Appl Intell (2018) 48:1423–1441
DOI 10.1007/s10489-017-0993-1

An effective operations permutation-based discrete harmony search approach for the flexible job shop scheduling problem with makespan criterion

Mehdi Gaham[1,2] · **Brahim Bouzouia**[1] · **Nouara Achour**[2]

Published online: 16 August 2017

Abstract The Flexible Job Shop Scheduling Problem (FJSSP) represents a challenging applicative problem for metaheuristic algorithms because it imposes the development of innovative domain-dependent search operators that have to deal both with its combined discrete and permutation nature. Emerging as an effective approach for the resolution of a broad spectrum of hard optimization problems, some few discrete declinations of the Harmony Search (HS) algorithm have been recently proposed for tackling the FJSSP. Recent advances include an investigation of an innovative and promising permutation-based proposal. Accordingly, this paper proposes an Effective Operations Permutation-based Discrete Harmony Search (EOP-DHS) approach for FJSSP with Makespan criterion. The approach adopts an integrated two-part "affectation-sequencing" representation of the solution harmony and a dedicated improvisation operator particularly adapted to the integer-valued and operations permutation-based used coding scheme. Besides, a Modified Intelligent Mutation (MIM) operator is integrated to the adopted framework in order to enhance its overall search ability. Mainly, by balancing maximum machine workload during the overall search process, MIM operator allows essentially maintaining and enhancing the reciprocal equilibrium of diversification and intensification abilities of the proposed EOP-DHS algorithm. Conducted numerical experimentations on 188 benchmarking instances validate the proposal comparatively to a representative set of previously deployed metaheuristic approaches to FJSSP with Makespan criterion. Furthermore, main contribution of the paper is extended with an experimental procedure proving the effectiveness of the adopted permutation-based HS scheme for the resolution of combinatorial optimization problems. Hard benchmarking instances of the classical Job Shop Scheduling Problem (JSSP) are thus considered for exemplification.

Keywords Flexible job shop · Job shop · Scheduling · Combinatorial optimization · Discrete harmony search · Permutation-based improvisation

✉ Mehdi Gaham
mgaham@cdta.dz

[1] Center of Development of Advanced Technologies, CDTA, SRP team, Baba Hassan, Algiers, Algeria

[2] USTHB University, LRPE Laboratory, BP 32, El Alia, Bab Ezzouar, Algiers, Algeria

1 Introduction

Scheduling problems arise in many management and engineering fields and are generally identified as belonging to the class of hard combinatorial optimization problems. One of those, the Job Shop Scheduling Problem (JSSP) describes a representative machine-scheduling problem where a set of independent jobs, each one constituted by an ordered set of operations, has to be executed on dedicated specialized machines while optimizing a predefined performance criterion. Mainly, the problem consists of finding the proper sequence of operations execution on machines considering imposed affectations. Formulated by Brucker and Schlie [1] as a generalization of JSSP, the Flexible Job Shop Scheduling Problem (FJSSP) relaxes affectation constraints described by JSSP and allows to operations to be executed by a machine chosen from a predefined set of allowable machines. Therefore, the resolution of the FJSSP implies both determination of a proper sequence of execution of operations on machines and a proper affectation of machines to operations. In accordance with its inherent complexity, FJSSP is classified as a harder generalization of the classical JSSP that is fundamentally known as NP-hard combinatorial optimization problem [2, 3].

Indeed, by integrating both sequencing and affectation problems, FJSSP is characterized by strong practical implications and an admitted academic relevance that is due to the fact that it incarnates a fair representation of the general domain of combinatorial optimization. Thus, developing effective resolution processes for FJSSP will allow the resolution of broad kinds of existing real-world combinatorial optimization problems. These facts motivated the utilization of a number of approaches for its resolution, ranging from exact mathematical, to approximation methods and computational intelligence tools. In [4] an exact mathematical and a heuristic approaches are considered for the resolution of FJSSP. Presented experimental and comparative results clearly indicate the ineffectiveness of the proposed mathematical approach for the resolution of medium size instances of the problem that consist of more than four jobs and five machines. Actually, due to their scalability issue, general deterministic methods are commonly admitted to be (being) unsuccessful for the resolution of large scale and realistic scheduling problems in an acceptable time and are generally considered for a small problem and academic studies.

