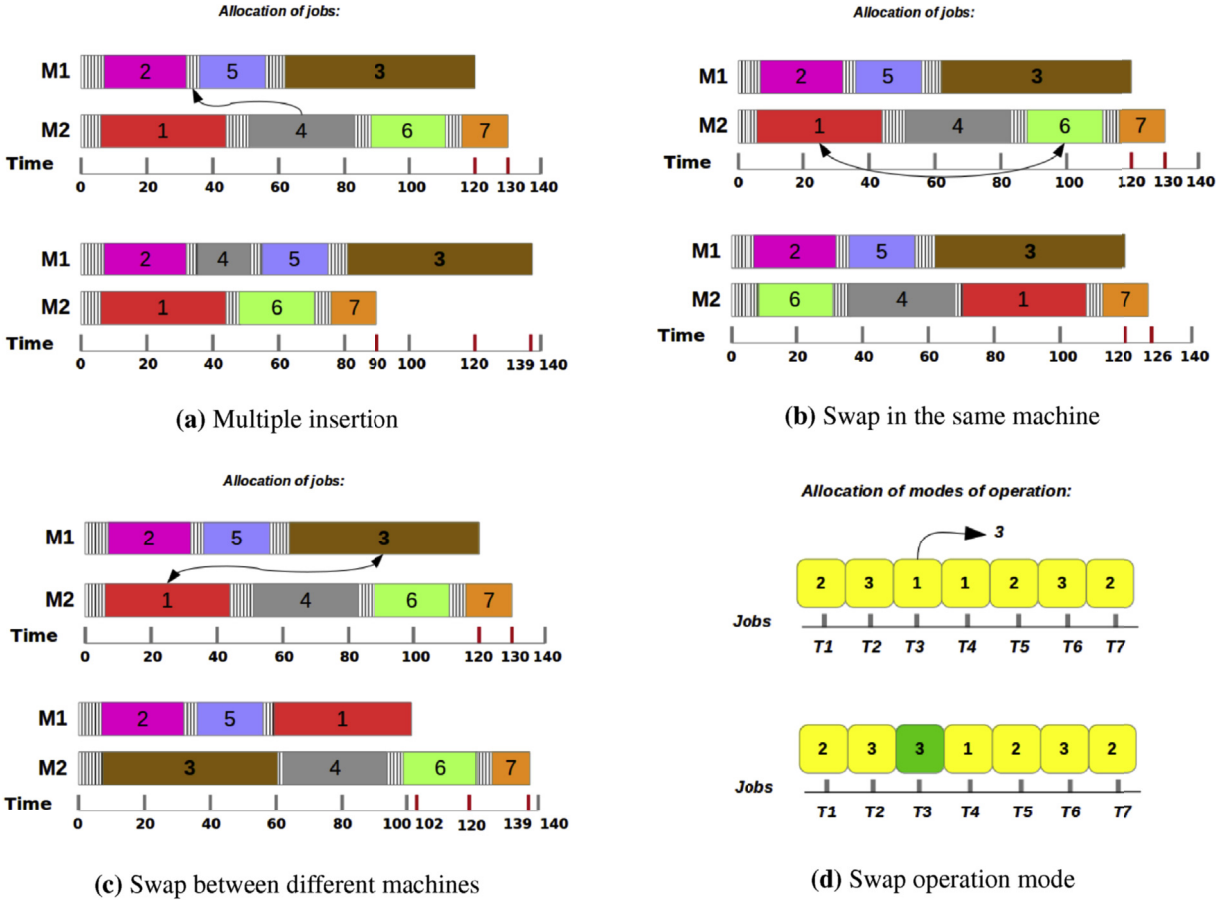

Fig. 4. Examples of movements.

Algorithm 2 Local search $FI_{SMO,}$

Input: s
Output: s
Let M the set of machines;
Let L the set of operation modes in random order;
Let K_1 the set M sorted in descending order by completion times;
foreach *machine* $k_1 \in K_1$ **do**
  foreach *machine* $j \in K_1$ **do**
    foreach *operation mode* $l_1 \in L$ **do**
      Let s' the result of changing the operation mode of j to l_1;
      if $f(s') \prec f(s) \vee (f(s') \not\prec f(s) \wedge f(s) \not\prec f(s'))$ **then**
        $s \leftarrow s'$;
        Break;
      end
    end
  end
end
Return s;

Algorithm 3 addSolution.

Input: Set of non-dominated solutions D; Solution s, and its evaluations $z(s)$
Output: D

Added $\leftarrow$ true ;
forall the $x \in D$ **do**
  if $z(x) \preceq z(s)$ **then**
    Added $\leftarrow$ false ;
    Break ;
  end
  if $z(s) \prec z(x)$ **then**
    $D \leftarrow D \setminus x$;
  end
end
if *Added* = *true* **then**
  $D \leftarrow D \cup s$;
end
return D ;

rithm is the sectorization of search in the Pareto front approximation. It divides the multi-objective problem into S mono-objective problems based on aggregations by means of different weight vectors.

The proposed algorithm here is called MO-ALNS/D. This algorithm has the structure of the classical MOEA/D, but it uses the ALNS with Learning Automata to solve the scalar subproblems. The pseudo-code of the MO-ALNS/D is shown in Algorithm 4.

The input parameters of Algorithm 4 can be divided into two groups: MO-ALNS/D parameters and ALNS with Learning Automata parameters. The parameters of the MO-ALNS/D are: *i*) S: total of subproblems generated by decomposition; *ii*) T: number of neighbors of each subproblem; and *iii*) G: maximum number of generations. The parameters of ALNS are the remaining $(t_{max}, K, q, a_1, a_2, a_3, b_1)$, and are the same described in Section 3.2.

A set of weights w for all subproblems is constructed using the Scheffé's method [56]. This method is also used in the classical MOEA/D. This method was proposed on experiments with mixtures and is defined by proposition $\{r, w_{max}\}$-simplex lattice. r is the number of objectives in optimization problems. In the Scheffé's method, $\binom{r+w_{max}-1}{w_{max}}$ points in r-dimensional space are generated, with $w_{max}+1$ points equally spaced in the boundary of the simplex and satisfying the condition: $\| \mathbf{w} \|_1 = w_1 + w_2 + \cdots + w_r = 1$.

After that, T nearest neighbors for each subproblem i are stored in the set B_i. The constructive procedure described in Section 3.2.1 is used to generate individuals from the population (*Pop*). Each individual is associated with a subproblem $i \in S$. After generation of the population, a vector z^* is constructed with the best solution for each objective.

The steps of the algorithm at each iteration are:

1. A neighbor v of subproblem i is randomly selected from B_i;
2. The neighbor v is optimized using the mono-objective ALNS with Learning Automata algorithm. An initial solution is not generated in this mono-objective version, because the initial solution (Pop_v) is passed as a parameter by the MO-ALNS/D. The evaluation function used in the ALNS with Learning Automata is replaced for the Tchebycheff Approach (TCH) aggregation function. This function is applied to the objective functions *makespan* and total of energy consumption;
3. Differently from MO-ALNS, single-objective local search is used in the RVND method, considering the aggregation method.
4. The vector of best solutions z^* is updated;

5. For each neighbor $j \in B_i$, it is verified if solution sol_v is better than the corresponding solution Pop_j using the TCH aggregation function. If it is, Pop_j receives the solution sol_v;
6. These steps are repeated until the number of generations (G) is exhausted. Next, the non-dominated solutions *Pop* are stored in the set D, using the *addSolution* method (described in Section 3.2.4). At the end, the set D is returned.

The removal and insertion methods described in Section 3.2.2 and the four local searches described in Section 3.2.3 are used by ALNS with Learning Automata. In local searches the only difference is about the criterion of acceptance, in which the evaluation is done by the aggregation function TCH.

3.3.1. Tchebycheff Approach aggregation function

The Tchebycheff aggregation function is used in the classical MOEA/D. The i-th subproblem is defined as follows [57]:

$$\text{minimize } g^{te}(x \mid w_i, z^*) = \max_{1 \le r \le R} \{w_i^r | f_r(x) - z_r^*|\} \quad (15)$$

in which $z^* = (z_1^*, \ldots, z_R^*)^T$ is the ideal reference point with:

$$z_r^* \le \min\{f_r(x) \mid x \in \Omega\} for r = 1, 2, \ldots, R. \quad (16)$$

4. Computational experiments

The computational experiments were performed in a computer with Intel Core i7, 1.9 GHz processor, 6 GB RAM and Ubuntu 16.04 operating system. The algorithms have been implemented in the JAVA language with Netbeans IDE 8.0.2.

4.1. Analysis of parameters of the MO-ALNS and MO-ALNS/D algorithms

In this subsection the values used for each parameter of the MO-ALNS and MO-ALNS/D algorithms are presented. The values are shown in Table 4.

