Applied Intelligence
https://doi.org/10.1007/s10489-018-1160-z

A hybrid estimation of distribution algorithm for flexible job-shop scheduling problems with process plan flexibility

Ricardo Pérez-Rodríguez[1] · **Arturo Hernández-Aguirre**[2]

Abstract
The flexible job-shop environments have become increasingly significant because of rapid improvements on shop floors such as production technologies, manufacturing processes and systems. Several real manufacturing and service companies have had to use alternative machines or processes for each operation and the availability of alternative process plans for each job in order to achieve good performance on the shop floor where conflicting objectives are common, e.g. the overall completion time for all jobs and the workload of the most loaded machine. In this paper, we propose a Pareto approach based on the hybridization of an estimation of distribution algorithm and the Mallows distribution in order to build better sequences for flexible job-shop scheduling problems with process plan flexibility and to solve conflicting objectives. This hybrid approach exploits the Pareto-front information used as an input parameter in the Mallows distribution. Various instances and numerical experiments are presented to illustrate that shop floor performance can be noticeably improved using the proposed approach. In addition, statistical tests are executed to validate this novel research.

Keywords Multiobjective optimization · Evolutionary optimization · Estimation of distribution algorithm · Flexible job-shop scheduling · Mallows distribution · Routing flexibility · Process plan flexibility

1 Introduction

The flexible job-shop scheduling problem (FJSP) is commonly defined as follows: there are n jobs $J = \{J_1 J_2, \ldots, J_n\}$ to be processed on m machines $M = \{M_1 M_2, \ldots, M_m\}$. A job J_i is formed by a sequence of n_i operations $\{O_{i,1}; O_{i,2}; \ldots; O_{i,n_i}\}$ performed one after another according to a given sequence. The execution of $O_{i,j}$ requires one machine out of a set of $m_{i,j}$ given machines $M_{i,j} \subseteq M$. Preemption is not allowed, i.e., every operation must be completed without interruption once it starts. For each job, the corresponding operations have to be processed in the given order, that is, the starting time for an operation must not be earlier than the point at which the preceding operation is completed. This constraint is imposed simultaneously on all appropriate pairs of operations. The most common objective is to minimize the overall completion time for all the jobs. The FJSP assumes only one feasible process plan for each job. Although it is common in real-world manufacturing and service environments, a wider space for choices exists. Normally, there is more than one option to construct schedules among alternative machines in which to perform an operation and from alternative process plans of a job, i.e. routing flexibility and process plan flexibility respectively. In order to make the flexible job-shop scheduling problem more understandable, there is an example of the inner ring grinding process plan for a spherical roller bearing, whose process plan flexibility is typical in the bearing manufacture industry. The process plan is detailed below and involves eleven operations. It does not matter if the second operation is processed after the third, so the processing sequence can be rearranged. The eighth and ninth operations have the same situation. In addition, the sixth operation can be placed in any position between the fifth and the eleventh operation. In this example, we have three alternative process plans for an inner ring.

✉ Ricardo Pérez-Rodríguez
ricardo.perez@cimat.mx

Arturo Hernández-Aguirre
artha@cimat.mx

[1] CONACYT - CIMAT A.C., Bartolomé de las Casas 314, Barrio la Estación, 20259, Aguascalientes, Ags, México

[2] CIMAT A.C., Jalisco s/n, Valenciana, 36240, Guanajuato, Gto, México

Published online: 18 April 2018

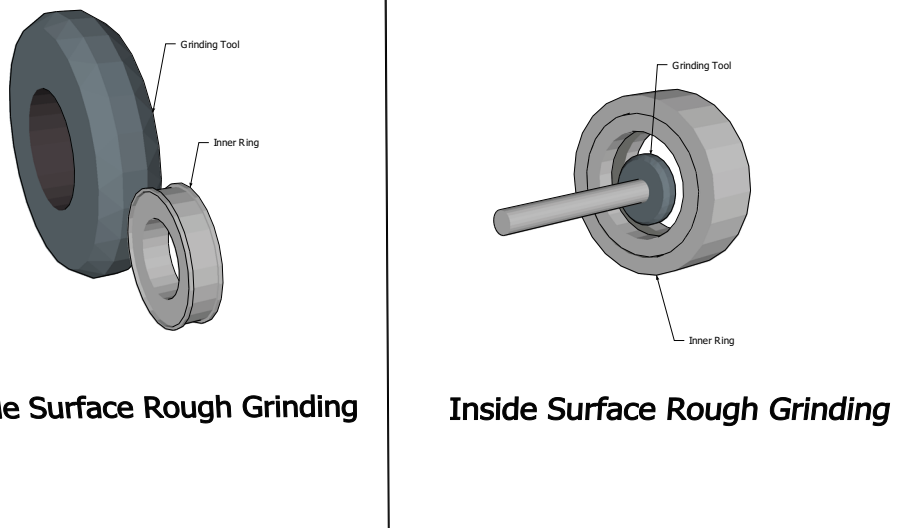


Fig. 1 A graphical description of a process plan flexibility

Figure 1 illustrates a graphical description for the second operation and the third operation. Both operations can be rearranged without affecting the overall process.

Operation number	Operation name
1	ring face grinding
2	outside surface rough grinding
3	inside surface rough grinding
4	raceway rough grinding
5	temper treatment
6	chamfer brightening
7	outside surface fine grinding
8	inside surface fine grinding
9	raceway fine grinding
10	rib grinding
11	surface brightening

The FJSP incorporates the routing flexibility component. However the FJSP with process plan flexibility (FJSP-PPF) is a more complicated combinatorial optimization problem Özgüven, Özbakır, and Yavuz [40]. The FJSP-PPF considers multiple process plans for the jobs by excluding the assumption of only one feasible process plan for each job. The FJSP-PPF consists of a set of n jobs, each having a set of P process plans. The process plan p_k of job j (p_{kj}) is an ordered list of n_i operations. It is assumed that the process plans are known in advance. The execution of the operation i of job j in the process plan p_{kj} ($O_{i,j,p_{kj}}$) requires one machine out of a set of $m_{i,j}$ given machines $M_{i,j} \subseteq M$. Only one of the alternative plans is adopted for each job. The FJSP-PPF not only deals with routing and sequencing sub-problems but also the process plan selection sub-problem: choosing a process plan for each job j from a given set of P_j process plans, assigning each operation ($O_{i,j,p_{kj}}$) to a machine selected from the set $M_{i,j}$ and ordering operations in the machines so that conflicting objectives are optimized such as the overall completion time for all the jobs and the maximum working time spent on any machine.

A complete formulation of the mathematical integer linear programming model for the FJSP-PFF was developed by Özgüven et al. [40]. Figure 2 shows a scheduling example with four jobs $J = \{J_1, J_2, J_3, J_4\}$. Each job J_i is formed by a sequence of three operations $\{O_{i,1}; O_{i,2}; O_{i,3}\}$ performed one after the other. Table 1 details the alternative machines for each operation with two different process plans. The scheduling scheme, shown in Fig. 2, satisfies the set of constraints for the traditional FJSP-PPF, i.e., it ensures that all the operations are assigned to a single machine. Furthermore, it takes care of the requirement that the operations must respect precedence, i.e., the precedence relationships between the operations of a job are not violated. In addition, the solution depicted in Fig. 2 guarantees that multiple operations cannot be executed at the same time on any possible machine. Finally, the schedule determines the completion times (of the final operations) of the jobs, the makespan, and determines the maximal workload. Additionally, Fig. 2 presents the alternative process plan B for the aforementioned program. Based on Fig. 2, the performance is improved using the second process plan, and choosing alternative machines.

Any solution to the FJSP-PPF should be a combination of process plan selection, operation scheduling decision, and machine assignment. Thus, the selection of a process plan for each job, the processing sequence of operations on the

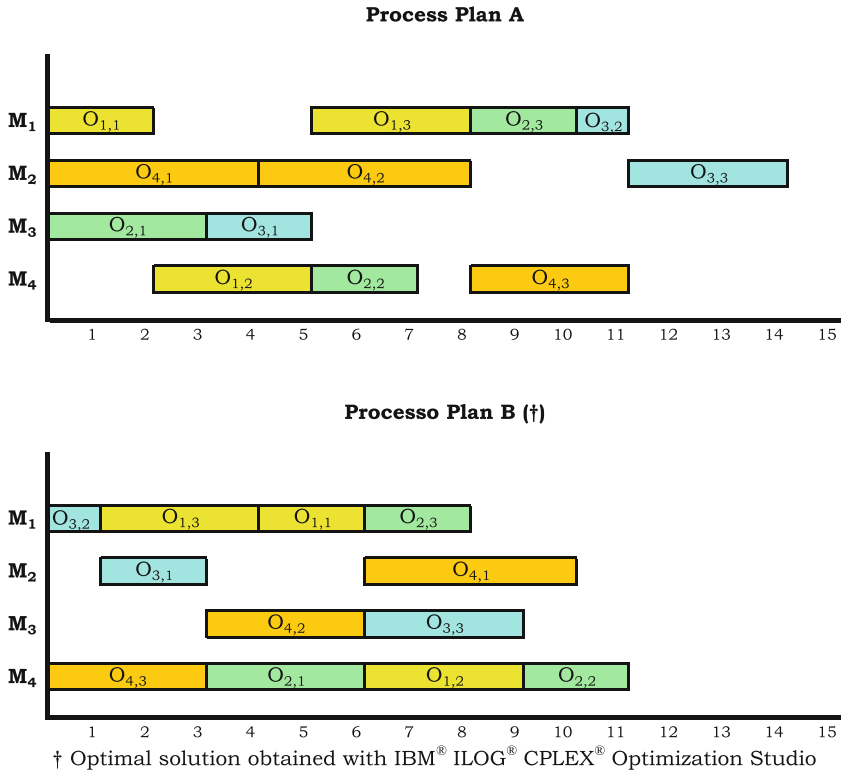


Fig. 2 Illustration of a schedule for the FJSP-PPF

machines and the assignment of operations on machines should express a solution. The processing sequence of operations in the machines can be constructed using a permutation-based representation. In this context, an operation sequence vector is a permutation for the FJSP-PPF and a set of permutations is a set of operation sequence vectors. In this research, we intend to characterize the space of possible solutions explicitly, i.e., to characterize a feasible subset of permutations by listing them based on the constraints of the problem. The proposal is to use a probability distribution associated with the space of possible solutions. An example is shown in Fig. 3 in order to understand how a probability distribution is sensitive to solutions. Different possible solutions are provided at the beginning of Fig. 3, i.e., different feasible permutations are used as an input parameter in order to build probability distributions. Each position, i.e., each column from the set of permutations is considered as an independent variable X. Based on Fig. 3, we have five independent variables, i.e., five positions (or columns) X1, X2, and so on. Therefore, we can build five probability distributions using the set of permutations in Fig. 3. After that, we select a subset of permutations, the best hypothetical solutions, and build the corresponding updated probability distributions. These are shown at the lower part of Fig. 3. The updated probability distributions are different from the previous probability distributions because they are sensitive to solutions used in the example.

In order to obtain an explicit probability distribution over the FJSP-PPF, we use an Estimation of Distribution Algorithm (EDA). The EDA is another paradigm in the field of evolutionary computation. Introduced by Mühlenbein and Paaβ [39] and compared to other evolutionary algorithms, the EDA reproduces a new population implicitly instead of using traditional evolutionary operators. To create an EDA, a probability model of the most promising area is generated by statistical information based on the search experience, and then the probability model is used for sampling to generate new individuals. The EDA makes use of the probability model to describe the distribution of the solution space while the updating process reflects the evolutionary trend of the population. To describe the distribution of the solution space mentioned above, the EDA tries to determine the relationship or interaction between the variables of the problem. However, how can we improve the estimation of the relationships between the variables of the problem? How influential is the estimation mentioned in the performance of the algorithm? We use the Mallows distribution in this research to obtain a better estimate of the relationships between the variables. The Mallows model was initially proposed by Mallows [35] and later was improved by Fligner and Verducci [18] through the generalized Mallows distribution (GMD). A more extensive explanation of the GMD characteristics is found in Fligner and Verducci [18] and Fligner and Verducci [19]. More recently, Ceberio et al. [8] contributed to an

Table 1 An FJSP-PPF example

Operations Job, Precedence	Processing time machines			
	M_1	M_2	M_3	M_4
Process Plan A				
$O_{1,1}$	2	4	3	5
$O_{1,2}$	4	4	–	3
$O_{1,3}$	3	–	–	–
$O_{2,1}$	3	–	3	3
$O_{2,2}$		5	6	2
$O_{2,3}$	2	–	–	–
$O_{3,1}$		2	3	3
$O_{3,2}$	1	–	–	–
$O_{3,3}$		3	3	–
$O_{4,1}$	6	4	5	5
$O_{4,2}$	2	4	3	5
$O_{4,3}$	4	4	–	3
Process plan B				
$O_{1,3}$	3	–	–	–
$O_{1,1}$	2	4	3	5
$O_{1,2}$	4	4	–	3
$O_{2,1}$	3	–	3	3
$O_{2,3}$	2	–	–	–
$O_{2,2}$		5	6	2
$O_{3,2}$	1	–	–	–
$O_{3,1}$		2	3	3
$O_{3,3}$		3	3	–
$O_{4,3}$	4	4	–	3
$O_{4,2}$	2	4	3	5
$O_{4,1}$	6	4	5	5

initial application of the GMD to solve the flow-shop scheduling problem (FSP) coupled with an EDA.

The GMD is a popular exponential model on the rankings [37]. Based on Fligner and Verducci [19], the GMD is suitable when there is a need to rank a fixed set of k items according to some judgment. The process of ranking the items is decomposed into $k-1$ stages, i.e., $k-1$ positions (or columns) as shown in Fig. 3. The general problem is to characterize the population based on a random sample of rankings. Historically, models for random rankings emerged from the literature on paired comparisons, where the probability that one item is being preferred to another is as specified by a given model. Fligner and Verducci [19]. A basic model can be built to rank the items, i.e., the most preferred item is selected in the first stage, the best of the remaining items is selected in the second stage, and so on until the least preferred item is selected by default. In this research the items mentioned above are operations to sequence in machines. Whatever the nature of these operations, each processing sequence of operations in the machines produces a ranking. Previous research uses probability models for adoption at each stage, and the properties of the resulting models for randomly sampled rankings have been investigated for years. Some probability models assume that each item i is thought to have an intrinsic value θ_i, and the probability of choosing a particular item i at any stage, depends on the set of items not previously chosen. Other models add interaction terms that theoretically extend the usefulness of previous approach when the basic model does not adequately describe the data. The major difficulty with these formulations is the analytic intractability of the models [19], especially when the number k of items under consideration is more than a few. The GMD avoids the analytical difficulties by assuming that the accuracy of the choice made at any stage is independent of the accuracies in the other stages. The accuracy is assessed with respect to a central ranking that indicates the general consensus, obtained from the population of vectors with respect to the ordering of the items, i.e., operations to sequence in machines for the FJSP-PPF. Given a set of rankings (permutations), over n items (operations), the central ranking (central permutation) is the ranking that minimizes the total number of disagreements with rankings contained in the set of rankings [2]. Various measures of agreement have been proposed such as Kendall's metric [18]. The Kendall's metric has been the measure of choice in many recent applications centered on the analysis of ranked data [13]. In the GMD such as Mallows model, the probability of observing any ranking π decreases as the distance (Kendall-tau metric) between π and the central ranking increases. Finding the central ranking is unfortunately a computational challenge, since the problem is NP-hard even for only four items [5, 13]. Since the problem is important across a variety of fields, many researchers across these fields have converged on finding good and practical algorithms for its solution [2]. Once a central ranking has been determined, the GMD creates a multistage model that is parameterized by the probabilities of the different degrees of accuracy in each stage.

Although the EDA was originally designed to solve integer or real-valued domain problems, it cannot produce a feasible solution from a probability model built on a permutation-based representation [29]. Furthermore, the EDA requires being adapted to deal with permutation-based problems by means of a modification in the algorithm process. Then, the implementation of a specific probability model for this issue would be more competitive against other models. The Mallows model [35] is a distance-based exponential probabilistic model considered analogous to the Gaussian probability distribution over the space of

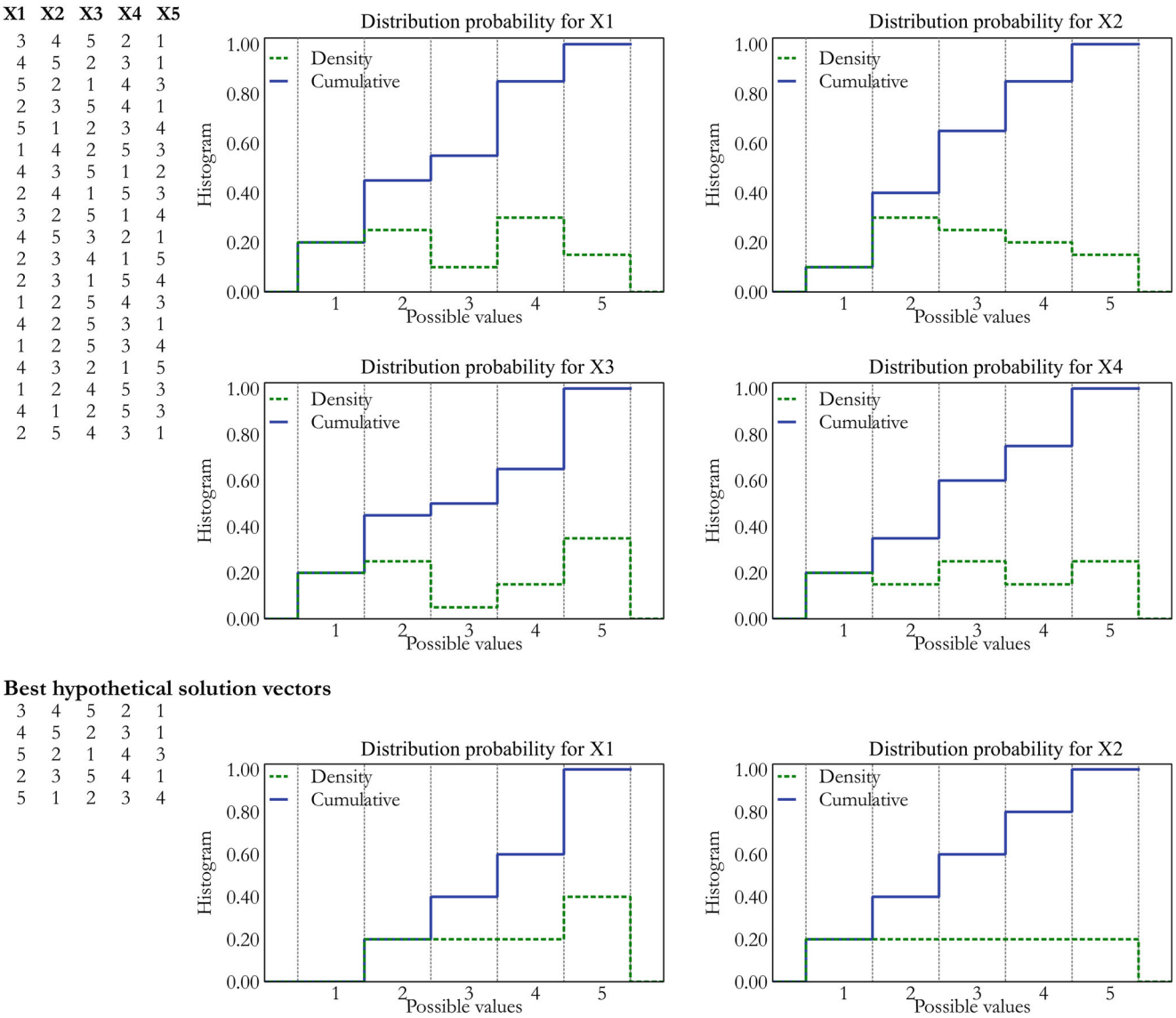


Fig. 3 An example of a probability distribution associated with the space of possible solutions

permutations. In the Mallows distribution, the probability value of each permutation $\sigma \in T_n$(where T_n stands for the set of $n!$ permutations of n items) depends on just two parameters: a spread parameter θ and the distance to a central permutation σ_0. This model assigns to each permutation σ a probability that decays exponentially with respect to its distance to the central permutation σ_0. In this way, $P(\sigma)$ can be considered as an exponential non-uniform distance-based ranking model [8]. Both parameters, i.e., the spread parameter θ and the central permutation σ_0, are estimated. Therefore, different estimates of the parameters mentioned should be considered in order to get a better performance to solve combinatorial optimization problems such as FJSP-PPF.

Contrary to current research where diverse techniques and approaches have been proposed in order to solve the problem that is being examined, this paper contributes to the state of the art as follows:

- To introduce the GMD, detailed in Fligner and Verducci's [18] research, for the FJSP-PPF as a way of estimating an explicit probability distribution in the domain of permutations for any EDA.
- To apply the GMD coupled with an EDA, called MEDA (Mallows Estimation of Distribution Algorithm), for solving the FJSP-PPF on a multi-objective perspective.
- To propose a different way of improving the estimation of the central permutation σ_0 in the GMD process using

a Pareto-front approach to help the MEDA to find better solutions for the FJSP-PPF.
- The purpose of this research is to propose a different experimental technique to enhance the performance of the algorithm mentioned above for solving the FJSP-PPF using a Pareto-front approach.

The proposed algorithm has a wide range of applications for automotive, aerospace, chemical, metallurgical, pharmaceutical, and food industries among others. The users of this contribution are academics, engineers, managers and practitioners related to the scheduling of manufacturing operations.

The remainder of this paper is organized as follows: in Section 2, the related work is depicted. In Section 3, the proposed MEDA is detailed. In Section 4, computational results and comparison with other multiobjective and recent algorithms are presented. The concluding remarks are made in Section 5.

2 Related work

2.1 Mathematical programming

Özgüven et al. [40] detail the mathematical models for job-shop scheduling problems with routing and process plan flexibility. These are mixed-integer linear programming models. Fattahi et al. [16] compared the first model with the randomly generated test problems. The instances mentioned divides into two categories: small and medium size. The results achieved in the same test problems indicate that the overall performance of the Özgüven et al. model is superior to that of the Fattahi et al. model. For small size instances the makespan and the overall completion time for all the jobs, obtained by the Özgüven et al. model and Fattahi et al. model overlap, and both models find optimal solutions. However, a significant difference occurs between the two models for medium-large size problems. Although the Özgüven et al. model significantly dominates the Fattahi et al. model in terms of the makespan results, it cannot find optimal solutions for medium-large size instances. The Özgüven et al. model is extended to enable it to handle the process plan flexibility component. According to the research of Özgüven et al. [40], there is no comparable mathematical model in the literature. Therefore, the outcome is a second model that was presented with computational results on hypothetically generated test problems. The second model finds the optimal solutions for a few small size problems. For the remaining ones, as the size of the problem increases, the gap between the best linear solution, in terms of the makespan results, and the best integer solution widens. Also, Özgüven et al. [41] explains an advanced form of the FJSP-PPF that considers separable and non-separable sequence dependent setup times using two mixed-integer goal programming models. In the first model, the sequence dependent setup times are non-separable and in the second model the sequence dependent setup times are separable. In addition, minimizing the makespan is the primary goal while balancing the workloads of the machines is the secondary goal for both models. In Özgüven et al.'s [41] research about the literature for the mathematical modeling approaches, all except one of the models they came across have different variations of the scheduling problems either with routing or with process plan flexibility, but not both. The only exception is the model of Lee et al. [31]. Since Özgüven et al. [41] came across no test problem that addresses the models developed in their paper the computational results are obtained on small size instances. Although the models do not find optimal solutions based on the results, the authors analyze the significance of the differences between the effects of non-separable and separable setup times on the performance of scheduling system. The results denote that there is a significant difference between the performance values of the problems with non-separable and separable setup times. Although the models mentioned above require assumptions to satisfy from an industrial perspective, these are used to continue research on the problem.

2.2 Evolutionary algorithms

Lee et al. [31] proposed a genetic algorithm-based heuristic approach for scheduling in a manufacturing supply chain. The approach considers industrial real features, such as outsourcing, customer due dates and alternative process plans for job types. The objective is to minimize the makespan for the due date of each order. The authors focused on the finite capacity and resource capacity of a flow shop and studied the strength of integrated process, planning, and scheduling, with specific due dates for orders. In numerical experiments, the GA-based approach efficiently solved the problem and produced the best process plans (operation sequence and machine selection with outsourcing) and schedules for all the orders in order to minimize the makespan. The mentioned experiments were tested on small instances, ranging from one or two customers, less than ten jobs and machines. In addition, the experiments were tested on the data from Sundaram and Fu [49] with only five jobs and five machines. There is no information on experiments with large size problems.

There are other papers that considered the process plan flexibility concept using evolutionary algorithms such as the Kim et al. [28] research. The authors presented an artificially intelligent search technique, called symbiotic evolutionary algorithm to address the integrated problem of process planning and scheduling in job-shop flexible

manufacturing systems where it is possible to generate many feasible process plans for each job. This approach is characterized by its ability to perform the effective and simultaneous search of the solution space formed by the two problems and then the scheduling problem is considered under the constraint of the solution. For the process planning problem, minimizing the absolute deviation of machine loads is used as an objective in this paper. That is, minimize $\sum_i |w_i - \overline{w}|$, where w_i is the load of machine i and $\overline{w}$ is the mean of machine loads. The scheduling objectives of both minimizing the makespan and minimizing mean flow time involve the maximum utilization of the machines. The performance of the proposed algorithm was compared with those of a traditional hierarchical approach and an existing cooperative coevolutionary algorithm. Twenty-four test problems were constructed by the authors. The test problems range between 6 and 18 jobs. The number of operations, corresponding to the jobs, ranges between 8 and 22. The experimental results show that the proposed algorithm outperforms the compared algorithms. Although there is no information about the optimal solutions with respect to the test problems used in the comparison, the main contribution of the authors is to adopt the strategies of localized interactions, steady-state reproduction, and random symbiotic partner selection to improve the algorithm, i.e., to enhance population diversity and search efficiency.

Park and Choi [43] also focused on the integration problem of process planning and scheduling in a job-shop environment. In their research, a GA-based scheduling method was considered to minimize the makespan of all of the jobs in the integration problem with alternative machines and alternative operations sequences. In order to evaluate the performance of the suggested GA, the authors used an instance of an injection molding company with multiple process plans. Through this real case, the authors compared two kinds of scheduling results, that is, the integrated scheduling considering the alternative machines and operations sequences and the traditional scheduling considering them sequentially. Based on the results, the integrated scheduling reduces the makespan for the real case. In addition, a couple of benchmark cases were tested for performance evaluation. The first benchmark case was based on Chambers and Barnes [9] using the Fisher and Thompson [17] 10x10 instance for making an alternative machine problem. The criterion is the sum of processing times required for each machine. The second benchmark case was based on Tan [50]. The instance mentioned has five jobs with twenty operations. In this problem, most jobs and operations have more than four operations sequences and alternative machines. Additionally, each job has a different release time. For both cases, the integrated scheduling reduces the makespan. A good contribution of the authors is the chromosome representation. It can maintain feasibility after a genetic operator and easily handle the integration problem. Although there is no information about the optimal solutions with respect to the test problems used in the comparison, Park and Choi's [43] research is another example of implementing evolutionary algorithms in real manufacturing processes.

Rossi and Dini [46] proposed an ant colony optimization-based software system for solving flexible manufacturing systems scheduling in a job-shop environment with routing flexibility, sequence-dependent setup and transportation time. The algorithm was tested using standard benchmarks and problems instances belong to the following datasets: Fisher and Thompson [17] instances, Lawrence [30] instances, and Applegate and Cook [3] instances. The effectiveness of the proposed system was verified by comparison with alternative approaches such as ant colony algorithm, genetic algorithm, and local search algorithm. Based on the results, the ant colony approach proposed outperforms the others with respect to the average makespan. Also, the approach mentioned outperforms the others with respect to the mean relative error between the average makespan and the optimal solution. The key point of their research is to use an effective pheromone trail coding and tailored ant colony operators for improving solution quality. In addition, the flexible manufacturing system (FMS) layout is being considered in this research.

2.3 The multi-objective approach

Gao et al. [20] addressed the FJSP with three objectives: the makespan, the maximal machine workload i.e., the maximum working time spent in any machine and the total workload, which represents the total working time assigned to all machines. A hybrid GA with variable neighborhood descent (VND) is detailed in their research. Several sets of problem instances were considered for the computational study of the experiment. One of the sets is from Fisher and Thompson [17], and another set is from Lawrence [30]. Almost 50% of the instances, 87 out of 181 of the proposed hybrid GA outperforms others with respect to optimal solutions. Although the Gao et al. [20] research contributes to the use of an advanced crossover and mutation operators to adapt to the special chromosome structure and the characteristics of the problem, the hybrid approach was not implemented in the industrial environments.

Huang et al. [23] presented a multi-objective particle swarm optimization integrating with variable neighborhood search for the FJSP to optimize three objectives, i.e., the makespan, the total workload and the critical machine workload. Four Kacem et al. [26] instances, ranging from 4 jobs x 5 machines to 15 jobs x 10 machines and ten Brandimarte [7] instances, ranging from 10 jobs x

6 machines to 20 jobs x 15 machines were used to evaluate the performance of their algorithm and several published algorithms were applied in comparison with the proposed algorithm. Based on the results reported, the proposed algorithm obtains more high-quality solutions for the instances mentioned. Their contribution is to use cognitive memory and social memory strategies to update and preserve the non-dominated individuals. The industrial perspective with the process plan flexibility was not considered in this research.

Dauod et al. [14] contributed to a GA-based approach to an order-scheduling problem in a mail-order pharmacy automation system. The objectives were to minimize the makespan and the collation delays, i.e., the completion time difference between the first and the last medications within the same order was defined as the order collation delay. The objective of this analysis is to understand how machine flexibility and multi-medication orders proportion affect the system performance, mainly in terms of collation delays. Therefore, the authors built three scenarios to compare the GA proposed with the CPLEX tool optimization software and two industry heuristics: the longest processing time (LPT) and the least total workload (LTW). However, no standard instances were used in the comparison and no similar algorithms were considered in this research.