The parameters were calibrated with empirical tests using five instances, one from each group of jobs. The calibration was performed using one factor at a time. For each parameter, a set of values was tested by fixing the others. The stopping criterion used for the MO-ALNS/D algorithm is a total of 50 generations ($G = 50$). This value was generated by empirical tests using a range from 20 to 200. The value 50 was the minimum number of generations for convergence of the algorithm. As the stopping criterion of the MO-ALNS algorithm is the run time ($t_{\max}$), we used $t_{\max}$ equal to the time spent by MO-ALNS/D. The parameter K defines the periodicity that the probabilities of the Learning Automata are updated. α^- is the set of removal methods and α^+ is the set of insertion methods. The Learning Automata needs a minimum number of iterations with the environment to learning. The value 6 was chosen in a range from 2 to 20 after several empirical tests.

The parameter q defines the number of jobs that are removed from the current solution and then re-inserted. The value of this parameter was defined by empirical tests using all instances and variations from 4% to 30%. Rewards and penalty values are used to update the Learning Automata according to the results obtained. These values were based on discussions and tuning in Refs. [14,52]. The parameter S is the total number of subproblems generated in MO-ALNS/D algorithm. This parameter defines the granularity of weights in the search. The value of this parameter was defined by empirical tests using a range from 20 to 200. The parameter T is the number of neighbors of each subproblem. This parameter is commonly used with the value equal to 2 in MOEA/D implementations. The parameter $t_{\max}^{ALNS}$ defines the maximum execution time of the single objective ALNS in the MO-ALNS/D algorithm, where n is the number of jobs (problem size). The value of this parameter was defined by empirical tests using a range of $10 \times n$ (ms) to $60 \times n$ (ms). The purpose of these tests was to identify a minimum time for the ALNS to satisfactorily solve each subproblem.

Table 4
Values of hyper-parameters for the MO-ALNS and MO-ALNS/D algorithms.

Parameters	Value in MO-ALNS	Value in MO-ALNS/D
Stopping criterion - Maximum execution time ($t_{\max}$)	Time spent by MO-ALNS/D	–
Stopping criterion - Number of generations (G)	–	50
Periodicity that the probabilities are updated (K):	$6 \times \max(\lvert\alpha^-\rvert, \lvert\alpha^+\rvert)$	$6 \times \max(\lvert\alpha^-\rvert, \lvert\alpha^+\rvert)$
Number of jobs that are removed and then re-inserted (q)	5% of the jobs	5% of the jobs
First reward (a_1)	0.2	0.2
Second reward (a_2)	0.1	0.1
Third reward (a_3)	0.05	0.05
Penalty (b_1)	0.02	0.02
Total of subproblems (S)	–	50
Number of neighbors (T)	–	2
Maximum run time of single-objective ALNS ($t_{\max}^{ALNS}$)	–	$20 \times n$ (ms)

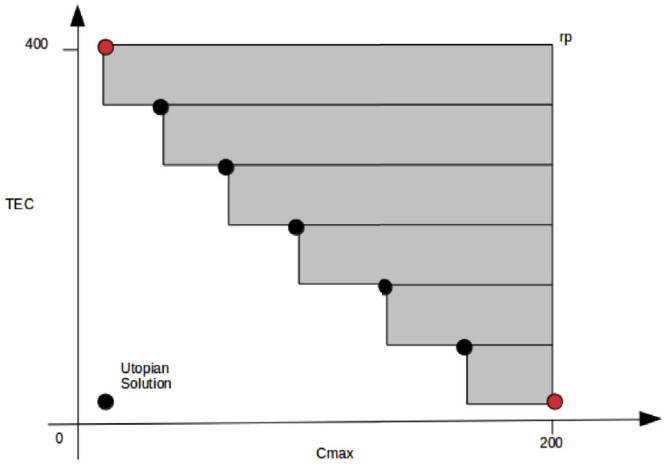

Fig. 5. Example of the hypervolume indicator.

4.2. Algorithms validation

In this section, MO-ALNS and MO-ALNS/D algorithms are executed for the small and medium-sized instances proposed in Ref. [45] (80 instances in total). The results of the algorithms are compared to the values obtained by the MILP model following the ϵ-constrained method, which is available in Ref. [45]. The main goal of this section is to validate the results of MO-ALNS and MO-ALNS/D algorithms for the addressed problem.

The hypervolume (HV) [58] indicator is used to compare the results of the algorithms and the ϵ-constrained method. The hypervolume of an estimated Pareto front is the sum of the hypercubes that each set of solutions contains. Fig. 5 illustrates an example of the hypervolume for an instance in a minimization problem and a reference point rp. The red dots are the upper limits, the reference point is $rp = (200, 400)$, and the black dots are the non-dominated solutions found.

In this section, the reference point rp of the hypervolume used is given by the limits generated in Ref. [45]. These limits were generated by using the ϵ-constrained method to find the extreme points of the Pareto front. One extreme point corresponds to the minimization of only the makespan and the other extreme point corresponds to the minimization of only the total energy consumption. The hypervolume values

Algorithm 4 MO-ALNS/D.

input : $S, T, G, t_{\max}^{ALNS}, K, q, a_1, a_2, a_3, b_1$
output: D
$w \leftarrow$ Scheffé method(S);
$B \leftarrow$ neighbors generation(T, w, S);
$Pop \leftarrow \emptyset$;
for $i = 1$ *until* S **do**
  $Pop_i \leftarrow$ constructive procedure ();
end
$z^* \leftarrow \emptyset$;
$z^* \leftarrow$ updateZ(Pop, z^*);
while $G \geq 0$ **do**
  for $i = 1$ *until* S **do**
    Randomly select one neighbor v from B_i;
    $sol_v \leftarrow$ ALNS$(Pop_v, t_{\max}^{ALNS}, K, q, a_1, a_2, a_3, b_1)$;
    updateZ(sol_v, z^*);
    forall the *neighbor* $j \in B_i$ **do**
      if $g^{te}(sol_v|w_j, z^*) \leq g^{te}(Pop_j|w_j, z^*)$ **then**
        $Pop_j \leftarrow sol_v$;
      end
    end
  end
end
for $i = 1$ *until* S **do**
  $D \leftarrow$ addSolution (D, Pop_i, f);
end
return D

were normalized between 0 and 1, by dividing the sum of the hypercubes by the maximum area containing the Pareto front, as described in (17).

$$\frac{\sum_{i=1}^{P} hy_i}{(f_b^{C_{\max}} - f_a^{C_{\max}}) \times (f_b^{TEC} - f_a^{TEC})} \tag{17}$$

$f_a^{C_{\max}}$ and $f_b^{C_{\max}}$ are respectively the minimum and maximum values of $C_{\max}$ in the Pareto front identified by the ϵ-constrained method. Likewise, f_a^{TEC} and f_b^{TEC} are respectively the minimum and maximum values of TEC (total energy consumption) in the Pareto front found by the ϵ-constrained method.

Due to the stochastic nature of MO-ALNS and MO-ALNS/D algorithms, they were executed five times for each instance ($5 \times 80 = 400$ samples in total) and only the mean value of the hypervolume was considered. The results of the ϵ-constrained method[1] were extracted from Ref. [45].

Table 5 presents the results of the algorithms and the ϵ-constrained method. The values are grouped by instances with the same number of jobs.

The best average results of each group of instances are highlighted in blue with the standard deviation presented between parentheses. Considering the computational experiment, the results suggest a good performance of the proposed algorithms, and also suggest that both methods have a good convergence towards the true Pareto front. In the results with the ϵ-constrained method using MILP, only ten values of ϵ were used, which explains why in some cases the average results achieved by the heuristic methods are higher. We performed a statistical test of Analysis of Variance (ANOVA) [59],[2] with 95% confidence (threshold = 0.05) to verify if there is statistical difference between the results of the HV indicator for MO-ALNS and MO-ALNS/D algorithms. The ANOVA test was applied with blocking for the groups of instances with the same number of jobs. The test found a p-value equal to 0.1479, this result suggests there is no statistical difference between the results of the HV indicator. This means that both algorithms achieved convergence to the Pareto-optimal front in small and medium-sized instances, hence validating them for problem. A graphical analysis of the algorithms convergence was performed. Two instances were randomly selected to illustrate the results. Figs. 6 and 7 show the results.

Figs. 6 and 7 also suggest that the proposed algorithms have a good

Table 5
Results of the HV indicator with small and medium-sized instances.

Set of instances	ϵ-constrained	MO-ALNS	MO-ALNS/D
6	0.708 (± 0.086)	0.817 (± 0.021)	0.755 (± 0.046)
8	0.768 (± 0.093)	0.822 (± 0.113)	0.791 (± 0.139)
10	0.791 (± 0.055)	0.825 (± 0.008)	0.810 (± 0.013)
12	0.863 (± 0.093)	0.841 (± 0.129)	0.831 (± 0.131)
15	0.833 (± 0.088)	0.799 (± 0.052)	0.832 (± 0.133)

[1] In Ref. [45] the computer used in the experiments with the ϵ-constrained method is the same one used in this work.