2.4 The Pareto approach

New research is considering the use of the Pareto approach for scheduling problems. Pareto is hybridized with other evolutionary algorithms. Kacem et al. [26] show a Pareto approach based on the hybridization of fuzzy logic (FL) and an evolutionary algorithm (EA) to solve the FJSP. The objective is to minimize the makespan, the total workload of machines and the workload of the most loaded machine. In the research, many examples are presented to illustrate some theoretical considerations and to show the efficiency of the suggested methodology. The authors use numerous instances that have been simulated to test the efficiency of the suggested approach. In addition, the authors present the lower-bound values to give a precise evaluation of the solution quality. However, no algorithms have been used for a comparison with respect to the simulated instances. The hybridization proposed by Kacem et al. [26] exploits the knowledge representation capabilities of FL and the adaptive capabilities of EA. The industrial perspective is out of scope for this research.

Yue et al. [56] proposed the hybridization Pareto approach with an artificial bee colony algorithm for a single machine-scheduling problem with sequence-dependent setup times. The aim is to minimize the makespan and the total weighted tardiness simultaneously. The authors illustrate computational experiments and results over five different categories of problems. All test problems consider setup times and due dates for each job. These test problems are much closer to real-world problems. The authors make comparisons of results among three famous multi-objective optimization algorithms, i.e., the non-dominated sorting genetic algorithm II (NSGAII) Deb et al. [15], the improved strength Pareto evolutionary algorithm (SPEA2) Zitzler et al. [57] and particle swarm optimization algorithm (PSO) Kennedy and Eberhart [27]. The performance of the proposed algorithm is based on different metrics including diversity and quality of solutions, inverted generational difference and spacing of Pareto points on the Pareto fronts. The hybridization approach gives better results and number of Pareto points as compared to the other algorithms for medium and large size test problems. Similar results are obtained for the other metrics. Their contribution considers that the actual processing time of jobs may be reduced due to the "learning effect" of manual workers performing repetitive production operations. However, the research does not consider more than one machine.

2.5 Recent work

As a state of the art, recent authors have used well-recognized and published methodologies with some modifications to solve the FJSP. Phanden and Jain [45] offer a simulation-based genetic algorithm approach to solve the FJSP-PPF and assess the effect of flexible process plan of a part-type in a production order on a makespan performance measure. Two case studies of varying sizes are considered to assess the effect of changing the flexible process plans of a part-type in a production order. For these case studies, a shop floor consists of 15 machines. The results indicate that selecting the best process plan among flexible process plans on the basis of a minimum total production time criterion for a part-type may not yield optimal schedule. However, no other algorithms are used for a comparative section.

Li and Gao [32] proposed an effective hybrid algorithm (HA) that hybridizes the genetic algorithm and tabu search, well-recognized and published methodologies for the FJSP with the objective of minimizing the makespan. In order to solve the FJSP effectively, effective encoding method, genetic operators and neighborhood structure are used in this study. As the authors explain, six famous benchmark set of instances (including 201 open problems) of FJSP have been used to evaluate the performance of the proposed HA. The set data were adopted from Fattahi et al. [16], Kacem et al. [26], Barnes and Chambers [4], Brandimarte [7], among others. Comparisons among proposed HA and other ten state-of-the-art reported algorithms are also provided to show the effectiveness and efficiency of the proposed method. From the results of each experiment, the proposed HA can obtain the best results for most

problems. In addition, some of them are new solutions to some benchmark problems which were not previously found. Also, the computational time of the proposed HA has been compared with other algorithms, and the proposed HA seems to cost less computational time than other algorithms. However, scheduling problems in the manufacturing field are neither the aim of this research nor the multi-objective perspective.

Xu et al. [53] established a mathematical model for the FJSP-PPF, and an improved bat algorithm was proposed to solve the mathematical model mentioned. The objective was to seek an appropriate schedule that costs minimum time to complete all operations, i.e., the makespan. A crossover and a mutation operation, well-recognized and published operators, were designed to strengthen the searching ability of the algorithm. The authors carried out experiments on actual examples, i.e., this paper uses the actual data of a manufacturing enterprise. The data consists of 6 jobs of 27 operations that can be implemented on 8 machines. It can be seen from the experimental results that the robustness and optimization ability of the algorithm proposed are better than the genetic algorithm and the Discrete Particle Swarm Optimization algorithm (DPSO) used for comparison. For the purposes of overcoming the shortcomings of the fixed parameters in bat algorithm, the value of the inertia weight was adjusted, and a linear decreasing inertia weight strategy was proposed. However, no standard instances were used in the comparative section.

Yin et al. [55] detailed a new low-carbon mathematical scheduling model for the flexible job-shop environment. The authors select productivity, energy consumption and noise emission as the three low-carbon scheduling optimization objectives. A multi-objective genetic algorithm based on a simplex lattice design was proposed to solve the mathematical model effectively. The well-recognized and published operators such as the corresponding encoding/decoding method, fitness function, and crossover/mutation operators were designed specifically for the features of this problem. Three problem instances with different scales and one engineering case study illustrate and evaluate the performance of this method. The instances mentioned considered 4 jobs x 5 machines, 10 jobs x 6 machines and 20 jobs x 5 machines respectively. The engineering case study is related to a cooling system. The cooling system is an important component of any automotive engine. This automotive part requires seven main steps to build a whole cooling system. The operations for each step of the process range from 13 to 15 tasks. Eight machines are available for execution of these operations. It is a classical FJSP configuration. To test the efficiency of the proposed algorithm, it was compared with the NSGA-II for the aforementioned cases. Based on the results the proposed algorithm outperforms NSGA-II with regard to all metrics. No standard instances were used in this research. It is because there is no information about energy consumption and noise emission for standard instances.

The review mentioned above clearly underlines several gaps in the actual state of the art. The current research mentioned above does contain greedy procedures to obtain promising solutions. The performance of the algorithms mentioned above is implicit in the greedy procedures. For example, the traditional evolutionary operators, used by the genetic algorithms, produce a new population entirely, i.e., performing two main processes (crossing and mutation) to different solutions (parents) in order to build a new offspring (child). The processes mentioned have an implicit contribution to the performance of the genetic algorithms. However, the works mentioned above do not permit to have control in characterizing the solution space explicitly. Therefore, the greedy procedures are replaced by an explicit probability distribution over the FJSP-PPF in this research. With this change, we have control to characterize the solution space explicitly. Although the EDA has successfully been used to solve complex combinatorial optimization problems (with or without hybridization with other algorithms), there are currently no publications that use an EDA for the FJSP-PPF. We can cite some studies such as Chen et al. [10–12], Liu et al. [34], Peña et al. [44], Jarboui et al. [25], Wang et al. [51], and Pan and Ruiz [42]. These studies have tackled different scheduling problems by means of EDA. A new research using EDA for scheduling is reported in Wang et al. [52]. The authors address distributed permutation flow shop scheduling problems by a fuzzy logic-based hybrid EDA. In order to explore more promising search space, the proposed algorithm hybridizes the probabilistic model of EDA with crossover and mutation operators of a genetic algorithm to produce new offspring. The Mallows distribution is reported in Irurozki et al. [24]. The authors address the problems of sampling and learning of such distributions using the Cayley distance as a distance metric between permutations. Finally, Table 2 depicts the state of the art on the FJSP-PPF with other related problems.

3 MEDA – for the FJSP-PPF

A feasible solution representation for the FJSP-PPF uses three different vectors. The first vector is named the process plan vector: its length equals the total of jobs, where each element shows the corresponding chosen process plan for each job. The second vector is named the operation sequence vector: its length equals the total number of operations, where every element contains a non-repeated integer value, i.e. the vector is a permutation-based representation. The third vector is named the machine assignment vector: each element represents the

Table 2 State of the art for the scheduling problems

Current Research	Features Configuration style	Type of instance	Type of solution	Type of Algorithm	Hybridization	Objective	Approach
Yin et al. [55]	FJSP	Real scenarios	no optimal	Genetic Algorithm	–	Productivity Energy consumption Noise emission	Industrial
Xu et al. [53]	FJSP-PPF	Real scenarios	no optimal	Bat Algorithm	–	Makespan	Industrial
Dauod et al. [14]	FJSP	Real scenarios	no optimal	Genetic Algorithm	–	Makespan Collation days	Industrial
Huang et al. [23]	FJSP	Benchmark scenarios	optimal	Particle Swarm Optimization	Variable Neighborhood Search	Makespan Total workload Critical machine workload	Academic
Li and Gao [32]	FJSP	Benchmark scenarios	optimal	Genetic Algorithm	Tabu Search	Makespan	Academic
Yue et al. [56]	Single machine	Real scenarios	no optimal	Artificial Bee Colony Algorithm	Pareto	Makespan Total weighted tardiness	Industrial
Phanden and Jain [45]	FJSP-PPF	Real scenarios	no optimal	Genetic Algorithm	–	Makespan	Industrial
Wang et al. (2015)	PFSP	Hypothetical scenarios	no optimal	Estimation of Distribution Algorithm	Genetic Algorithm Fuzzy Logic Event-Discrete Simulation	Makespan	Academic
Chen et al. [10]	PFSP	Benchmark scenarios	no optimal	Estimation of Distribution Algorithm	Genetic Algorithm	Makespan	Academic
Chen et al. [10]	PFSP	Benchmark scenarios	no optimal	Estimation of Distribution Algorithm	Genetic Algorithm Variable Neighborhood Search	Makespan	Academic
Özgüven et al. [41]	FJSP-PPF	Hypothetical scenarios	no optimal	Mixed Integer Goal Programming	–	Makespan Total workload	Academic
Pan and Ruiz [42]	PFSP	Hypothetical scenarios	no optimal	Estimation of Distribution Algorithm	Local Search	Makespan	Academic
Wang et al. [51]	FJSP	Benchmark scenarios	optimal no optimal	Estimation of Distribution Algorithm	Local Search	Makespan	Academic
Liu et al. [34]	PFSP	Benchmark scenarios	optimal	Particle Swarm Optimization	Estimation of Distribution Algorithm	Makespan	Academic

Table 2 (continued)

Current Research	Features Configuration style	Type of instance	Type of solution	Type of Algorithm	Hybridization	Objective	Approach
Chen et al. [11]	Single machine	Benchmark scenarios	no optimal	Estimation of Distribution Algorithm	–	Earliness Tardiness	Academic
Özgüven et al. [40]	FJSP-PPF	Hypothetical scenarios	optimal	Mixed Integer Goal Programming	–	Makespan	Academic
Jarboui et al. [25]	PFSP	Benchmark scenarios	no optimal	Estimation of Distribution Algorithm	Variable Neighborhood Search	Total flowtime	Academic
Gao et al. [20]	FJSP	Benchmark scenarios	optimal	Genetic Algorithm	Variable Neighborhood Descent	Makespan Total workload Maximal machine workload	Academic
Rossi and Dini [46]	FJSP	Benchmark scenarios	no optimal	Ant Colony Optimization	–	Makespan	Academic
Park and Choi [43]	FJSP	Real scenarios Benchmark scenarios	no optimal	Genetic Algorithm	–	Makespan	Industrial
Peña et al. [44]	Different problems	Hypothetical scenarios Benchmark scenarios	optimal	Genetic Algorithm	Estimation of Distribution Algorithm	Different objectives	Academic
Kim et al. [28]	FJSP	Hypothetical scenarios	no optimal	Symbiotic Evolutionary Algorithm	–	Total absolute deviation of machine loads Makespan Mean flow time	Academic
Kacem et al. [26]	FJSP	Hypothetical scenarios	no optimal	Evolutionary Algorithm	Pareto Fuzzy Logic	Makespan Total workload Maximal machine workload	Academic
Lee et al. [31]	FJSP	Hypothetical scenarios	no optimal	Genetic Algorithm	–	Makespan	Industrial

Table 3 An instance of the FJSP-PPF

Alternative process plans for jobs					Alternatives machines for operations and processing times		
Jobs	Process plans	Operations	Operation sequence		Operations	Machines	Processing times
1	2	3	$3 \prec 1 \prec 2$	$O_{13} - O_{11} - O_{12}$	1	3	98
			$2 \prec 1 \prec 3$	$O_{12} - O_{11} - O_{13}$		1	17
2	3	4	$3 \prec 1 \prec 4 \prec 2$	$O_{23} - O_{21} - O_{24} - O_{22}$		2	48
			$4 \prec 2 \prec 3 \prec 1$	$O_{24} - O_{22} - O_{23} - O_{21}$			
			$1 \prec 3 \prec 2 \prec 4$	$O_{21} - O_{23} - O_{22} - O_{24}$	2	2	12
3	4	4	$1 \prec 2 \prec 3 \prec 4$	$O_{31} - O_{32} - O_{33} - O_{34}$		3	62
			$4 \prec 3 \prec 2 \prec 1$	$O_{34} - O_{33} - O_{32} - O_{31}$			
			$1 \prec 3 \prec 2 \prec 4$	$O_{31} - O_{33} - O_{32} - O_{34}$	3	1	90
			$3 \prec 4 \prec 1 \prec 2$	$O_{33} - O_{34} - O_{31} - O_{32}$		2	9
4	2	2	$1 \prec 2$	$O_{41} - O_{42}$			
			$2 \prec 1$	$O_{42} - O_{41}$	4	1	68
5	2	2	$1 \prec 2$	$O_{41} - O_{42}$			
			$2 \prec 1$	$O_{42} - O_{41}$			
6	2	3	$3 \prec 1 \prec 2$	$O_{13} - O_{11} - O_{12}$			
			$2 \prec 1 \prec 3$	$O_{12} - O_{11} - O_{13}$			

corresponding selected machine for each operation. To explain the representation, we provide an example by considering a problem with at most 4 process plans, 6 jobs and 3 machines as shown in Table 3. The example mentioned above is an extension of the example shown in Table 1 and Fig. 2. Figure 4 illustrates the representation of a feasible solution.