[2] The null-hypothesis (H_0) is that the average results of the metric are equal. The alternative hypothesis (H_1) is that the average results of the metric are different, that is, at least one of the results is different from the others. We verified that the data follow a normal distribution using the Shapiro-Wilk [60] test. The null-hypothesis (H_0) of this test is that the population is normally distributed. Alternative hypothesis (H_1), the population is not normally distributed.

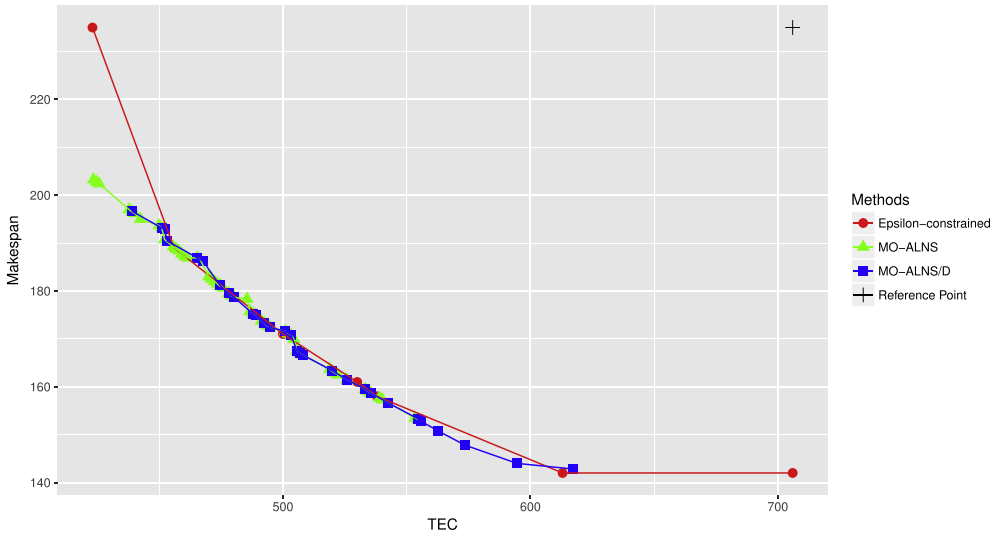

Fig. 6. Pareto front obtained with the two algorithms proposed and the ϵ-constrained method for an instance with 8 jobs and 2 machines.

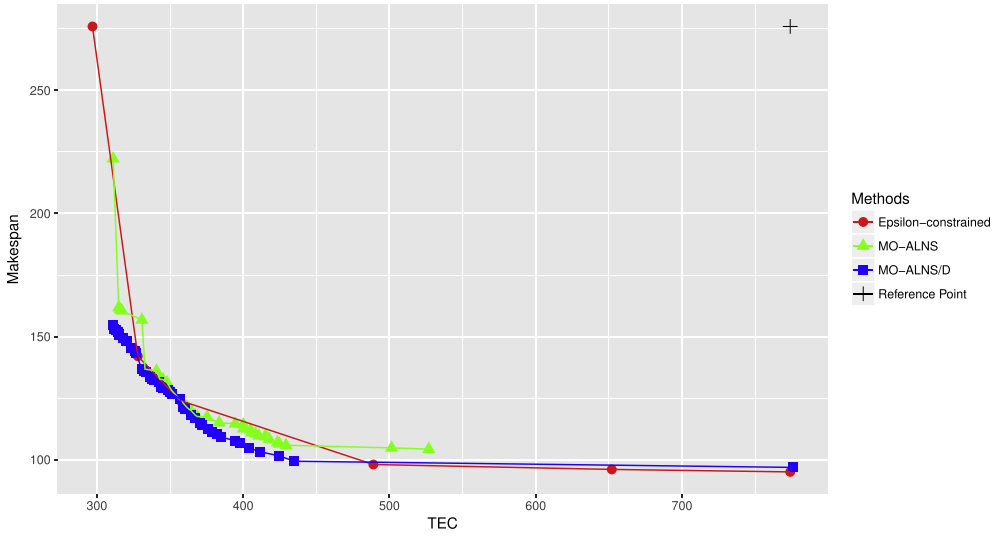

Fig. 7. Pareto front obtained with the two algorithms proposed and the ϵ-constrained method for an instance with 12 jobs and 4 machines.

Table 6
Characteristics of large size instances.

Parameters	Levels	Based on
Number of jobs (n):	50, 100, 150, 200, 250	[8]
Number of machines (m):	10, 20, 30	[8]
Number of operation modes (o):	5	[2,61]
Processing time (p_{ij}):	$U[1, 99]$	[8]
Sequence dependent setup time (S_{ijk}):	U [1,9], $U[1, 124]$	[8]
Machine power (π_i):	$U[40, 200]$	–
Multiplying factor of speed (v_l):	1.2, 1.1, 1, 0.9, 0, 8	[2,61]
Multiplying factor of power (λ_l):	1.5, 1.25, 1, 0.8, 0.6	[2,61]

convergence towards the Pareto front, considering the experimental environment.

4.3. Large instances generation

In this paper we propose an additional set of large scale instances. This set has 30 instances from combinations of 50, 100, 150, 200 and 250 jobs with 10, 20 and 30 machines. All combinations have five operation modes ($o = 5$). Table 6 presents the characteristics of instances.

The processing times and the setup times were generated using the uniform distributions proposed by Ref. [8] for a similar problem. The set of instances of [8] is often used in the literature. The power consumption of the machines were generated by uniform distribution between 40 and 200 kW. Five modes of operation were used representing five speeds: *i*) very slow; *ii*) slow; *iii*) normal; *iv*) fast; and *v*) very fast.

4.4. Comparison between MO-ALNS with and without LA

In this subsection we performed the comparison between the MO-ALNS and MO-LNS algorithms. The MO-LNS is a multi-objective version of the Large Neighborhood Search [62] method. Basically, MO-LNS uses: the Random machine removal to remove jobs and the Greedy insertion to insert the jobs. These removal and insertion methods are described in Subsection 3.2.2. The local search procedure in Subsection 3.2.3 is used to perform the local searches in both algorithms. The goal of this experiment is to verify if the adaptation and learning done by LA is relevant to the performance of MO-ALNS.

The comparison is performed using the large instances. The stopping criterion of the MO-LNS algorithm is the same used with the MO-ALNS.

Table 7
Results of the HV indicator for MO-ALNS and MO-LNS.

N of jobs (n)	N of machines (m)	Setup time (S_{ijk})	MO-ALNS	MO-LNS
50	10	$U[1,9]$	0.676	0.630
		$U[1,124]$	0.776	0.692
	20	$U[1,9]$	0.752	0.529
		$U[1,124]$	0.753	0.664
	30	$U[1,9]$	0.754	0.451
		$U[1,124]$	0.804	0.526
100	10	$U[1,9]$	0.542	0.537
		$U[1,124]$	0.757	0.615
	20	$U[1,9]$	0.687	0.523
		$U[1,124]$	0.707	0.421
	30	$U[1,9]$	0.770	0.429
		$U[1,124]$	0.621	0.414
150	10	$U[1,9]$	0.792	0.509
		$U[1,124]$	0.688	0.322
	20	$U[1,9]$	0.732	0.542
		$U[1,124]$	0.873	0.504
	30	$U[1,9]$	0.585	0.343
		$U[1,124]$	0.813	0.658
200	10	$U[1,9]$	0.753	0.402
		$U[1,124]$	0.759	0.538
	20	$U[1,9]$	0.711	0.559
		$U[1,124]$	0.717	0.518
	30	$U[1,9]$	0.627	0.627
		$U[1,124]$	0.780	0.667
250	10	$U[1,9]$	0.747	0.473
		$U[1,124]$	0.507	0.503
	20	$U[1,9]$	0.822	0.404
		$U[1,124]$	0.695	0.391
	30	$U[1,9]$	0.614	0.799
		$U[1,124]$	0.448	0.373

The indicator used to compare the results of the two algorithms is the HV and the reference point is the largest solution found for each of the objectives. All solutions found by the two algorithms (MO-ALNS and MO-LNS) were used to identify the minimum and maximum values of each objective in the estimated Pareto fronts. The values of the HV indicator also were normalized between 0 and 1, as in Subsection 4.2.

As the algorithms have stochastic nature, they were executed five times for each instance ($5 \times 30 = 150$ samples in total) and only the average result was considered. Table 7 presents the average results of the algorithms for the HV indicator. The values are grouped by instances with the same number of jobs.

The best average results in Table 7 are highlighted in blue. The MO-ALNS algorithm found the best result in most instances. Considering the complete results, the MO-ALNS achieved the best performance in 93% of cases. Fig. 8 shows the box plot of these results.

The box plot graph also suggests a better performance of MO-ALNS algorithm. To check if there is statistical difference between the results of HV indicator for MO-ALNS and MO-LNS algorithms, the ANOVA statistical test[3] with 95% confidence (threshold = 0.05) was applied. The ANOVA test was applied with blocking for the groups of instances with the same number of jobs. The p-value found is equal to 2.25×10^{-9}. This suggests there is statistical difference between the results of HV indicator.

[3] We verified that the data follows a normal distribution using the Shapiro-Wilk test.

The results of this section show that the MO-ALNS algorithm is more efficient than the MO-LNS. Therefore, the inclusion of removal/insertion operators and the Learning Automata in MO-ALNS do affect performance of the method positively. Therefore, in the next subsection MO-ALNS algorithm is used in the comparison with the MO-ALNS/D algorithm and with the classic MOEA/D [38].

4.5. Comparison of MO-ALNS/D, MO-ALNS and MOEA/D results

The MO-ALNS, MO-ALNS/D and the MOEA/D algorithms are evaluated with large instances (Table 6) in this subsection. Regarding the MOEA/D, the evaluations were conducted using the PlatEMO [63] with default parameters and suitable operators for the green machine scheduling problem shown in this work. For the batch of tests conducted on PlatEMO, each solution is represented by a vector of size $2n + m$ composed by elements that indicate tasks, modes of operations and marks (non-repeating negative integers) which denote the end of a sequence of tasks scheduled in a machine. Clearly, the number of marks depends on the number of machines. Let $\boldsymbol{Y} = [Y_1, Y_2, \ldots, Y_{2n+m}]$ be such a vector. The elements from Y_1 to Y_{n+m-1} denote tasks and marks and the elements from Y_{n+m+1} to Y_{2n+m} represents the operation modes of the n tasks. The element Y_{n+m} contains a fixed value equal to $-m$. For example, the solution representation illustrated in Fig. 3, must have the following format:

$$[7, 3, 4, -\boldsymbol{1}, 2, 1, 6, 5, -\boldsymbol{2}, 2, 3, 1, 1, 2, 3, 2].$$

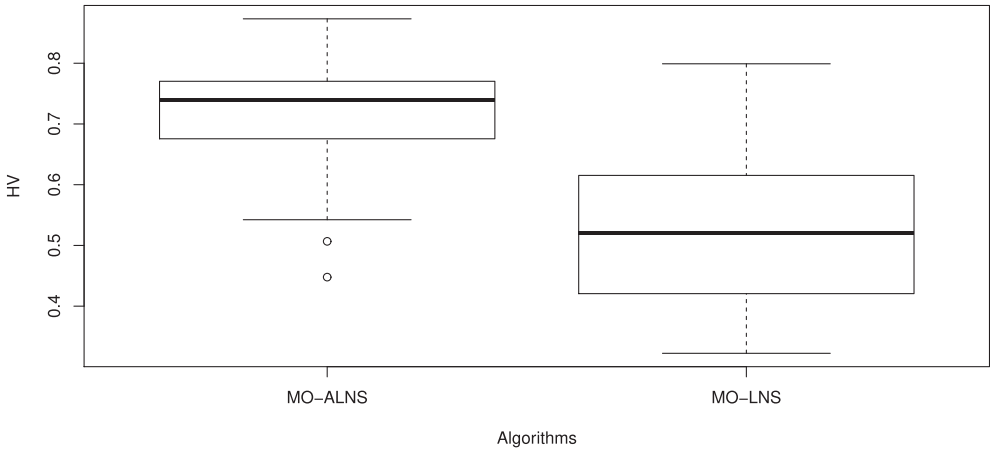

Fig. 8. Box plot results of the MO-ALNS and MO-LNS algorithms for the HV indicator.

Table 8
Results of the HV indicator for the MO-ALNS, MO-ALNS/D and MOEA/D algorithms.

N of jobs	N of machines	Setup time	MO-ALNS	MO-ALNS/D	MOEA/D
50	10	$U[1,9]$	0.861	0.947	0.45
		$U[1,124]$	0.844	0.94	0.318
	20	$U[1,9]$	0.847	0.968	0.404
		$U[1,124]$	0.835	0.968	0.491
	30	$U[1,9]$	0.935	0.975	0.355
		$U[1,124]$	0.87	0.92	0.564
100	10	$U[1,9]$	0.813	0.961	0.292
		$U[1,124]$	0.895	0.969	0.372
	20	$U[1,9]$	0.889	0.968	0.246
		$U[1,124]$	0.919	0.979	0.183
	30	$U[1,9]$	0.964	0.982	0.217
		$U[1,124]$	0.776	0.934	0.239
150	10	$U[1,9]$	0.957	0.977	0.335
		$U[1,124]$	0.941	0.978	0.305
	20	$U[1,9]$	0.958	0.984	0.163
		$U[1,124]$	0.94	0.976	0.2
	30	$U[1,9]$	0.947	0.984	0.078
		$U[1,124]$	0.716	0.982	0.142
200	10	$U[1,9]$	0.901	0.975	0.134
		$U[1,124]$	0.929	0.974	0.192
	20	$U[1,9]$	0.958	0.98	0.037
		$U[1,124]$	0.951	0.98	0.101
	30	$U[1,9]$	0.954	0.989	0.057
		$U[1,124]$	0.694	0.976	0.172
250	10	$U[1,9]$	0.929	0.982	0.083
		$U[1,124]$	0.751	0.985	0.279
	20	$U[1,9]$	0.963	0.982	0.031
		$U[1,124]$	0.844	0.944	0.162
	30	$U[1,9]$	0.937	0.987	0.019
		$U[1,124]$	0.759	0.953	0.175

Taking into account this representation scheme, we created operators of crossover and mutation for the MOEA/D of PlatEMO. Regarding the crossover, it was partially based on the crossover operator for single machine schedule presented in Ref. [64]. For the green machine schedule problem, such operator was divided into two parts. The first addresses the items from Y_1 to Y_{n+m-1}. It defines how we must schedule the tasks in the resulting solution (offspring). The second deals with the operation modes of this solution, from Y_{n+m+1} to Y_{2n+m}.

The first part is defined using the same approach presented by Ref. [64], in which, a randomly generated bit string template determines for each parent which elements are carried forward into their offspring. The main difference is the existence of marks (non-repeating negative integers) that denote the end of a sequence of tasks scheduled in a machine. In summary, given parent-1 and parent-2, $\boldsymbol{Y}$ from Y_1 to

Table 9
Results of CS indicator for the MO-ALNS, MO-ALNS/D and MOEA/D algorithms.

n	m	S_{ijk}	MO-ALNS × MO-ALNS/D	MO-ALNS/D × MO-ALNS	MO-ALNS × MOEA/D	MOEA/D × MO-ALNS	MO-ALNS/D × MOEA/D	MOEA/D × MO-ALNS/D
50	10	$U[1,9]$	0.014	0.357	0.950	0.000	1.000	1.000
		$U[1,124]$	0.025	0.257	1.000	0.000	1.000	0.000
	20	$U[1,9]$	0.051	0.324	1.000	0.000	1.000	0.000
		$U[1,124]$	0.021	0.572	1.000	0.000	1.000	0.000
	30	$U[1,9]$	0.081	0.368	1.000	0.000	1.000	0.000
		$U[1,124]$	0.105	0.297	0.600	0.000	0.975	0.000
100	10	$U[1,9]$	0.021	0.395	1.000	0.000	1.000	0.000
		$U[1,124]$	0.035	0.377	1.000	0.000	1.000	0.000
	20	$U[1,9]$	0.056	0.235	1.000	0.000	1.000	0.000
		$U[1,124]$	0.047	0.170	1.000	0.000	1.000	0.000
	30	$U[1,9]$	0.228	0.182	1.000	0.000	1.000	0.000
		$U[1,124]$	0.120	0.244	1.000	0.000	1.000	0.000
150	10	$U[1,9]$	0.050	0.298	1.000	0.000	1.000	0.000
		$U[1,124]$	0.038	0.305	1.000	0.000	0.970	0.000
	20	$U[1,9]$	0.104	0.213	1.000	0.000	0.988	0.000
		$U[1,124]$	0.082	0.098	1.000	0.000	0.983	0.000
	30	$U[1,9]$	0.088	0.061	1.000	0.000	1.000	0.000
		$U[1,124]$	0.000	0.188	1.000	0.000	0, 643	0.000
200	10	$U[1,9]$	0.047	0.189	1.000	0.000	1.000	0.000
		$U[1,124]$	0.067	0.164	1.000	0.000	0.940	0.000
	20	$U[1,9]$	0.105	0.105	1.000	0.000	0.850	0.000
		$U[1,124]$	0.089	0.054	1.000	0.000	1.000	0.000
	30	$U[1,9]$	0.174	0.140	1.000	0.000	0.891	0.000
		$U[1,124]$	0.000	0.432	1.000	0.000	0.960	0.000
250	10	$U[1,9]$	0.065	0.195	1.000	0.000	1.000	0.000
		$U[1,124]$	0.000	0.173	1.000	0.000	0.771	0.000
	20	$U[1,9]$	0.190	0.055	1.000	0.000	0.916	0.000
		$U[1,124]$	0.229	0.269	1.000	0.000	0.895	0.000
	30	$U[1,9]$	0.209	0.104	1.000	0.000	0.933	0.000
		$U[1,124]$	0.200	0.284	0.960	0.000	0.853	0.000

Y_{n+m-1} is defined using the [64] method. The second part is defined such a way that the segment of $\boldsymbol{Y}$ that represents the operation modes can get information from both parents. Thus, a randomly integer R from $n + m + 1$ to $2n + m$ is generated. As a result, the offspring from Y_{n+m+1} to Y_R contains operation modes of parent-1, while from Y_{R+1} to Y_{2n+m} includes operation modes of parent-2. Regarding the mutation operator, the four examples of movements illustrated in Fig. 4a–d were randomly (uniform) selected and applied on the offspring.