The solution representation mentioned above, depicted in Fig. 4 starts with the process plan vector. The mentioned vector is located in the upper left corner of Fig. 4. Each element shows the corresponding chosen process plan for each job, i.e., $P = \{1\ 3\ 2\ 1\ 1\ 2\}$. Based on Table 3, $P(1) = 1$ corresponding to operation sequence $3 \prec 1 \prec 2$. These three operations, from job number one, correspond to the

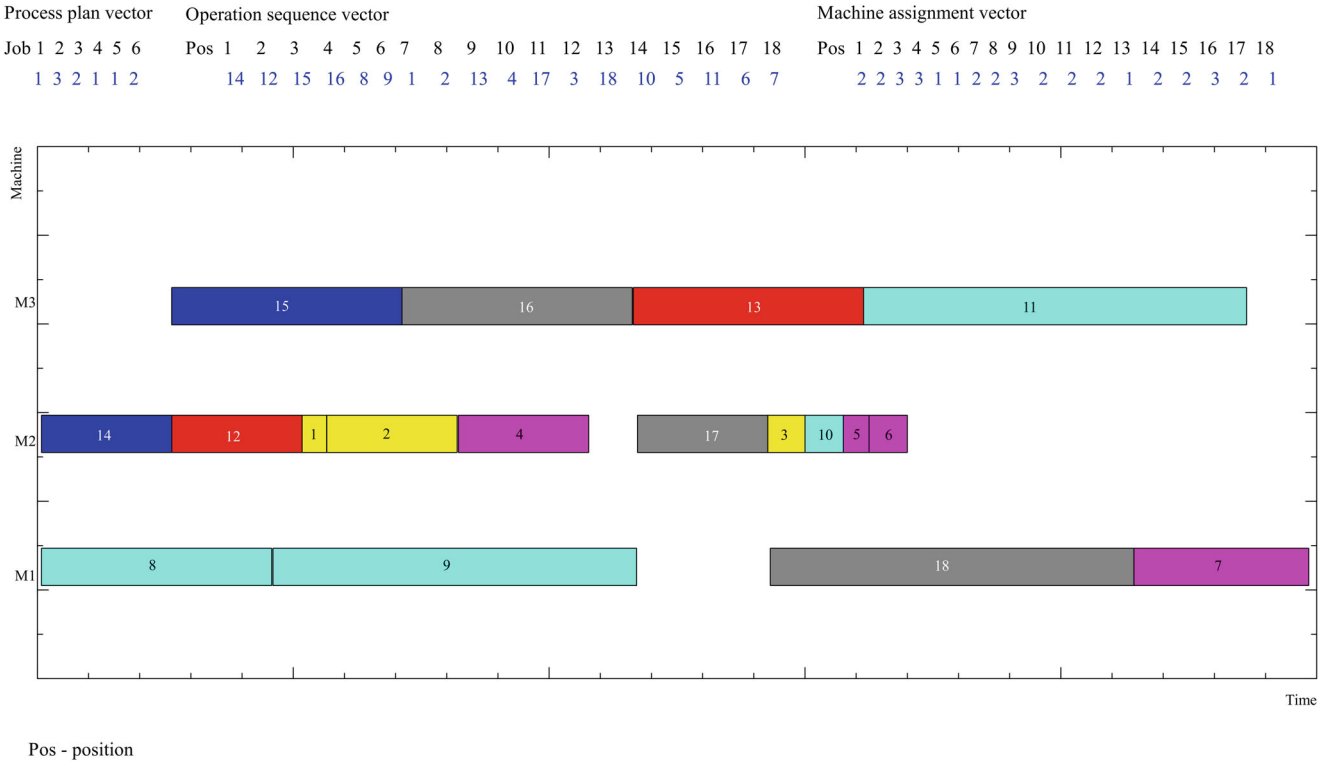


Fig. 4 Illustration of the representation of a feasible solution

Fig. 5 **a** A sample of process plan vectors. **b** A sample of operation sequence vectors. **c** A sample of machine assignment vectors

a

Pseudocode 1(a). Population initialization for process plan vectors

```
n ← jobs
M ← Individuals
for i := 1 to M
        for j := 1 to n
                Process plan vector [i][j] = choose randomly a possible process plan for the job j
        endfor
endfor
```

Population initialization
Process plan vectors

2	1	2	2	1	2
2	2	2	1	1	2
1	3	2	2	2	2
2	2	4	2	1	2
2	3	1	2	2	1
2	3	3	2	1	2
2	3	3	1	2	1
1	3	3	1	2	2
2	2	4	1	1	2
1	2	3	2	1	1
Job 1	2	3	4	5	6

b

Pseudocode 1(b). Population initialization for operation sequence vectors

```
m ← total of operations
n ← jobs
M ← Individuals
for i := 1 to M
        for j := 1 to n
                total operations per job[j] = identify the number of operations for the job j
        endfor

        index := 1
        while (index ≤ m) do
                job = choose randomly a job
                if(total operations per job[job] > 0) then
                        //pending operations to be sequenced
                        tentative sequence vector[i][index] = job
                        total operations per job[job] = total operations per job[job] – 1
                        index = index + 1
                endif
        endwhile

        for j := 1 to n
                total operations per job[j] = identify the number of operations for the job j
        endfor

        operation := 1
        job := 1
        while(job ≤ n) do
                index := 1
                while(index ≤ m) do
                        if(tentative sequence vector[i][index] == job) then
                                if(total operations per job [job] > 0) then
                                        sequence vector[i][index] = operation
                                        total operation per job[job] = total operations per job[job] – 1
                                        operation = operation + 1
                                endif
                        endif
                        index = index + 1
                endwhile
                job = job + 1
        endwhile
endfor
```

Population initialization
Operation sequence vectors

	12	1	13	2	16	14	8	3	4	15	9	17	18	10	11	5	6	7
	4	5	8	14	15	9	10	12	1	2	3	11	16	6	17	13	7	18
	4	1	16	5	12	8	13	17	2	6	3	18	14	7	9	15	10	11
	14	12	16	13	1	15	4	5	2	6	7	17	18	3	8	9	10	11
	12	8	4	16	17	18	14	5	15	6	7	1	2	9	3	10	13	11
	14	8	1	9	12	15	4	10	16	11	13	5	2	17	18	3	6	7
	4	12	14	8	13	9	16	17	1	15	2	10	18	3	11	5	6	7
	14	1	4	12	5	8	9	6	16	2	13	10	15	11	3	17	7	18
	4	1	5	6	12	13	8	9	10	16	14	15	11	2	7	17	18	3
	1	2	3	4	12	5	14	16	17	8	9	18	6	7	10	15	11	13
index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

Fig. 5 (continued)

c

Pseudocode 1(c). Population initialization for machine assignment vectors

```
m ← total of operations
M ← Individuals
for i ≔ 1 to M
        for k ≔ 1 to m
                //choose randomly a possible machine for the operation at position k from operation sequence vector
                Machine assignment vector [i][j] =  choose randomly a possible machine for the operation at index k
        endfor
endfor
```

Population initialization
Machine assignment vectors

	3	2	1	3	3	2	1	1	1	2	1	1	1	3	3	2	1	3
	1	2	1	2	3	2	2	2	2	1	1	2	2	2	2	3	2	1
	1	1	3	1	3	1	3	2	3	3	2	1	2	1	2	1	3	1
	3	3	2	3	3	3	1	2	2	2	1	3	2	1	2	1	1	2
	3	3	1	2	3	3	3	2	1	2	1	3	2	3	1	1	2	1
	1	2	3	1	3	3	2	3	2	1	2	2	1	3	1	1	2	1
	1	3	2	2	3	2	1	1	3	1	1	2	2	1	1	1	3	1
	2	1	3	3	2	2	1	2	2	3	3	3	3	1	3	3	1	1
	1	2	3	2	1	3	1	1	2	3	3	3	2	3	1	1	2	2
	2	1	2	1	3	3	2	2	3	3	2	2	1	2	3	3	1	2
index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

values $1-2-3$ in the operation sequence solution vector σ. $P(2) = 3$ corresponds to operation sequence $1 \prec 3 \prec 2 \prec 4$. These four operations, from job number two, correspond to the values $4 - 5 - 6 - 7$ in the operation sequence solution vector σ, and so on. Although each element in σ can be located in any position along the solution vector, as any permutation-based representation, the precedence between operations of each job must be kept in σ. For example, the values $1 - 2 - 3$, from job number one, must appear in the same order in σ, from left to right. If the vector solution of the operation sequence satisfies the precedence mentioned above, it satisfies the corresponding original operation sequence from the job mentioned. This representation is based on Gen et al. [21]. Therefore, the operation sequence vector located next to the process plan vector of Fig. 4 contains non-repeated integer values and is feasible, i.e., $\sigma = \{14\ 12\ 15\ 16\ 8\ 9\ 1\ 2\ 13\ 4\ 17\ 3\ 18\ 10\ 5\ 11\ 6\ 7\}$. Finally, the machine assignment vector, located next to the operation sequence vector of Fig. 4, represents the corresponding selected machine for each operation, i.e., $A = \{2\ 2\ 3\ 3\ 1\ 1\ 2\ 2\ 3\ 2\ 2\ 2\ 1\ 2\ 2\ 3\ 2\ 1\}$. Based on Table 3, $A(1) = 2$ considers using machine number two, for the first value in σ, i.e., $\sigma\ (1) = 14$. The value 14 in $\sigma\ (1)$ represents the first operation of job number five.

The initialization of the population is implemented as an array of M vectors of size n jobs for the process plan vectors. Another array of M vectors of size m, where m represents the total number of operations to sequence for the operation sequence vectors and another array of M vectors of size m, where m represents the total number of operations to assign for the machine assignment vectors. Initial population members are generated randomly in order to enable a wide range of solutions [22]. Figure 5a shows different process plan vectors obtained with the corresponding Pseudocode 1(a). Figure 5b details different operation sequence vectors obtained with the corresponding Pseudocode 1(b). Finally, Fig. 5c depicts different machine assignment vectors obtained with the corresponding Pseudocode 1(c).

3.1 Pareto-front construction phase

In each generation, a Pareto-front approach is built with the population. A Pareto-front approach is considered in this research based on Kacem et al. [26]. The optimality notion in the Pareto approaches can be formulated as follows: the Pareto-optimal set is formed by non-dominated solutions. x vector dominates y vector if $\forall 1 \leq q \leq L,\ f_q(x) \leq f_q(y)$ where $f_q\ (.)$ is the qth objective function, L is the number of objective functions and at least one index r exists such that $f_r\ (x) < f_r(y)$. A solution is non-dominated if it is not dominated by any other solution. Figure 6a details the fitness of two feasible solutions, i.e., the makespan and the maximal workload. Figure 6b depicts a Pareto-optimality approach example. A non-dominated set of solutions in each generation is found and used for the selection process, as well as for the improvement of the central permutation estimate σ_0 in the GMD process.

The selection process takes the subset N from M parents (where $N < M$) are chosen by a tournament selection which is executed based on where the candidate solutions are located on the Pareto-front approach. Different cases are possible as follows: if a solution is feasible and another is

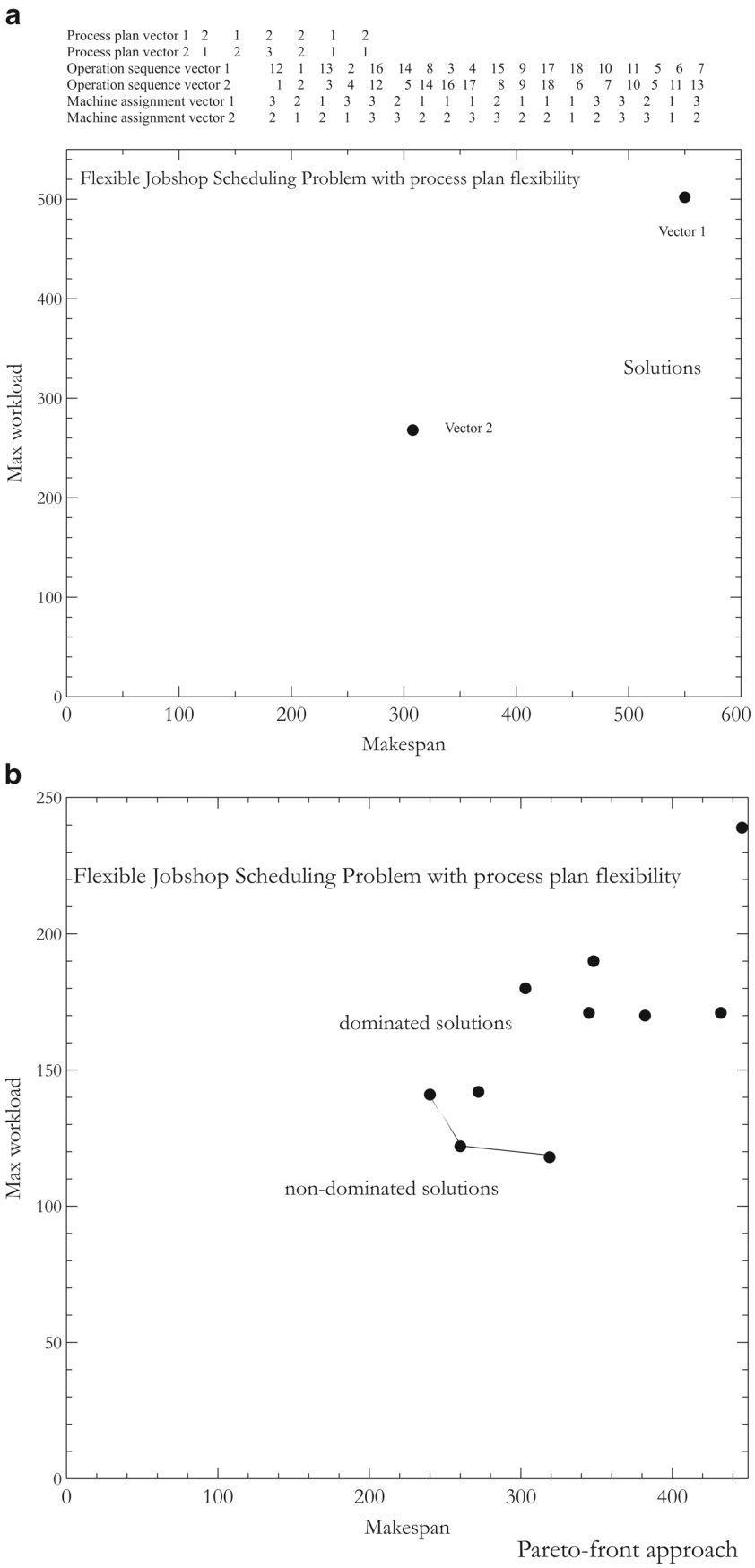


Fig. 6 **a** A graphical representation fitness of two feasible solutions. **b** Pareto-optimality approach

infeasible, the feasible solution is preferable. If a solution is non-dominated and another is dominated, the non-dominated solution is preferable. If both solutions are non-dominated, the solution that belongs to a better Pareto-front layer is preferable.

In order to determine which Pareto-front layer belongs to each non-dominated solution, we calculate two entities: 1) domination count n_p, the number of solutions which dominate the solution p, and 2) S_p, a set of solutions that the solution p dominates. All solutions in the first non-dominated front will have their domination count as zero. Now, for each solution p with $n_p = 0$ we visit each member (q) of its set S_p and reduce its domination count by one. In doing so, if for any member q the domination count becomes zero, we put it on a separate list Q. These members belong to the second non-dominated front. Now, the above procedure is continued with each member and the third front is identified. This process continues until all fronts are identified.

3.2 Probability model proposed

3.2.1 Probability model for process plan vectors

Unlike GA which produces offspring through crossover and mutation operators, EDA does it by sampling according to a probability model. Therefore, the probability model has a great effect on the performance of EDA. MEDA contains three probability models. The first probability model aims to determine an estimate of a distribution model

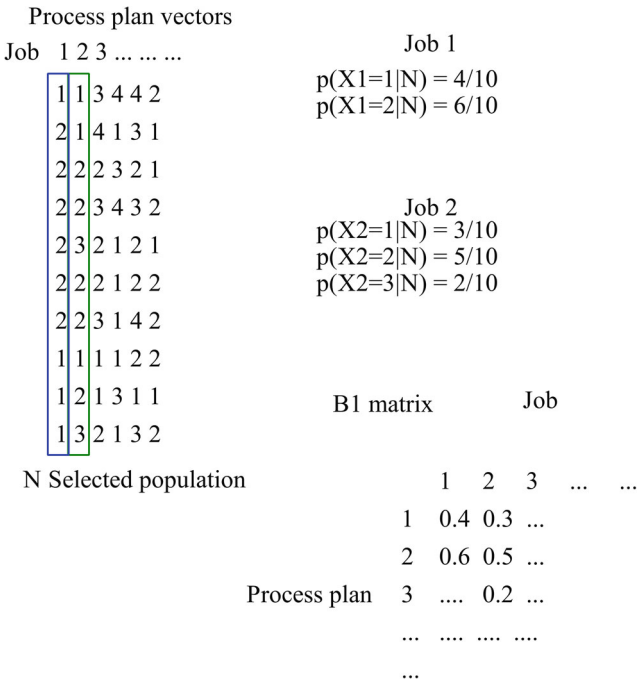


Fig. 7 Building an estimation of distribution model for the process plan vectors

to generate new offspring (process plan vectors) using a subset of N chosen process plan vectors in the previous step. To obtain the estimate we use the UMDA algorithm introduced by Mühlenbein [38]. The algorithm mentioned uses the simplest approach to estimate the joint probability distribution of the selected individuals in each generation, $p(\mathbf{x})$. This joint probability distribution is factorized as a product of univariate marginal distributions, $p(\mathbf{x}) = p(x|N) = \prod_{i=1}^{n} p(x_i)$.

Every univariate marginal distribution is estimated from marginal frequencies, $p(x_i) = \frac{\sum_{j=1}^{N} \delta_j(X_i = x_i|N)}{N}$ where $\delta_j(X_i = x_i \mid N) = \begin{cases} 1, & \text{if the jth case of N, } X_i = x_i \\ 0, & \text{otherwise} \end{cases}$

Figure 7 contains an example to realize how the probability distribution is built in a subset of N process plan vectors. The element $p_{kj}(l)$ of the process plan probability matrix called B1 matrix represents the probability that process plan k be selected for the job j of the process plan vector at generation l. The value of p_{kj} refers to the importance of a process plan for a specific job. For all k ($k = 1, 2, \ldots, P_j$) and j ($j = 1, 2, \ldots, n$)

3.2.2 Probability model for operation sequence vectors

The second probability model aims to determine an estimate of an explicit probability distribution in the domain of permutations by the GMD to generate new offspring (operation sequence vectors) using a subset of N previous sequence vectors. The GMD can be used to solve permutation-based optimization problems. Formally, the Mallows model is defined as:

$$P(\sigma) = \psi(\theta)^{-1} exp(-\theta D(\sigma, \sigma_0)) \quad (1)$$

where θ is a spread parameter, and $D(\sigma, \sigma_0)$ is the distance from σ to the central permutation σ_0, where $\psi(\theta)$ is a normalization constant. In the present work, Kendall's -τ is the distance metric with which the Mallows model is coupled. Let's consider σ_0 = {1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18} as a central permutation, i.e., a feasible operation sequence vector. The number of elements in σ_0 is based on the detailed example in Table 3 and Fig. 4. Let σ_1 = {14 12 15 16 8 9 1 2 13 4 17 3 18 10 5 11 6 7} as another feasible operation sequence vector with $D(\sigma_1, \sigma_0) = 86$. Let σ_2 = {14 15 16 8 9 1 2 4 12 13 17 3 18 10 5 11 6 7} with $D(\sigma_2, \sigma_0) = 82$.

Based on Mallows model, every permutation σ_i gets a probability that decays exponentially with respect to its distance to the central permutation σ_0. Therefore, σ_1 has less probability than σ_2 even though both permutations are feasible and these were selected on previous step. Figure 8 depicts a visual representation of the previously detailed example.

The GMD decomposes Kendall's -τ distance metric into $m - 1$ terms in such a way that it can be expressed as

$D(\sigma,\ \sigma_0) = D_\tau(\sigma,\ \sigma_0) = \sum_{q=1}^{m-1} V_q(\sigma,\ \sigma_0)$. The GMD is given as follows

$$P(\sigma) = \psi(\theta)^{-1} exp\left(\sum_{q=1}^{m-1} -\theta_q V_q(\sigma,\ \sigma_0)\right) \quad (2)$$

where $V_q(\sigma,\ \sigma_0)$ is the number of positions on the right of q with values smaller than the current position on the permutation (σ, σ_0). Based on the example above for σ_1, the $V_q(\sigma_1,\ \sigma_0)$ for the qth position is detailed below:

qth	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	$\sim m-1$
σ_1	14	12	15	16	8	9	1	2	13	4	17	3	18	10	5	11	6	7
$V_q(\sigma_1,\ \sigma_0)$	13	11	12	12	7	7	0	0	7	1	6	0	5	3	0	2	0	$\sim \sum 86$

The same procedure to identify the Kendall's -τ distance metric in σ_2, i.e., the $V_q(\sigma_2,\ \sigma_0)$ for the qth position is detailed below:

qth	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	$\sim m-1$
σ_2	14	15	16	8	9	1	2	4	12	13	17	3	18	10	5	11	6	7
$V_q(\sigma_2,\ \sigma_0)$	13	13	13	7	7	0	0	1	6	6	6	0	5	3	0	2	0	$\sim \sum 82$

Instead of a single spread parameter θ as in the Mallows distribution, the GMD uses $m-1$ spread parameters $\theta = (\theta_1, \theta_2, \ldots, \theta_{m-1})$, every θ_q affecting a particular position q of the permutation. Based on Fligner and Verducci [19], it is possible to determine any permutation (σ, σ_0) with the $m-1$ integers $V_1(\sigma, \sigma_0), \ldots, V_{m-1}(\sigma, \sigma_0)$ in which Kendall's-τdistance $D_\tau(\sigma, \sigma_0)$ decomposes. Under the uniform distribution, the $V_q(\sigma, \sigma_0)$ variables that define a permutation are independent [18] and as a consequence, the probability distribution of the random variables $V_q(\sigma, \sigma_0)$ under the GMD given by (2) can be written as:

$$P\left(V_q(\sigma, \sigma_0) = r_q\right) = \frac{exp\left(-\theta_q r_q\right)}{\psi_q\left(\theta_q\right)} \qquad r_q \in \{0, \ldots, m-q \quad (3)$$

The normalization constant $\psi(\theta)$ in the GMD can be simplified as the product of $m-1$ terms

$$\psi(\theta) = \prod_{q=1}^{m-1} \psi_q(\theta_q) = \prod_{q=1}^{m-1} \frac{1 - exp(-\theta_q(m-q+1))}{1 - exp(-\theta_q)} \quad (4)$$

where m is the total of items in the permutation. When the GMD considers Kendall's-τ distance, it can be expressed as a multi-stage ranking model [19]. Therefore, the probability distribution of a given permutation σ is given as

$$P(\sigma) = \prod_{q=1}^{m-1} P(V_q(\sigma, \sigma_0) = r_q) \quad (5)$$

where r_q is a possible value for the q position on the V_q.

The first stage consists of computing an approximation of the central permutation σ_0. Although the behavior of the GMD depends on the spread vector θ that determines the shape of the distribution Ceberio et al. [8], future research work is aimed at improving the estimation of the central permutation σ_0 to enhance the performance of the algorithm. To avoid losing diversity through the modification in the central permutation is the goal. In this research, when a non-dominated set of solutions is found in the Pareto-front approach mentioned above, the corresponding operation sequence vectors of the non-dominated set of solutions are used as an input parameter by the Borda algorithm [6] to get a better approximation to the central permutation σ_0. To the best of our knowledge, a modification on the parameter described has not been developed for these kind of algorithms. The following procedure details an example in order to estimate the central permutation σ_0 with the Borda algorithm using the corresponding operation sequence vectors of the non-dominated set of solutions

Pseudocode 2 Central permutation computation σ_0

```
m ← number of positions (total operations)
M ← Individuals
from the non − dominated set of permutations {σ_1, ..., σ_M}
for j := 1 to m
        Compute π(j) = Σ_{i=1}^{M} σ_i(j)/M
Endfor

visited = {∅}, auxiliary array
for j := 1 to m
        Find the lowest value from π(i)
                index ← arg min_i {π(i) | i ∉ visited}
        Building σ_0 ...
                σ_0(index) = j
        visited = visited ∪ index
Endfor
```

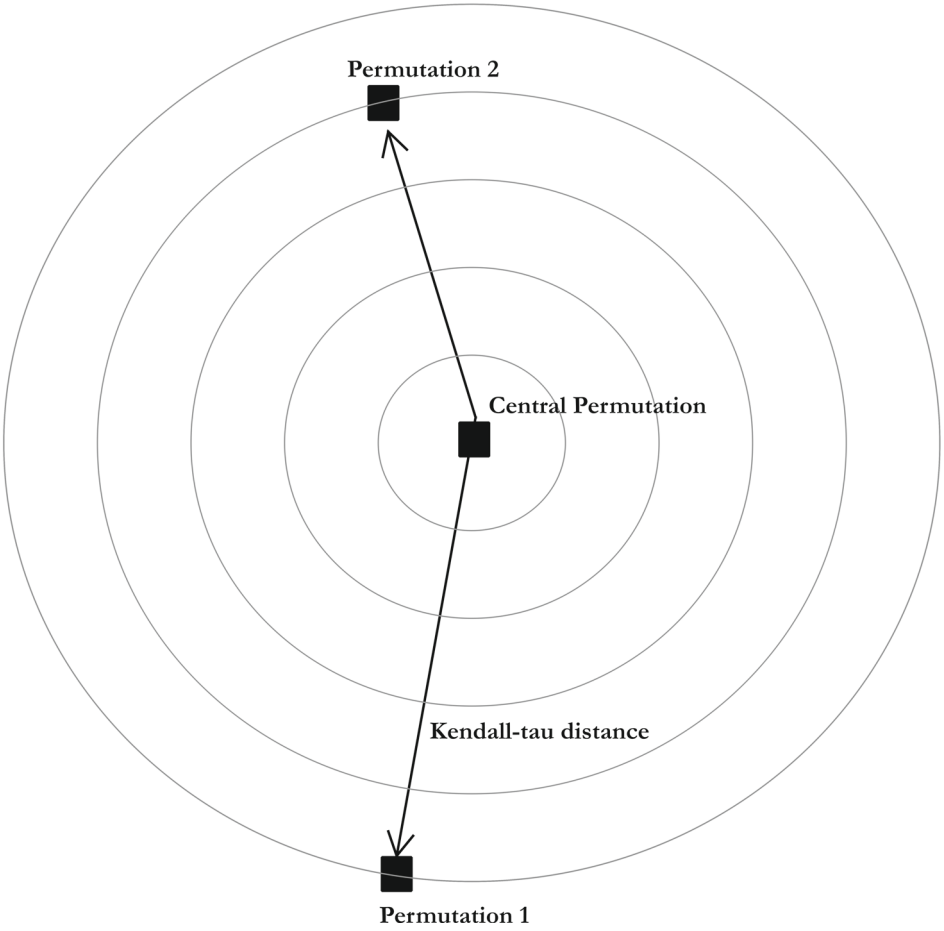
Once the central permutation σ_0 is approximated using the Pareto-front approach, the second stage consists of computing the dispersion parameters θ_j by solving

$$\overline{V}_j = \frac{1}{exp(\theta_q) - 1} - \frac{m-q+1}{\exp\left(\theta_q(m-q+1)\right) - 1},\ q = 1 : m-1 \quad (6)$$

where $\overline{V}_q = \frac{1}{N} \sum_{i=1}^{N} V_q(\sigma_i \sigma_0)$

Central Permutation	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
Permutation 1	14	12	15	16	8	9	1	2	13	4	17	3	18	10	5	11	6	7	Kendall-tau distance = 86
Permutation 2	14	15	16	8	9	1	2	4	12	13	17	3	18	10	5	11	6	7	Kendall-tau distance = 82

Fig. 8 A visual representation of the Kendall's -τ distance metric



With the spread parameter vector generated θ and the central permutation estimated σ_0, it is possible to create the operations sequence probability matrix called B2 to obtain new $\mathbf{V}(\sigma, \sigma_0)$ vectors by (3). The element $p_{ijq}(l)$ of the operation sequence probability matrix B2 represents the probability that operation $O_{i,j}$ requires r number of adjacent transpositions required to match the operation $O_{i,j}$ in the same qth position as the central permutation σ_0 at generation l.

Figure 9 depicts a visual example for only two positions on the sequence, in the operation sequence probability matrix B2. Based on the previous example above, the r_q possible values for a new $\mathbf{V}(\sigma, \sigma_0)$ vector will be between 0 and $m - q$, where q represents the positions on the operation sequence.

For each qth position, the probability for each r_q value i.e., p_{ijq}, refers to the importance of an operation for a specific position of the permutation-based representation. For all $(i = 1, 2, \ldots, n_i)$, $j\,(j = 1, 2, \ldots, n)$ and $q\,(q = 1, 2, \ldots, m - 1)$.

In this way, this research characterizes the space of possible solutions explicitly through the estimation of the probability that an operation for a specific position of the permutation-based representation is selected.

The process of generating the new valid permutations (operations sequence vectors) is decoding the $\mathbf{V}(\sigma, \sigma_0)$ vectors. This research uses the Meilă et al. [37] algorithm for transforming the vectors mentioned above in valid permutations. The following procedure represents an example to get a valid permutation.