The HV and coverage between two sets (CS) [58] indicators are used to compare the convergence of the algorithms. The unary indicators (as the hypervolume) have a limited efficiency because the sets of non-dominated solutions are analyzed separately [65,66]. Therefore, to compare the results of the algorithms, the coverage between two sets (CS) binary indicator is also used. The CS indicator determines the percentage of solutions from one set (X'') that a given set (X') dominates. Eq. (18) shows the calculation of coverage indicator between sets.

$$CS(X', X'') = \frac{|a'' \in X''; \exists a' \in X' : a' \quad \text{cover} \quad a''|}{|X''|} \tag{18}$$

The operation a'covera'' determines that a' dominates a'' or a' equals to a''. The results of CS indicator are mapped to [0,1]. If the result of CS is equal to 1, this indicates that all points of X'' are dominated or equal to points in X'.

The reference point used in the HV indicator is the largest solution found for each of the objectives, considering all the solutions found by the MO-LNS, MO-ALNS and MO-ALNS/D algorithms. The values of the HV indicator were normalized between 0 and 1, as in the validation phase (Subsection 4.2). As the algorithms have stochastic nature, they were executed five times for each instance and only the average results are considered.

Tables 8 and 9 present the average results obtained for the HV and CS indicators.

The best results of Tables 8 and 9 are highlighted in blue. The results suggest that the MO-ALNS/D algorithm achieved better results in most cases for the HV indicator. Considering the complete results the MO-ALNS/D achieved the best performance in 70% of cases. The MO-ALNS/D algorithm also found better results for the CS indicator. Analyzing the complete results the MO-ALNS/D achieved the best performance in 76% of cases. It can be observed that the MO-ALNS/D solutions cover some parts of the Pareto front which are not covered by the MO-ALNS solutions. Figs. 9 and 10 present the box plot of the results.

These graphs also suggest a better performance of the MO-ALNS/D algorithm for the HV and CS indicators. A statistical test was applied to verify if there is statistical difference between the results of HV indicator. It was applied an ANOVA test[4] with 95% confidence (threshold = 0.05). The ANOVA test was applied with blocking for the groups of instances with the same number of jobs. The value obtained for the p-value is equal to 0.000344. As the p-value is smaller than the threshold, the results suggest there is statistical difference between the results of the hypervolume.

[4] We verified that the data follow a normal distribution using the Shapiro-Wilk test.

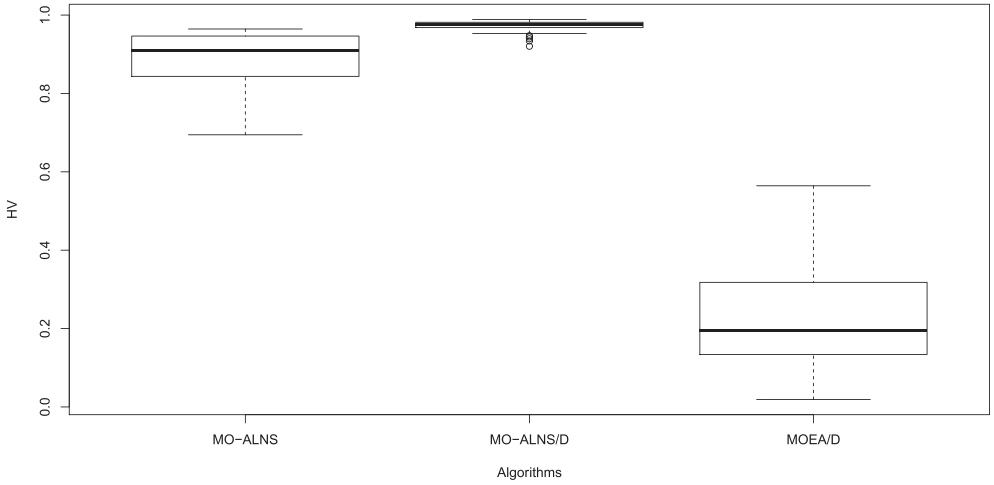

Fig. 9. Box plot results of the MO-ALNS, MO-ALNS/D and MOEA/D algorithms for the HV indicator.

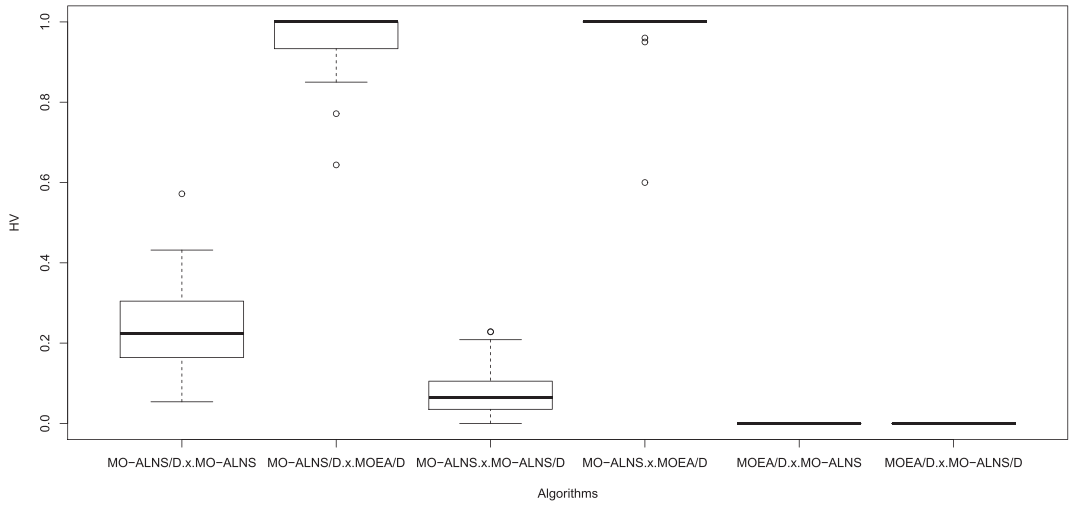

Fig. 10. Box plot results of the MO-ALNS, MO-ALNS/D and MOEA/D algorithms for the CS indicator.

To identify a significant difference between the results of paired algorithms, the Tukey HSD test [59] was done with 95% of confidence level (threshold = 0.05). Fig. 11 shows the result of this test.

The result of Tukey HSD test suggests that MO-ALNS/D is statistically better than the other algorithms in a pairwise comparison. This result also suggests the efficiency of the proposed algorithm under the defined conditions of the experiment.

A third indicator also is used to evaluate the quality of the Pareto front approximations obtained by the algorithms. This indicator is the hierarchical cluster counting (HCC) [67]. The HCC is able to measure both the uniformity and extension of the estimate set. In this indicator, the non-dominated solutions (or points) of an estimate set are sequentially grouped into clusters. The agglomerative clustering procedures start with each point being a cluster. In the next iterations the nearest clusters are joined together until all the data is grouped in only one class. The value of HCC is given by the integration (summation) of fusion distances used at each iteration of the hierarchical agglomerative clustering process. The estimate set that has the higher value for the HCC indicator has the better description of the Pareto front. The results of the algorithms for the HCC indicator are presented in Table 10.

The best average results of Table 10 are highlighted in blue. The MO-ALNS/D algorithm achieved better results in almost all cases for the HCC indicator. A statistical test was applied to identify if there is statistical difference between the average results of the HCC indicator. Kruskal-Wallis Test [59] non-parametric test[5] with 95% confidence (threshold = 0.05) was used. The result found is p-value equal to 0.005764. This value suggests that there is statistical difference between the average results of the HCC. The Tukey test was applied with 95% of confidence level (threshold = 0.05) to verify the difference between the pairwise differences. The results of the test are shown in Fig. 12.

The results suggest that MO-ALNS/D is statistically better than the MO-ALNS and MOEA/D algorithms, regarding the HCC metric. Therefore the Pareto front approximations achieved by MO-ALNS/D have better uniformity and extension. A graphical analysis is also performed to evaluate the typical behavior of the two best algorithms. A large instance was randomly selected and the Pareto front approximations found by the algorithms are shown in Fig. 13.