Pseudocode 3 Meilă et al. (2007) procedure

m ← number of positions (total operations)
Let a sample vector $\mathbf{V}(\sigma, \sigma_0) = 2, 0, 1$
therefore m := 4
insert m in $0 \rightarrow (\sigma, \sigma_0)^{-1} = 4$
insert j := 3 in $V_3 := 1 \rightarrow (\sigma, \sigma_0)^{-1} = 4, 3$
insert j := 2 in $V_2 := 0 \rightarrow (\sigma, \sigma_0)^{-1} = 2, 4, 3$
insert j := 1 in $V_1 := 2 \rightarrow (\sigma, \sigma_0)^{-1} = 2, 4, 1, 3$

Every permutation σ is obtained by inverting a composing with the central permutation σ_0, $((\sigma, \sigma_0)^{-1})^{-1}\sigma_0 = \sigma\sigma_0^{-1}\sigma_0 = \sigma$

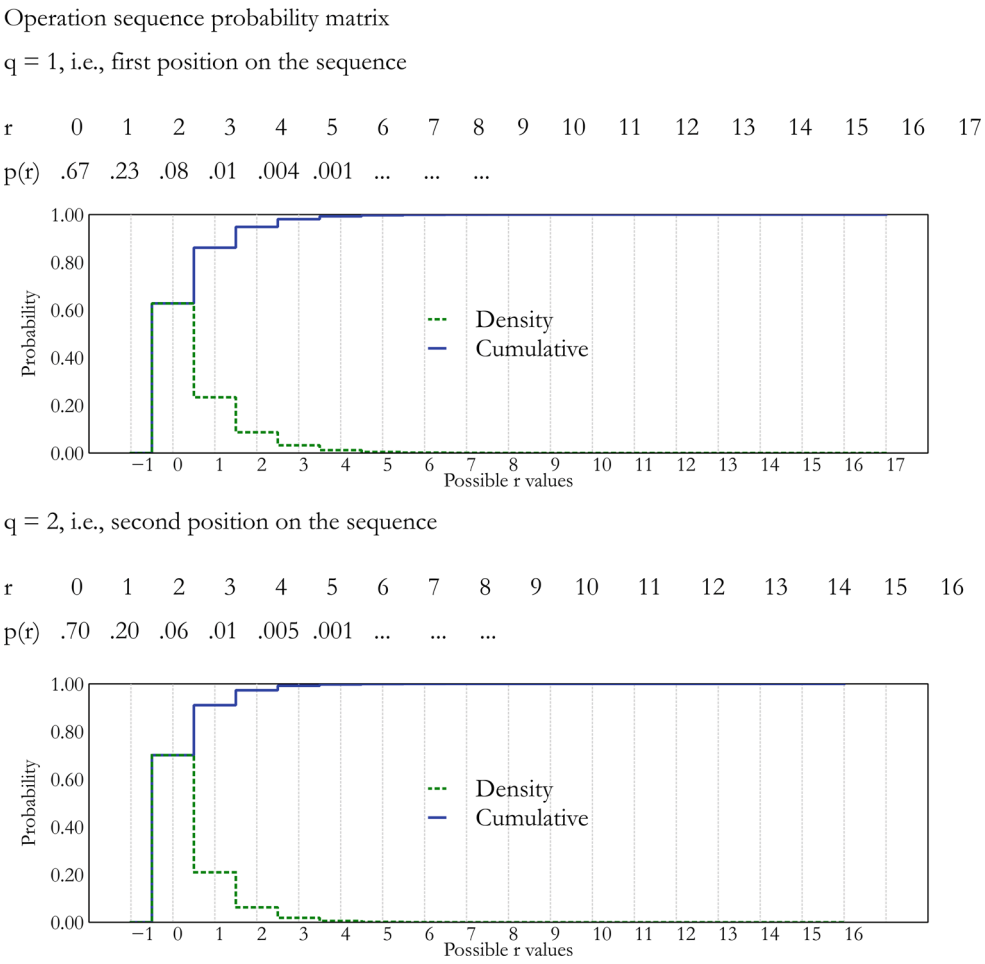


Fig. 9 An operation sequence probability matrix B2

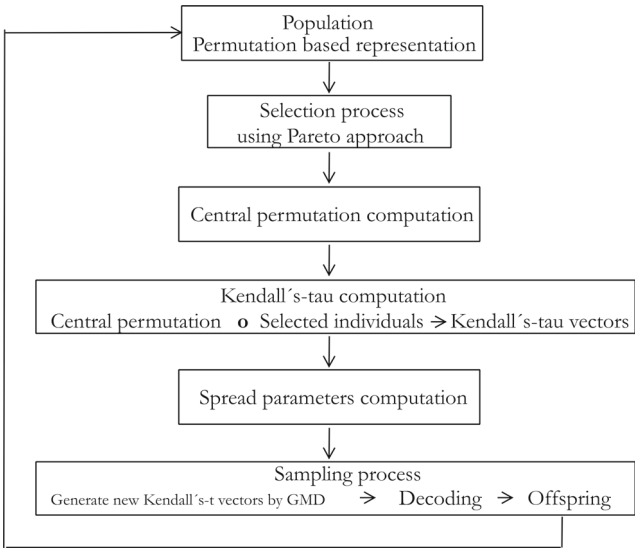


Fig. 10 GMD process for the FJSP-PPF problem

Finally, Fig. 10 details the overall process for GMD process mentioned above.

3.2.3 Probability model for machine assignment vectors

The third probability model aims to determine an estimate of a distribution model to generate new offspring (machine assignment vectors) using a subset of N selected assignments. To obtain the estimate we also use the UMDA algorithm. The element $p_{ijm}(l)$ of the machine probability matrix called B3 represents the probability that operation $O_{i,j}$ is processed on machine m at generation l. The value of p_{ijm} indicates the rationality of an operation processed on a certain machine.

3.3 Sampling process

The sampling process is made according to the probability matrices B1, B2 and B3. New promising solutions may be generated. In particular, to generate a new solution

Table 4 Comparison of results with standard benchmarking datasets

Intervals Instances	Trials	[0 , 0.051)	[0.051 , 0.101)	[0.101 , 0.151)	[0.151 , 0.201)	[0.201,0.251)	[0.251 ,0.301)	Outliers
abz	150	118	19	7	6	0	0	9
ft	90	67	19	3	1	0	0	2
la	1200	986	200	14	0	0	0	5
orb	300	229	66	5	0	0	0	0
swv	600	595	5	0	0	0	0	0
yn	120	111	9	0	0	0	0	0

the process plan vector should be generated first, then the operation sequence vector and finally the machine assignment vector. The MEDA procedure to solve the FJSP-PPF is illustrated below

Pseudocode 4 MEDA framework for the FJSP-PPF

$D_0 \leftarrow$ Generate M individuals
t ≔ 1
Do
  $D_{t-1} \leftarrow$ Evaluate individuals (makespan and max workload)
  $P_{t-1} \leftarrow$ Compute Pareto front from D_{t-1}
  $Ds_{t-1} \leftarrow$ Select N individuals from Ds_{t-1}
  Compute B1 matrix from Ds_{t-1}
  Compute B2 matrix from Ds_{t-1}
    Estimate central permutation σ_0 from P_{t-1}
    Estimate spread parameters $\boldsymbol{\theta}$ from Ds_{t-1} and σ_0
  Compute B3 matrix from Ds_{t-1}
  $Ds_t \leftarrow$ Sample M individuals from B1, B2, B3, matrices
  $D_t \leftarrow$ Find best individuals from $Ds_{t-1} \cup Ds_t$
  t ≔ t + 1
Until (stopping criterion is met)

4 Computational results and comparison

4.1 Comparison with standard benchmarking datasets

In order to validate the scientific relevance of this paper, we compare the MEDA results with others in some general and standard benchmarking datasets such as the Adams, Balas and Zawack [1] instances, called abz, taking five instances from abz5 to abz9 specifically; the Fisher and Thompson [17] instances, called ft, taking three instances, ft06, ft10, and ft20; the Lawrence [30] instances, called la, taking forty instances from la01 to la40; the Applegate and Cook [3] instances, called orb, taking ten instances from orb01 to orb10; the Storer, Wu and Vaccari [48] instances, called swv, taking twenty instances from swv01 to swv20; finally, the Yamada and Nakano [54] instances, called yn, taking four instances from yn1 to yn4.

The instances mentioned above are used as input data for the mentioned comparison. The experiments are executed in a Lanix® Titan HX 4200 computer, Intel® Core™ i7 processor, 3.4 GHz, 8 GB of RAM, Windows® 10 for 64 bits to run every instance. C++ language is used for the implementation for all the comparisons. To account for the stochastic nature of the MEDA, we run 30 trials for all the datasets. Each trial contains 50 generations, 1000

Table 5 A systematical approach used for determining an instance of the FJSP-PPF

Number of jobs (n)	Randomly [2, 50]
Number of process plans (p)	Randomly [2, 4]
Number of machines (m)	Randomly [2, 10]
Number of operations n_i for the job i	Randomly [2, 10]
m_i (machine for the n_i operation)	Randomly [1, m]
processing times	Randomly [1, 99]

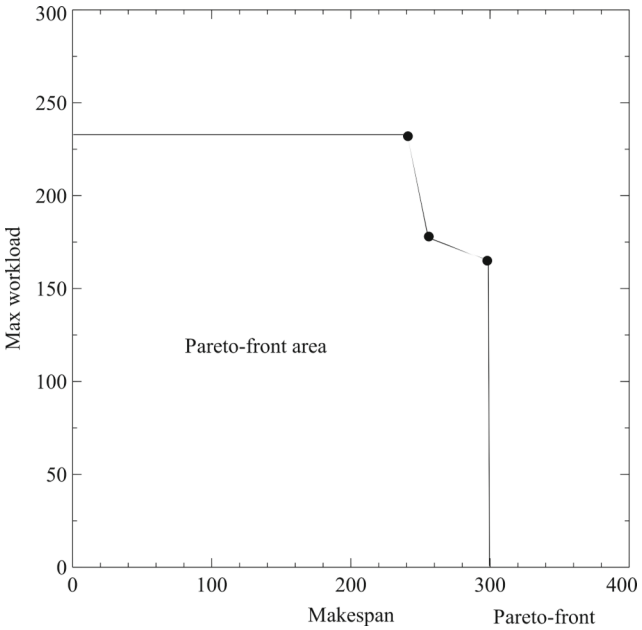


Fig. 11 First Pareto-front of the seventh instance obtained by the MEDA for the FJSP-PPF

Table 6 Comparison of the results

Size of the instance				
Algorithm	Small	Medium mean(std)	Large	Global performance
NSGA	1.727(3.838)	0.520(0.405)	0.404(0.283)	0.777(2.056)
NSGA-II	1.597(3.420)	0.513(0.419)	0.408(0.314)	0.744(1.841)
w-EDA	0.973(1.442)	0.391(0.268)	0.290(0.220)	0.493(0.816)
MEDA for the FJSP-PPF	0.457(0.384)	0.332(0.266)	0.232(0.161)	0.315(0.278)

solution vectors belonging to each generation. We measure the relative percentage increase (RPI) as:

$$\mathrm{RPI}\ (c_i) = \left(c_i - c^*\right)/c^* \tag{7}$$

where c_i is the makespan obtained in the ith replication, and c^* is the best makespan found and reported in the literature. The distribution of the experimental results in every interval is presented in Table 4. It is clear from the table that the results of the MEDA algorithm are comparatively concentrated, which is mainly in the range of [0, 0.05], over the best solution found, i.e., makespan, for all the datasets. In addition, Table 4 describes the outliers, i.e., deteriorations produced by MEDA. Based on the results the MEDA is a suitable approach for solving real flexible manufacturing processes.

4.2 Comparison with multi-objective algorithms

Furthermore, in order to enhance the novelty of the MEDA, we consider evaluating the MEDA with the same characteristics of the industrial environment. A hundred flexible job-shop instances have been developed replicating real flexible manufacturing processes, to analyze the solution capability of the MEDA on the FJSP-PPF. The instances contain a different number of jobs, machines, operations, process plans for each job and alternative machines for performing each operation.

An estimation of distribution algorithm coupled with the GMD process without using the Pareto-front approach, called w-EDA, is proposed as a benchmark for comparison with the MEDA scheme. In addition, the multiobjective algorithms proposed by Srinivas and Deb [47] called NSGA

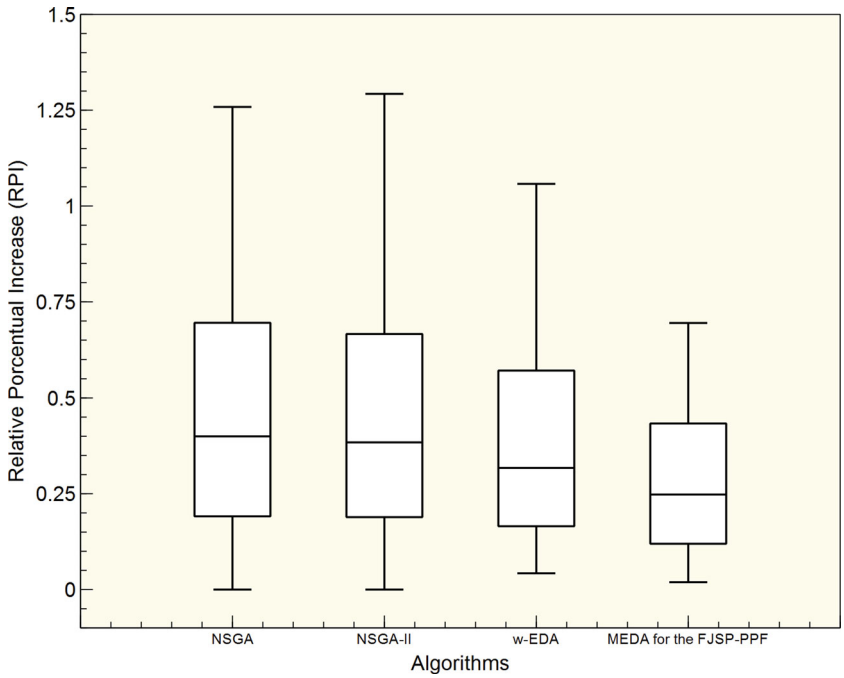


Fig. 12 Comparison of results using boxplots

Table 7 Distribution of the results

Intervals Algorithm	[0 , 3)	[3 , 6)	[6 , 9)	[9 , 12)	[12 , 15)	[15 , 18)	[18 , 21)	[21 , 24)	[24 , 27)
NSGA	2914	28	12	23	8	1	2	11	1
NSGA-II	2909	39	10	21	9	0	9	0	3
w-EDA	2970	17	6	6	1	0	0	0	0
MEDA for the FJSP-PPF	3000	0	0	0	0	0	0	0	0

and by Deb et al. [15] called NSGA-II are proposed as a benchmark for comparison with the MEDA scheme. The code of the proposed algorithms are obtained from the Kanpur Genetic Algorithms Laboratory web. The instances mentioned above are used as input data for the algorithms. The experiments are executed in the same computer and language specification. To account for the stochastic nature of the shop floor, we run the same number of trials, i.e., 30 for all the algorithms. Each trial contains the same number of generations, i.e., 50, and the same size of the population, i.e., 1000 solution vectors. Table 5 presents the systematical approach used for determining the jobs, machines, operations, process plans for each job, alternative machines for operations and processing times of operations for each job in order to replicate this experiment and get similar results. Although the systematical approach mentioned above is detailed in Table 5, the instances are available at https://goo.gl/1JeHDr

Two conflicting objectives are analyzed; the makespan and the maximal machine workload, i.e., the maximum working time spent on any machine. The aim is to prevent a solution from assigning too much work on a single machine and to keep the balance of work distribution over the machines. The makespan and the maximal machine workload are conflicting objectives in nature when there exist dominant machines with regard to processing times of the operations. In addition, while the number of available machines in the process is greater, the average utilization of every machine is lower. Therefore, the aim is to determine the balance between both objectives. The objectives mentioned above are used as input data to find the response variable for the experiment, i.e., we compute the