Fig. 13 helps to illustrate how algorithms behave in large instances of the problem. It can be observed that the Pareto front approximation of MO-ALNS/D is much better than the one of MO-ALNS, probably because of benefitting from the implicit parallelism of a population-based search and decomposition.

[5] Null-hypothesis (H_0) the average results of the metric are equals. Alternative hypothesis (H_1), the average results of the metric are different, that is, at least one of the results is different from the others. We verified that the data does not follow a normal distribution using the Shapiro-Wilk test.

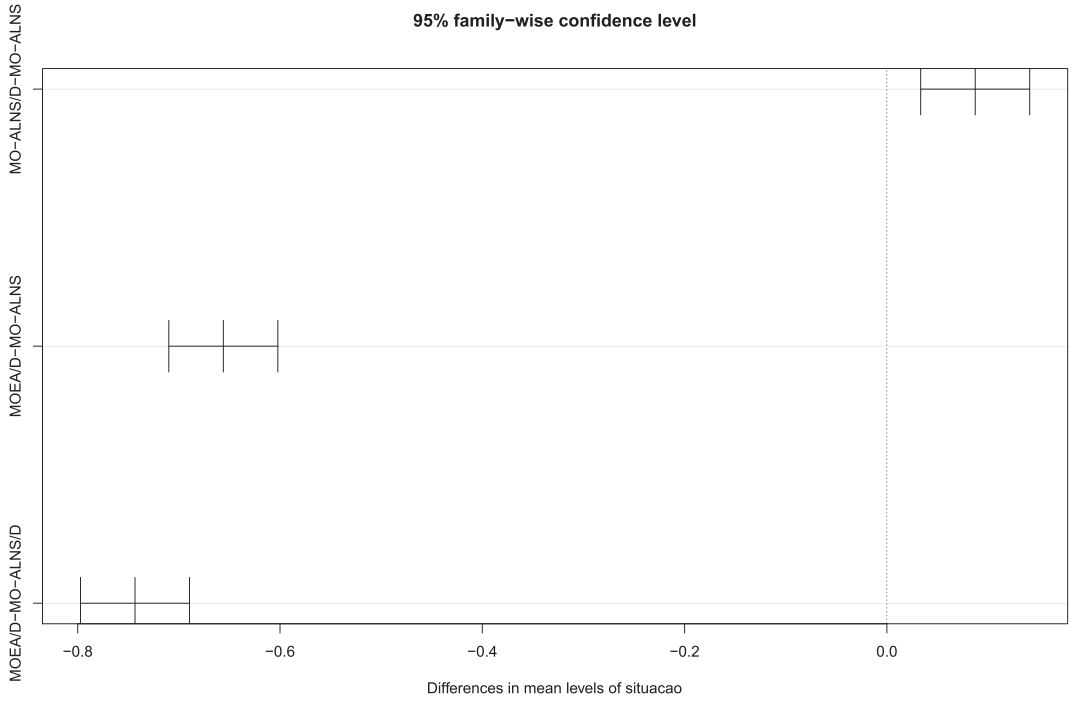

Fig. 11. Tukey HSD test results for the HV indicator.

Table 10
Results of the HCC indicator for MO-ALNS, MO-ALNS/D and MOEA/D algorithms.

N of jobs (n)	N of machines (m)	Setup time (S_{ijk})	MO-ALNS	MO-ALNS/D	MOEA/D
50	10	$U[1,9]$	785.580	987.022	234.534
		$U[1,124]$	1462.067	1632.118	374.014
	20	$U[1,9]$	377.797	428.030	219.868
		$U[1,124]$	321.796	767.863	173.054
	30	$U[1,9]$	186.924	260.785	157.394
		$U[1,124]$	1778.784	2411.798	155.750
100	10	$U[1,9]$	989.166	1997.447	691.072
		$U[1,124]$	1118.967	1623.842	696.266
	20	$U[1,9]$	761.315	911.411	544.676
		$U[1,124]$	710.803	990.797	592.069
	30	$U[1,9]$	351.530	500.012	507.960
		$U[1,124]$	3179.294	5161.904	46.020
150	10	$U[1,9]$	2195.810	2486.353	979.543
		$U[1,124]$	2689.037	2176.719	2185.338
	20	$U[1,9]$	710.263	1118.581	813.999
		$U[1,124]$	844.299	1160.862	907.669
	30	$U[1,9]$	273.688	621.843	264.065
		$U[1,124]$	2885.948	6241.462	1112.973
200	10	$U[1,9]$	1520.393	4125.787	2359.738
		$U[1,124]$	2892.856	4355.026	3637.622
	20	$U[1,9]$	781.307	1221.719	790.317
		$U[1,124]$	501.817	976.311	473.209
	30	$U[1,9]$	420.987	961.407	1570.445
		$U[1,124]$	2537.263	7043.882	1670.432
250	10	$U[1,9]$	1451.937	3290.318	2409.430
		$U[1,124]$	3494.344	8865.062	1806.802
	20	$U[1,9]$	1024.350	1256.221	662.576
		$U[1,124]$	3136.675	9654.496	2616.524
	30	$U[1,9]$	592.521	880.586	988.264
		$U[1,124]$	5841.017	10853.497	2735.411

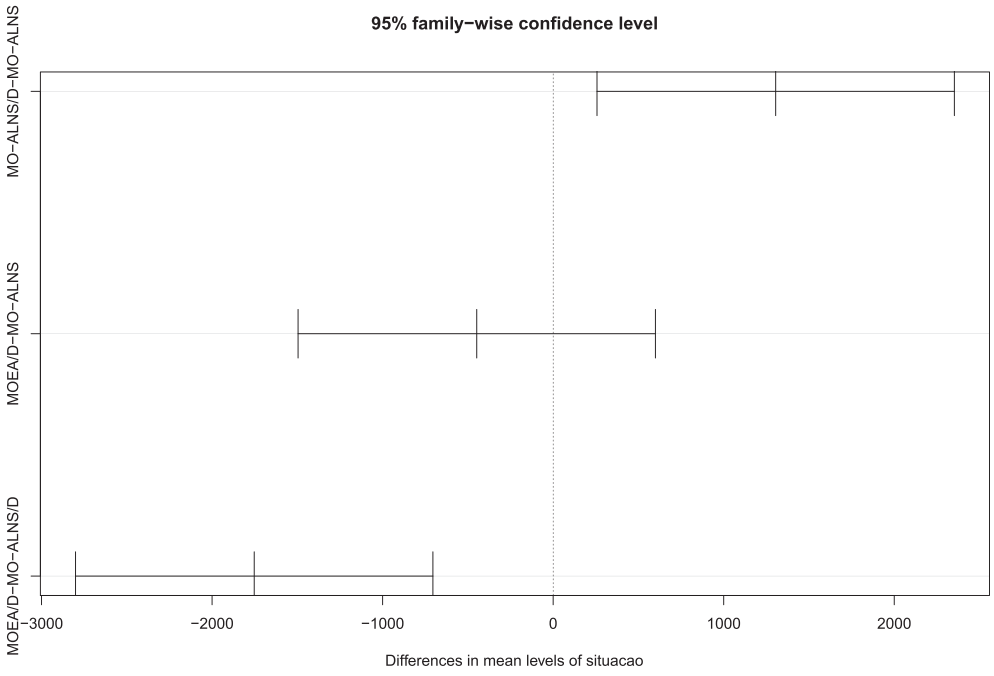

Fig. 12. Tukey HSD test results for the HCC indicator.

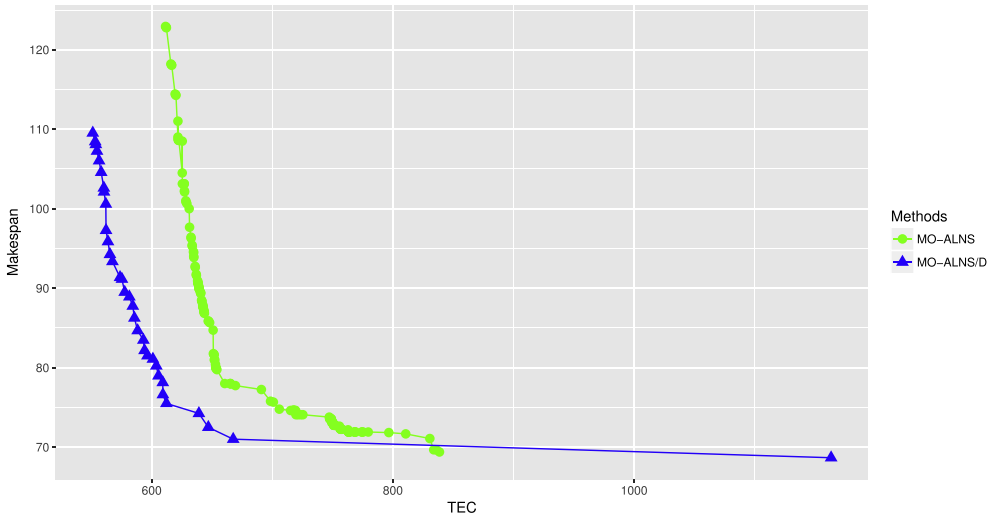

Fig. 13. Pareto front obtained with the MO-ALNS and MO-ALNS/D algorithms for an instance with 50 jobs and 10 machines.