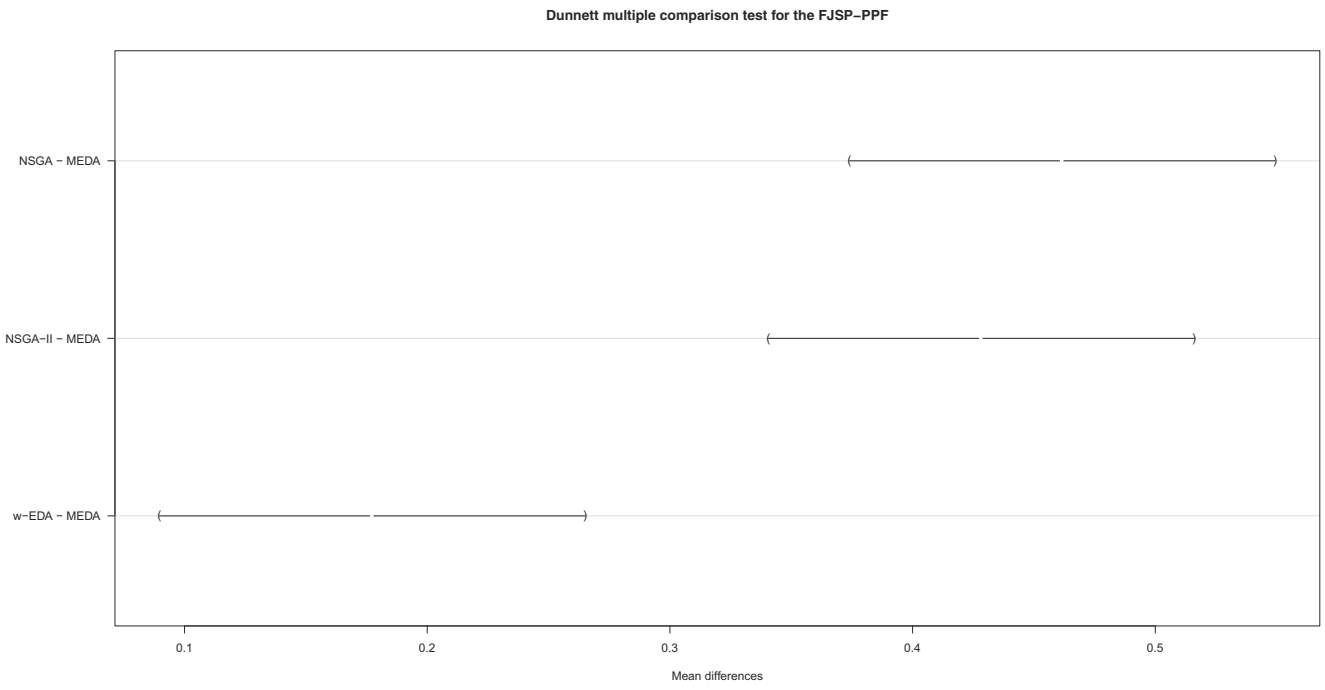


Fig. 13 Statistical test

Table 8 Component parts of an automotive instrument panel

Part number	Part name
1	Crash pad main assembly
2	Side facia panel
3	Crash pad defroster nozzle
4	Glove box mounting bracket
5	Glove box upper cover
6	Glove box
7	Cluster assembly
8	Cluster facia panel
9	Center facia panel support bracket
10	Lower crash pad center facia panel
11	Ash tray
12	Steering column shroud
13	Lower crash pad support bracket
14	Crash pad lower cover bracket
15	Crash pad lower cover

area obtained from the best Pareto-front after running every algorithm. As an example, Fig. 11 shows the area computed by the first Pareto-front in the seventh instance.

We measure the relative percentage increase (RPI) using the same (7), i.e., c_i is the area obtained from the best Pareto-front obtained in the ith replication by a given algorithm configuration, and c^* is the best area found by any of the algorithm configurations. Note that for this problem, there are no known effective or exact techniques and comparison with an optimal solution is not possible. Table 6 depicts the mean and standard deviation metrics for all the algorithms used for the comparison of over 30 independent runs per instance.

Figure 12 includes four box plots: the NSGA, the NSGA-II, the MEDA, and the w-EDA performance after running all the instances. As we can see, the EDA coupled with the GMD process without using the Pareto-front approach,

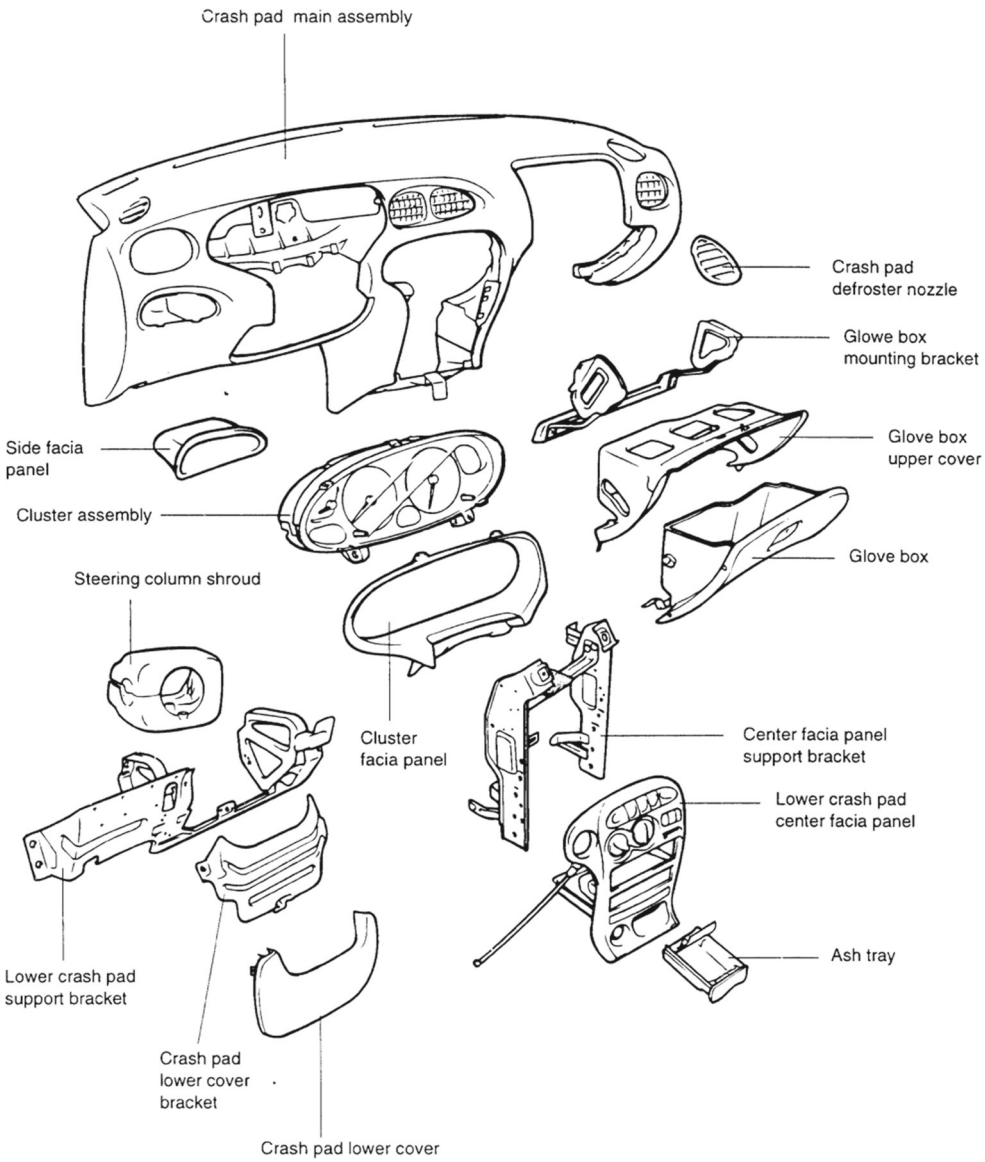


Fig. 14 Instrument panel manufacturing

Table 9 Comparison of results between recent algorithms

Size of the production order				
Algorithm	Small	Medium mean(std)	Large	Global performance
Phanden and Jain [45]	1.412(1.746)	0.576(0.336)	0.528(0.814)	0.770(1.136)
Xu et al. [53]	1.478(1.888)	0.575(0.344)	0.538(0.874)	0.792(1.223)
MEDA for the FJSP-PPF	1.466(1.825)	0.572(0.344)	0.545(0.949)	0.791(1.223)
Li and Gao [32]	1.511(1.907)	0.594(0.342)	0.536(0.898)	0.804(1.243)
Li et al. [33]	1.488(1.892)	0.578(0.339)	0.542(0.895)	0.797(1.233)

Table 10 Distribution of the results for the instrument panel manufacturing case

Intervals Algorithm	[0 , 2)	[2 , 4)	[4 , 6)	[6 , 8)	[8 , 10)	[10 , 12)	[12 , 14)
Phanden and Jain [45]	2814	128	29	18	2	3	6
Xu et al. [53]	2822	106	32	24	6	5	5
MEDA for the FJSP-PPF	2812	107	38	30	3	8	2
Li and Gao [32]	2803	124	32	24	6	7	4
Li et al. [33]	2799	127	31	24	9	7	3

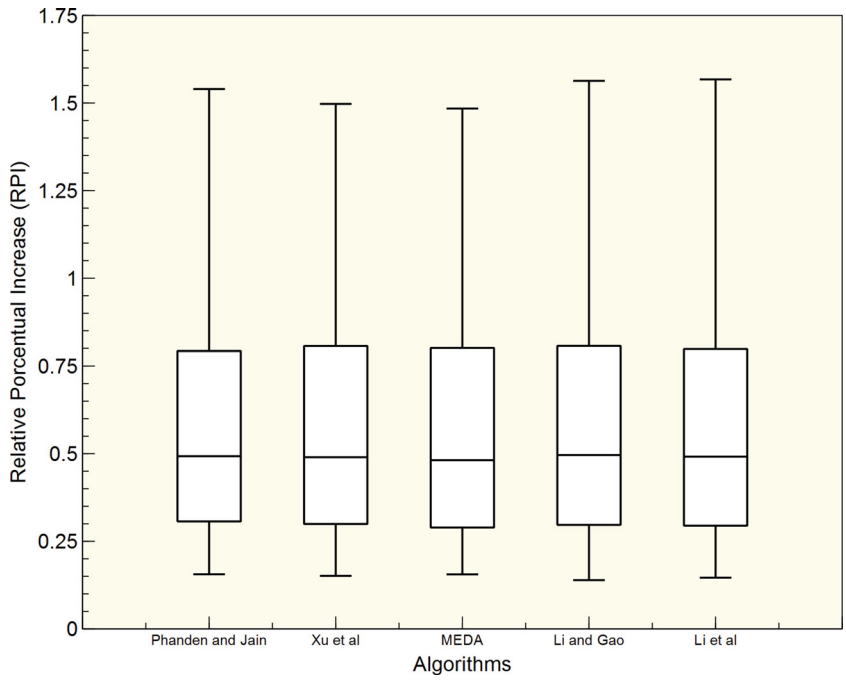


Fig. 15 Comparison of results between recent algorithms using boxplots

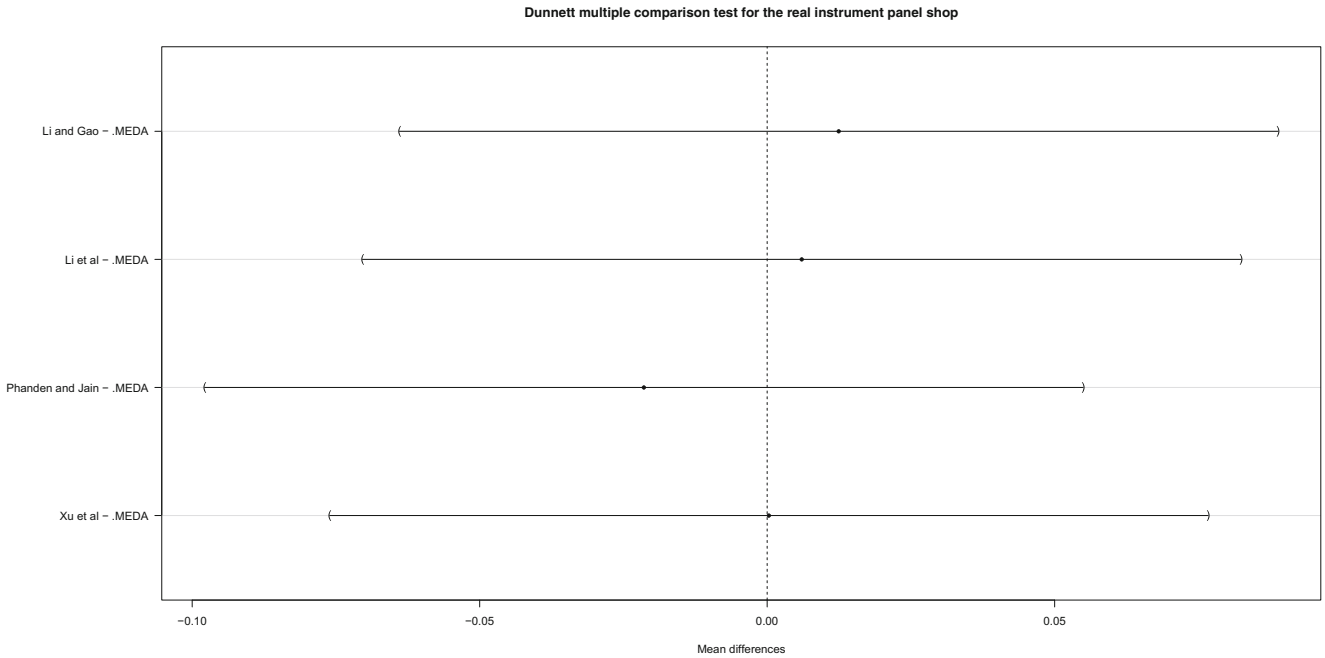


Fig. 16 The Dunnett test

i.e., the w-EDA, is competitive in order to identify the best-performing ones. Although the GMD process is enough to find suitable solutions, the Pareto-approach helps to improve the results when the non-dominated solutions are considered in the GMD process. The MEDA results report how the performance of the algorithm is improved.

The distribution of the experimental results in each interval is presented in Table 7. It is clear from the table that the results of the MEDA algorithm are comparatively concentrated, which is mainly in the range of [0, 3], whereas the results of the other algorithms are relatively distributed. In addition, we analyze whether there is a statistically significant difference between the averages of the algorithms. Figure 13 depicts the statistical test, i.e., the Dunnett test. As we can see, the proposed MEDA is competitive. There is a difference between the MEDA with

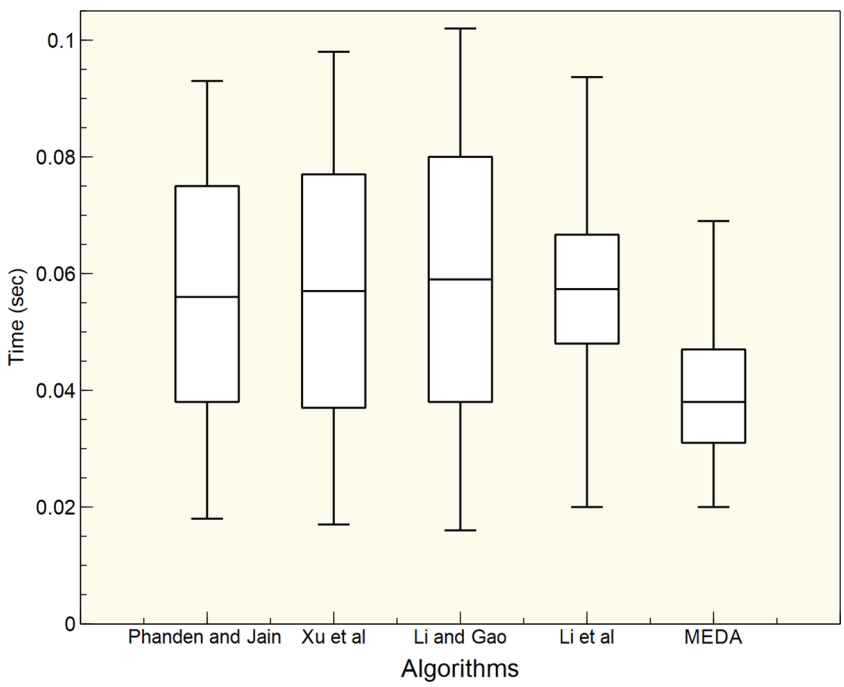


Fig. 17 Computational time comparative

the other algorithms for the FJSP-PPF. We confirm that the hybridization between the GMD process and the Pareto-front approach is suitable for the EDA scheme. Thus, the MEDA algorithm in terms of solving the FJSP-PPF is more robust than the other algorithms.

4.3 Comparison with recent algorithms for the FJSP

Based on the previous results in the last comparison mentioned above, recent algorithms are proposed as a benchmark for comparison with the MEDA scheme. Because MEDA is built to be functional in flexible manufacturing processes, the recent algorithms mentioned above are the simulation-based genetic algorithm presented by Phanden and Jain [45], the improved bat algorithm proposed by Xu et al. [53], the hybrid algorithm that hybridizes the genetic algorithm and tabu search detailed by Li and Gao [32], and the genetic algorithm designed by Li et al. [33]. All the algorithms mentioned are considered recent algorithms for the FJSP. These algorithms have been implemented by the authors. The experiments are executed with the same parameters and specification detailed above. Furthermore, a real-life problem is considered for comparison with the MEDA scheme. It is related to the instrument panel manufacturing. An automotive supplier located in the middle of Mexico receives different automotive parts for assembling instrument panels every day. Those parts are enlisted in Table 8. All the parts mentioned are depicted in Fig. 14.

Multiple machines are available to process all the parts, and the processing times vary for different machines and operating procedures. As we can see in Fig. 14, there is more than one process plan for the instrument panel manufacturing, i.e., the main assemblies (cluster panel, facia panel, glove box, and lower cover) can be built independently between them. Thus, the shop floor represents a typical flexible job-shop with process plan flexibility. We established a workload to evaluate and find the best schedule. Our experiments are based on the production of 10,000 assemblies. The workload mentioned contains different orders, due dates, and the kinds of instrument panels produced in the manufacturing process mentioned above, replicating the actual manufacturing process.

As in the previous comparison, we measure the relative percentage increase (RPI) detailed in (7). Table 9 describes the mean and standard deviation metrics for the recent algorithms used for the comparison of over 30 independent runs per instance. Based on Table 9, there is no significant difference between the performance of the algorithms, i.e, the global performance is the same. The differences are only in thousandths. Therefore, the MEDA is competitive in the same industrial environment.

The distribution of the experimental results in each interval is presented in Table 10. It is clear from the table that the results of all the algorithms are relatively distributed at the same intervals. Moreover, most results are concentrated in the first interval for whatever algorithm. The difference between the worst and the best performance is only 0.007%, i.e., 2799/3000 vs 2822/3000. The MEDA is a suitable approach for solving real flexible manufacturing processes.

Figure 15 includes five boxplots: the Phanden and Jain [45] algorithm, the Xu et al. [53] algorithm, the MEDA, the Li and Gao [32] algorithm, and the Li et al. [33] algorithm performance after running all the instances. As we can see, the performance of the MEDA is competitive in the same characteristics of the industrial environment. We emphasize that the real problem used for analyzing the solution capability of the MEDA on the FJSP-PPF replicates a real flexible manufacturing process.

In addition, we analyze whether there is a statistically significant difference between the averages of the algorithms. Figure 16 depicts the statistical test, i.e., the Dunnett test. As we can see, the proposed MEDA is competitive and there is no difference between the algorithms for the FJSP-PPF.

Finally, the computational time comparison is shown in Fig. 17.

5 Conclusions and future research

This paper discusses the FJSP-PPF, which considers multiple process plans for jobs by excluding the assumption of only one feasible process plan for each job of FJSP. To solve this problem, we propose the EDA application scheme. By means of numerical experiments, this approach generates competitive solutions.

This novel research concludes that the GMD can be coupled with the EDA scheme in order to solve combinatorial optimization problems such as the FJSP-PPF from a multi-objective perspective. By using the Pareto-front approach it is possible to obtain better solutions for the FJSP-PPF. The computational results establish that the different probability models used for the FJSP-PPF with large data sets are suitable. The Pareto-front approach is not only used for obtaining the best solutions for the MEDA scheme, it is also used to propose a different way of improving the estimation of the central permutation σ_0 in the GMD process. The MEDA provides an effective estimate of the operations at the positions in the sequence vector for the FJSP-PPF.

Explicit probability distributions in the domain of permutations for the FJSP-PPF should be considered in future research. Estimating relationships for other related

dynamic job-shop issues, e.g., shutdown, maintenance, new products, and online environments, should be reconsidered in light of these results. Since the MEDA presents stability, it appears very suitable for implementation in software systems for practical purposes. Further research directions may deal with an extension of the MEDA for building effective modules for specific users in the industry. Finally, the MEDA might be used in other types of job-shop systems.

Acknowledgments We would like to express our gratitude to all the reviewers for their comments in improving the manuscript.

References

1. Adams J, Balas E, Zawack D (1988) The shifting bottleneck procedure for job shop scheduling. Manag Sci 34(3):391–401
2. Ali A, Meilă M (2012) Experiments with Kemeny ranking: what works when? Math Soc Sci 64(1):28–40
3. Applegate D, Cook W (1991) A computational study of the job-shop scheduling problem. ORSA J Comput 3(2):149–156
4. Barnes JW, Chambers JB (1996) Flexible job shop scheduling by tabu search. Graduate Program in Operations and Industrial Engineering, The University of Texas at Austin, Technical Report Series, ORP96-09
5. Bartholdi J, Tovey CA, Trick MA (1989) Voting schemes for which it can be difficult to tell who won the election. Soc Choice Welf 6(2):157–165
6. Borda JD (1784) Mémoire sur les élections au scrutin, vol 1781. Histoire de l'Academie Royale des Sciences pour, Paris
7. Brandimarte P (1993) Routing and scheduling in a flexible job shop by tabu search. Ann Oper Res 41(3):157–183
8. Ceberio J, Irurozki E, Mendiburu A, Lozano JA (2014) A distance-based ranking model estimation of distribution algorithm for the flowshop scheduling problem. IEEE Trans Evol Comput 18(2):286–300
9. Chambers JB, Barnes JW (1996) New tabu search results for the job shop scheduling problem. The University of Texas, Austin, Technical Report Series ORP96-06, Graduate Program in Operations Research and Industrial Engineering
10. Chen SH, Chang PC, Cheng TCE, Zhang Q (2012) A self-guided genetic algorithm for permutation flowshop scheduling problems. Comput Oper Res 39(7):1450–1457
11. Chen SH, Chen MC, Chang PC, Zhang Q, Chen YM (2010) Guidelines for developing effective estimation of distribution algorithms in solving single machine scheduling problems. Expert Syst Appl 37(9):6441–6451
12. Chen YM, Chen MC, Chang PC, Chen SH (2012) Extended artificial chromosomes genetic algorithm for permutation flowshop scheduling problems. Comput Ind Eng 62(2):536–545
13. Cohen WW, Schapire RE, Singer Y (1998) Learning to order things. J Artif Intell Res 10:243–270
14. Dauod H, Li D, Yoon SW, Srihari K (2016) Multi-objective optimization of the order scheduling problem in mail-order pharmacy automation systems. Int J Adv Manuf Technol:1–11, https://doi.org/10.1007/s00170-016-9123-1
15. Deb K, Pratap A, Agarwal S, Meyarivan TAMT (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197
16. Fattahi P, Mehrabad MS, Jolai F (2007) Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. J Intell Manuf 18(3):331
17. Fisher H, Thompson GL (1963) Probabilistic learning combinations of local job-shop scheduling rules. In: Muth JF, Thompson GL (eds) Industrial Scheduling. Prentice Hall, Englewood Cliffs, pp 225–251
18. Fligner MA, Verducci JS (1986) Distance based ranking models. J R Stat Soc Ser B Methodol 48(3):359–369. http://www.jstor.org/stable/2345433
19. Fligner MA, Verducci JS (1988) Multistage ranking models. J Amer Stat Assoc 83(403):892–901
20. Gao J, Sun L, Gen M (2008) A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. Comput Oper Res 35(9):2892–2907
21. Gen M, Tsujimura Y, Kubota E (1994) Solving job-shop scheduling problems by genetic algorithm. In: 1994 IEEE International Conference on Systems, Man, and Cybernetics, 1994. Humans, Information and Technology, vol 2. IEEE, pp 1577–1582
22. Greenwood AG, Vanguri S, Eksioglu B, Jain P, Hill TW, Miller JW, Walden CT (2005) Simulation optimization decision support system for ship panel shop operations. In: Proceedings of the 37th conference on Winter simulation. Winter Simulation Conference, pp 2078–2086
23. Huang S, Tian N, Wang Y, Ji Z (2016) Multi-objective flexible job-shop scheduling problem using modified discrete particle swarm optimization. SpringerPlus 5(1):1432
24. Irurozki E, Calvo B, Lozano JA (2014) Sampling and learning Mallows and Generalized Mallows models under the Cayley distance. Methodol Comput Appl Probab 20(1):1–35. https://doi.org/10.1007/s11009-016-9506-7
25. Jarboui B, Eddaly M, Siarry P (2009) An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems. Comput Oper Res 36(9):2638–2646
26. Kacem I, Hammadi S, Borne P (2002) Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. Math Comput Simul 60(3):245–276
27. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, vol 1000, p IV
28. Kim YK, Park K, Ko J (2003) A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling. Comput Oper Res 30(8):1151–1171
29. Larrañaga P, Lozano JA (eds) (2001) Estimation of distribution algorithms: a new tool for evolutionary computation, vol 2. Springer Science and Business Media, Berlin
30. Lawrence S (1984) Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (supplement). Graduate School of Industrial Administration. Carnegie-Mellon University, Pittsburgh
31. Lee YH, Jeong CS, Moon C (2002) Advanced planning and scheduling with outsourcing in manufacturing supply chain. Comput Ind Eng 43(1):351–374
32. Li X, Gao L (2016) An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. Int J Prod Econ 174:93–110
33. Li X, Xing K, Wu Y, Wang X, Luo J (2017) Total energy consumption optimization via genetic algorithm in flexible manufacturing systems. Comput Ind Eng 104:188–200
34. Liu H, Gao L, Pan Q (2011) A hybrid particle swarm optimization with estimation of distribution algorithm for solving permutation flowshop scheduling problem. Expert Syst Appl 38(4):4348–4360

35. Mallows CL (1957) Non-null ranking models. I. Biometrika 44(1/2):114–130
36. Meilă M, Phadnis K, Patterson A, Bilmes J (2012) Consensus ranking under the exponential model. arXiv:1206.5265
37. Meilă M, Phadnis K, Patterson A, Bilmes J (2007) Consensus ranking under the exponential model. In: Proceedings of the 22nd Conference Uncertainty and Artificial Intelligence, Vancouver, pp 285–294
38. Mühlenbein H (1997) The equation for response to selection and its use for prediction. Evol Comput 5(3):303–346
39. Mühlenbein H, Paaß G (1996) From recombination of genes to the estimation of distributions. I binary parameters. In: Proceedings of the 4th International Conference on Parallel Problem Solving from Nature. Springer, Berlin, p 187
40. Özgüven C, Özbakr L, Yavuz Y (2010) Mathematical models for job-shop scheduling problems with routing and process plan flexibility. Appl Math Modell 34(6):1539–1548
41. Özgüven C, Yavuz Y, Özbakr L (2012) Mixed integer goal programming models for the flexible job-shop scheduling problems with separable and non-separable sequence dependent setup times. Appl Math Modell 36(2):846–858
42. Pan QK, Ruiz R (2012) An estimation of distribution algorithm for lot-streaming flow shop problems with setup times. Omega 40(2):166–180
43. Park BJ, Choi HR (2006) A genetic algorithm for integration of process planning and scheduling in a job shop. In: Australian conference on artificial intelligence, pp 647–657
44. Peña JM, Robles V, Larranaga P, Herves V, Rosales F, Pérez MS (2004) GA-EDA: Hybrid evolutionary algorithm using genetic and estimation of distribution algorithms. In: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems. Springer, Berlin, pp 361–371
45. Phanden RK, Jain A (2015) Assessment of makespan performance for flexible process plans in job shop scheduling. IFAC-PapersOnLine 48(3):1948–1953
46. Rossi A, Dini G (2007) Flexible job-shop scheduling with routing flexibility and separable setup times using ant colony optimisation method. Robot Comput-Integr Manuf 23(5):503–516
47. Srinivas N, Deb K (1994) Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evol Comput 2(3):221–248
48. Storer RH, Wu SD, Vaccari R (1992) New search spaces for sequencing problems with application to job shop scheduling. Manag Sci 38(10):1495–1509
49. Sundaram RM, Fu SS (1988) Process planning and scheduling. Comput Ind Eng 15:296–307
50. Tan W (1998) Integration of process planning and scheduling-a mathematical programming approach. University of Southern California, USA
51. Wang L, Wang S, Xu Y, Zhou G, Liu M (2012) A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem. Comput Ind Eng 62(4):917–926
52. Wang K, Huang Y, Qin H (2016) A fuzzy logic-based hybrid estimation of distribution algorithm for distributed permutation flowshop scheduling problems under machine breakdown. J Oper Res Soc 67(1):68–82
53. Xu H, Bao ZR, Zhang T (2017) Solving dual flexible job-shop scheduling problem using a Bat Algorithm. Adv Prod Eng Manag 12(1):5
54. Yamada T, Nakano R (1992) A genetic algorithm applicable to large-scale job-shop problems. In: PPSN, vol 2, pp 281–290
55. Yin L, Li X, Gao L, Lu C, Zhang Z (2017) A novel mathematical model and multi-objective method for the low-carbon flexible job shop scheduling problem. Sustain Comput: Inform Syst 13:15–30
56. Yue L, Guan Z, Saif U, Zhang F, Wang H (2016) Hybrid Pareto artificial bee colony algorithm for multi-objective single machine group scheduling problem with sequence-dependent setup times and learning effects. SpringerPlus 5(1):1593
57. Zitzler E, Laumanns M, Thiele L (2001) SPEA2: Improving the strength Pareto evolutionary algorithm