5. Conclusions

This work approached the unrelated parallel machine scheduling problem with sequence dependent setup times, attempting to minimize the *makespan* and total energy consumption, $R_M|S_{ijk}|(C_{\max}, TEC)$ by the formal definition. This problem was recently defined in the context of green scheduling. The classical version of the problem seeks to minimize only the *makespan*. The minimization of the energy consumption implies in reduction of costs for the industries and the conscious use of environmental resources.

In the literature this problem was solved using exact methods. Thereby, only small and medium instances have been treated. The focus of this work was to propose multi-objective algorithms to solve large instances of this problem. Two multi-objective algorithms were defined for this purpose. Both algorithms are based on the ALNS algorithm with Learning Automata.

The first algorithm is called MO-ALNS and is a multi-objective version of the ALNS algorithm with Learning Automata. In the MO-ALNS the acceptance criterion was modified to account for the Pareto dominance. A new local search has also been added to change the operation modes. The MO-ALNS does not offer control of the diversification in the Pareto front approximation because the current solution is chosen randomly at each iteration. The second algorithm has the structure of the classical MOEA/D, however it uses the ALNS with Learning Automata to solve the scalar subproblems rather than genetic operators. This algorithm is called MO-ALNS/D. It is noteworthy that decomposition and aggregation in MOEA/D algorithm controls diversification of the Pareto front approximation. The MO-ALNS/D algorithm preserves this characteristic.

In the computational experiments the two multi-objective algorithms were validated using the small and medium instances of the literature, and the results of the ϵ-constrained method. A set of large instances was generated to compare the proposed algorithms. The two

proposed algorithms were compared against the classical MOEA/D for a set of large instances. The hypervolume, coverage of two sets and hierarchical cluster counting indicators were used to measure the quality of the results. It was observed that the MO-ALNS/D algorithm found better results than MO-ALNS in most cases for the three indicators. Statistical tests were applied to verify if there is statistical difference between the results of the hypervolume and hierarchical cluster counting indicators. The results of these statistical tests indicated that such differences are significant, thus confirming the benefit of Learning Automata and decomposition in the MO-ALNS/D algorithm. These findings show that the decomposition approach can be beneficial to the search not only for evolutionary based algorithms but also for other metaheuristic methods based on local search. We believe this is a significant contribution of this study.

In principle, it is believed that the MO-ALNS/D algorithm might be successfully applied to other multi-objective scheduling problems. It combines two very interesting algorithms, the ALNS that has been successfully applied to solve several single objective scheduling problems and the MOEA/D that has obtained excellent results in solving various multi-objective and many-objective problems. Furthermore, the problem addressed is general in the parallel machine scheduling class. All components of the proposed algorithms can be expanded to several other instances of real parallel scheduling problems.

As future work we intend to apply the MO-ALNS/D to other real multi-objective and many-objective scheduling problems. In addition, we plan to improve the algorithm with the use of more efficient techniques for generating weights and other decomposition functions.

Conflict of interest

The authors declare that they have no conflict of interest.

Acknowledgement

The authors would like to thank the support given by the Instituto Tecnológico Vale and the Brazilian agencies CAPES, CNPq, and FAPEMIG.

References

[1] N.E. Karkalos, A.P. Markopoulos, J.P. Davim, Computational Methods for Application in Industry 4.0, Springer, 2019.
[2] S.A. Mansouri, E. Aktas, U. Besikci, Green scheduling of a two-machine flow shop: trade-off between makespan and energy consumption, Eur. J. Oper. Res. 248 (2016) 772–788.
[3] M.L. Pinedo, Scheduling: Theory, Algorithms, and Systems, Springer, 2018.
[4] G. Rabadi, R.J. Moraga, A. Al-Salem, Heuristics for the unrelated parallel machine scheduling problem with setup times, J. Intell. Manuf. 17 (1) (2006) 85–97.
[5] M. Garey, D. Johnson, Computers and intractability: A Guide to the Theory of NP-Completeness, WH Freeman & Co., San Francisco 174.
[6] R.M. Karp, Reducibility among combinatorial problems, Complexity of Computer Computations 40 (4) (1972) 85–103.
[7] A. Allahverdi, The third comprehensive survey on scheduling problems with setup times/costs, Eur. J. Oper. Res. 246 (2) (2015) 345–378.
[8] E. Vallada, R. Ruiz, A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times, Eur. J. Oper. Res. 211 (3) (2011) 612–622.
[9] L.P. Cota, M.N. Haddad, M.J.F. Souza, V.N. Coelho, AIRP: a heuristic algorithm for solving the unrelated parallel machine sheduling problem, in: Proceedings of the 2014 IEEE Congress on Evolutionary Computation, CEC 2014), Beijing, 2014, pp. 1855–1862.
[10] M.N. Haddad, L.P. Cota, M.J.F. Souza, N. Maculan, AIV: a heuristic algorithm based on iterated local search and variable neighborhood descent for solving the unrelated parallel machine scheduling problem with setup times, in: Proceedings of the 16th International Conference on Enterprise Information Systems, ICEIS 2014, Lisbon, Portugal, 2014, pp. 376–383, https://doi.org/10.5220/0004884603760383.
[11] M.N. Haddad, L.P. Cota, M.J.F. Souza, N. Maculan, Solving the unrelated parallel machine scheduling problem with setup times by efficient algorithms based on iterated local search, Lecture Notes in Enterprise Information Systems 227 (2015) 131–148.
[12] P. Hansen, N. Mladenovic, J.A.M. Pérez, Variable neighborhood search: methods and applications, 4OR, Quartely Journal of the Belgian, French and Italian operations research societies 6 (2008) 319–360.
[13] T.T. Tran, A. Araujo, J.C. Beck, Decomposition methods for the parallel machine scheduling problem with setups, Inf. J. Comput. 28 (1) (2016) 83–95, https://doi.org/10.1287/ijoc.2015.0666.
[14] L.P. Cota, F.G. Guimarães, F.B. Oliveira, M.J.F. Souza, An adaptive large neighborhood search with learning automata for the unrelated parallel machine scheduling problem, in: Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC 2017), Donóstia - San Sebastian, 2017, pp. 185–192, https://doi.org/10.1109/CEC.2017.7969312.
[15] K.R. Baker, D. Trietsch, Principles of Sequencing and Scheduling, John Wiley & Sons, Inc., 2009https://doi.org/10.1002/9780470451793.
[16] S. A. Mansouri, E. Aktas, U. Besikci, Minimizing energy consumption and makespan in a two-machine flowshop scheduling problem, J. Oper. Res. Soc.:10.1057/jors.2016.4.
[17] H. Zhang, F. Zhao, K. Fang, J.W. Sutherland, Energy-conscious flow shop scheduling under time-of-use electricity tariffs, Proceedings of the Annals - Manufacturing Technology 63 (2014) 37–40.
[18] J. Ding, S. Song, C. Wu, Carbon-efficient scheduling of flow shops by multi-objective optimization, Eur. J. Oper. Res. 248 (2016) 758–771.
[19] Y. Liu, H. Dong, N. Lohse, S. Petrovic, A multi-objective genetic algorithm for optimisation of energy consumption and shop floor production performance, Int. J. Prod. Econ. 179 (2016) 259–272, https://doi.org/10.1016/j.ijpe.2016.06.019.
[20] S.A. Mansouri, E. Aktas, Minimizing energy consumption and makespan in a two-machine flowshop scheduling problem, J. Oper. Res. Soc. 67 (11) (2016) 1382–1394, https://doi.org/10.1057/jors.2016.4.
[21] C. Artigues, P. Lopez, A. Haït, The energy scheduling problem: industrial case-study and constraint propagation techniques, Int. J. Prod. Econ. 143 (1) (2013) 13–23, https://doi.org/10.1016/j.ijpe.2010.09.030, http://www.sciencedirect.com/science/article/pii/S0925527310003683.
[22] M. Aghelinejad, Y. Ouazene, A. Yalaoui, Production scheduling optimisation with machine state and time-dependent energy costs, Int. J. Prod. Res. 56 (16) (2018) 5558–5575, https://doi.org/10.1080/00207543.2017.1414969.
[23] M. Aghelinejad, Y. Ouazene, A. Yalaoui, Complexity analysis of energy-efficient single machine scheduling problems, Operations Research Perspectives 6 (2019) 100105, https://doi.org/10.1016/j.orp.2019.100105, http://www.sciencedirect.com/science/article/pii/S2214716018301702.
[24] S.A. Mansouri, E. Aktas, U. Besikci, Green scheduling of a two-machine flowshop: trade-off between makespan and energy consumption, Eur. J. Oper. Res. 248 (3) (2016) 772–788, https://doi.org/10.1016/j.ejor.2015.08.064, http://www.sciencedirect.com/science/article/pii/S0377221715008206.
[25] S. Wang, M. Liu, F. Chu, C. Chu, Bi-objective optimization of a single machine batch scheduling problem with energy cost consideration, J. Clean. Prod. 137 (2016) 1205–1215, https://doi.org/10.1016/j.jclepro.2016.07.206, http://www.sciencedirect.com/science/article/pii/S0959652616311118.
[26] S. Wang, X. Wang, J. Yu, S. Ma, M. Liu, Bi-objective identical parallel machine scheduling to minimize total energy consumption and makespan, J. Clean. Prod. 193 (2018) 424–440, https://doi.org/10.1016/j.jclepro.2018.05.056, http://www.sciencedirect.com/science/article/pii/S0959652618313866.
[27] H. Safarzadeh, S.T.A. Niaki, Bi-objective green scheduling in uniform parallel machine environments, J. Clean. Prod. 217 (2019) 559–572, https://doi.org/10.1016/j.jclepro.2019.01.166, http://www.sciencedirect.com/science/article/pii/S0959652619301854.
[28] X. Wu, A. Che, A memetic differential evolution algorithm for energy-efficient parallel machine scheduling, Omega 82 (2019) 155–165, https://doi.org/10.1016/j.omega.2018.01.001, http://www.sciencedirect.com/science/article/pii/S0305048317307922.
[29] H.R. Lourenço, O. Martin, T. Stützle, Iterated local search, in: F. Glover, G. Kochenberger (Eds.), Handbook of Metaheuristics, Vol. 57 of International Series in Operations Research & Management Science, Kluwer Academic Publishers, Norwell, MA, 2003, pp. 321–353.
[30] S. Ropke, D. Pisinger, An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows, Transp. Sci. 40 (2006) 455–472.
[31] A.C. Beezzão, J.-F. Cordeau, G. Laporte, H.H. Yanasse, Scheduling identical parallel machines with tooling constraints, Eur. J. Oper. Res. 257 (3) (2017) 834–844.
[32] V. Ghilas, E. Demir, T. Van Woensel, An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines, Comput. Oper. Res. 72 (C) (2016) 12–30, https://doi.org/10.1016/j.cor.2016.01.018.
[33] G. Mattos Ribeiro, G. Laporte, An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem, Comput. Oper. Res. 39 (3) (2012) 728–735, https://doi.org/10.1016/j.cor.2011.05.005.
[34] J.E.C. Arroyo, R.S. Ottoni, A.P. Oliveira, Multi-objective variable neighborhood search algorithms for a single machine scheduling problem with distinct due windows, Electronic Notes in Theoretical Computer Science, in: Proceedings of the 2011 Latin American Conference in Informatics (CLEI), vol. 281, 2011, pp. 5–19, https://doi.org/10.1016/j.entcs.2011.11.022.
[35] M.J. Geiger, Decision support for multi-objective flow shop scheduling by the pareto iterated local search methodology, Comput. Ind. Eng. 61 (3) (2011) 805–812, https://doi.org/10.1016/j.cie.2011.05.013.
[36] A.P. Rifai, H. Nguyen, S.Z.M. Dawal, Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling, Appl. Soft Comput. 40 (2016) 42–57, https://doi.org/10.1016/j.asoc.2015.11.034.
[37] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P.N. Suganthan, Q. Zhang, Multiobjective evolutionary algorithms: a survey of the state of the art, Swarm and Evolutionary Computation 1 (1) (2011) 32–49, https://doi.org/10.1016/j.swevo.2011.03.001.

[38] Q. Zhang, H. Li, Moea/d: a multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (6) (2007) 712–731, https://doi.org/10.1109/TEVC.2007.892759.
[39] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: nsga-ii, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197, https://doi.org/10.1109/4235.996017.
[40] K. Deb, H. Jain, An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints, IEEE Trans. Evol. Comput. 18 (4) (2014) 577–601, https://doi.org/10.1109/tevc.2013.2281535.
[41] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: improving the strength pareto evolutionary algorithm for multiobjective optimization, in: K. Giannakoglou et al. (Ed.), Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001), International Center for Numerical Methods in Engineering (CIMNE), 2002, pp. 95–100.
[42] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: overview and conceptual comparison, ACM Comput. Surv. 35 (3) (2003) 268–308, https://doi.org/10.1145/937503.937505.
[43] I. Boussaïd, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Inf. Sci. 237 (2013) 82–117, https://doi.org/10.1016/j.ins.2013.02.041.
[44] P. Siarry (Ed.), Metaheuristics, Springer International Publishing, 2016.
[45] L.P. Cota, V.N. Coelho, F.G. Guimarães, M.J.F. Souza, Bi-criteria formulation for green scheduling with unrelated parallel machines with sequence dependent setup times, Int. Trans. Oper. Res. (2018), https://doi.org/10.1111/itor.12566.
[46] R. Graham, E. Lawler, J. Lenstra, A. Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, Annals of discrete Mathematics 5 (2) (1979) 287–326.
[47] M.L. Pinedo, Scheduling: Theory, Algorithms, and Systems, third ed., Springer Publishing Company, Incorporated, 2008.
[48] M.M. Alipour, A learning automata based algorithm for solving traveling salesman problem improved by frequency-based pruning, Int. J. Comput. Appl. 46 (17) (2012) 7–13, https://doi.org/10.5120/7007-9328.
[49] K.S. Narendra, M.A.L. Thathachar, Learning automata - a survey, IEEE Transactions on Systems, Man, and Cybernetics SMC- 4 (4) (1974) 323–334, https://doi.org/10.1109/TSMC.1974.5408453.
[50] K.S. Narendra, K.S. Thathachar, Learning Automata: an Introduction, Prentice-Hall, New York, 1989.
[51] K.S. Narendra, M.A. Thathachar, Learning Automata: an Introduction, Courier Corporation, 2012.
[52] R. Vafashoar, M.R. Meybodi, Multi swarm bare bones particle swarm optimization with distribution adaption, Appl. Soft Comput. 47 (2016) 534–552, https://doi.org/10.1016/j.asoc.2016.06.028.
[53] T. Lust, J. Teghem, Two-phase pareto local search for the biobjective traveling, J. Heuristics 16 (2010) 475–510.
[54] K.R. Baker, Introduction to Sequencing and Scheduling, John Wiley & Sons, Inc., 1974.
[55] M. Souza, I. Coelho, S. Ribas, H. Santos, L. Merschmann, A hybrid heuristic algorithm for the open-pit-mining operational planning problem, Eur. J. Oper. Res. 207 (2) (2010) 1041–1051.
[56] H. Scheffé, Experiments with mixtures, J. R. Stat. Soc. 20 (2) (1958) 344–360.
[57] A. Trivedi, D. Srinivasan, K. Sanyal, A. Ghosh, A survey of multiobjective evolutionary algorithms based on decomposition, IEEE Trans. Evol. Comput. 21 (3) (2017) 440–462, https://doi.org/10.1109/TEVC.2016.2608507.
[58] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach, IEEE Trans. Evol. Comput. 3 (1999) 257–271.
[59] D. Montgomery, Design and Analysis of Experiments, fifth ed., John Wiley & Sons, New York, NY, 2007.
[60] S.S. Shapiro, M.B. Wilk, An analysis of variance test for normality (complete samples), Biometrika 52 (1965) 591–611.
[61] C. Ahilan, S. Kumanan, N. Sivakumaran, J.E.R. Dhas, Modeling and prediction of machining quality in cnc turning process using intelligent hybrid decision making tools, Appl. Soft Comput. 13 (3) (2013) 1543–1551, https://doi.org/10.1016/j.asoc.2012.03.071.
[62] P. Shaw, Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems, Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, pp. 417–431.
[63] Y. Tian, R. Cheng, X. Zhang, Y. Jin, Platemo: a matlab platform for evolutionary multi-objective optimization [educational forum], IEEE Comput. Intell. Mag. 12 (4) (2017) 73–87.
[64] A. Sadegheih, Scheduling problem using genetic algorithm, simulated annealing and the effects of parameter values on ga performance, Appl. Math. Model. 30 (2) (2006) 147–154.
[65] J. Knowles, D. Corne, On metrics for comparing nondominated sets, in: Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on, vol. 1, 2002, pp. 711–716, https://doi.org/10.1109/CEC.2002.1007013.
[66] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, V.G. da Fonseca, Performance assessment of multiobjective optimizers: an analysis and review, IEEE Trans. Evol. Comput. 7 (2) (2003) 117–132, https://doi.org/10.1109/TEVC.2003.810758.
[67] F.G. Guimarães, E.F. Wanner, R.H.C. Takahashi, A quality metric for multi-objective optimization based on hierarchical clustering techniques, in: 2009 IEEE Congress on Evolutionary Computation, 2009, pp. 3292–3299, https://doi.org/10.1109/CEC.2009.4983362.