Metaheuristic optimization methods and in line with their ability to efficiently cope with the strong combinatorial nature of hard optimization problems stay the most reexamined class of approximation approaches for the resolution of a wild spectrum of hard manufacturing scheduling problems. Historically, trajectory based methods such as Tabu Search (TS) algorithm initiated the application of metaheuristic framework to FJSSP and are contentiously investigated [5–9]. Evolutionary technics also maintain a permanent position in FJSSP literature with a number of effective and superior achievements [10–16]. Besides, different recent metaheuristic approach proposals for tackling the complexity of FJSSP exemplify perfectly its academic relevance. Such proposals include Biogeography-based optimization [17], Variable Neighborhood Search [18], Artificial Immune Algorithms [19], Estimation of distribution algorithms [20], artificial bee colony algorithms [21], Particle Swarm Approaches [22, 23], and Hybrid of Artificial Immune/Simulated Annealing approach [24]. Further readings concerning the development of the FJSSP and a consolidated survey of various techniques that have been employed since 1990 for problem resolution are proposed in [25]. Moreover, it is important to note at this stage that despite the fact of being a generalization of JSSP, FJSSP motivated a dedicated branch of research literature, generally unsuitable for the JSSP and where common JSSP instances are omitted from the computational experiments. Actually, the two problems are generally tackled independently with few exceptions such as in [26]. Regarding the basic JSSP, a recent review paper of the application of Artificial Intelligence (AI) technics to the problem [27] shows that as for FJSSP, a significant spectrum of metaheuristic approaches is constantly proposed for JSSP. Example of recently published proposals include novel declinations of Evolutionary Algorithms [28], multi-population ones [29, 30], Biogeography-based Algorithm [31], and Parallel Bat Algorithm [32].

Proposed by Geem [33], the Harmony Search (HS) metaheuristic algorithm, emerged in just more than one decade of research, as an effective approach for the resolution of a broad kind of hard optimization problems, including construction, engineering, robotics, telecommunications, health, and energy [34]. The HS algorithm simulates the improvisation process of music players and was originally conceptualized using the musical process of searching for a perfect state of harmony. Indeed, HS is a population-based algorithm that maintains and improves a population of solutions, the Harmonies Memory (HM). It uses a replacement policy and an improvisation operator that produces at each iteration a new harmony (solution vector) merging the information of different solutions stored in HM. Generally, the newly generated Harmony replaces the bad harmony in HM if it is better. Particularly, HS improvisation operator generates the New Harmony by using a recombination mechanism that acts independently on each component of the solution vector and considers all the solutions stored in HM. As reported in [34] and [35] this distinctive characteristic of the HS algorithm from other population-based metaheuristic approaches is one of its important innovative operational procedures. Furthermore, it is stressed in [34] that by playing a major role in achieving a good trade-off between diversification and intensification, the improvisation operator constitutes another remarkable strength of the HS algorithm. Also, numerical comparisons in [36, 37] demonstrated that evolution in the HS algorithm was faster than the canonical Genetic Algorithm (GA). These particular features in addition to its simplicity and ease of implementation motivated the deployment of HS algorithm for the resolution of a continuously increasing number of hard optimization problems leading to a number of improvements and adaptation mechanisms. In fact, recent literature (See [35, 38]) reports different improvements of HS algorithm that covered different aspects related to algorithms' parameters adaptation, improvisation mechanisms and integration within the HS process of other metaheuristic search components such as the genetic mutation operator [39, 40].

Accordingly, different declinations of HS algorithm have been recently proposed for the resolution of various manufacturing scheduling problems. The existing contributions can be broadly classified into two principal categories: "continuous-discrete domain mapping-based approaches" and "discreet domain approaches". Developments of the first class of approaches are motivated by the difficulty to adapt the continuous domain operators of HS algorithm to manufacturing scheduling problems, particularly those where the solution vector is coded using a discrete permutation-based form such as for JSSP and FJSSP. Indeed, approaches falling within that category use the canonical continuous-domain HS algorithm and a dedicated mapping scheme for the generation and the evaluation of the solution in the permutation-based native domain of the scheduling problem. Different recently published contributions for tackling various classes of flow shop scheduling problems [41–43] and parallel machine one [44] belong to that category. Other works falling into the second category of applicative methodologies of HS algorithm to scheduling problems propose a fundamental adaptation of HS operators to the targeted permutation-based representation domain. In [45] and [46], different discrete HS strategies and job permutation-based improvisations operators are respectively proposed for the no-wait flow shop scheduling problem with total flow time criterion and the single-machine earliness/tardiness problem.

Lately, few but important contributions have been reported concerning the application of HS algorithm to FJSSP. Indeed, when solved using metaheuristic approaches, a FJSSP solution is naturally coded using a two-vectors representation that express both affectation of machines to operations and the sequencing of operations on machines (operations permutation). Thus, FJSSP can be considered as a challenging applicative problem for the HS algorithm because it imposes the development of innovative domain-dependent HS operators that have to deal both with its discrete and permutation nature in one effective and efficient research process. Yuan et al. [47] tackled the problem using a HS-based hybrid optimization scheme that make use of the continuous domain HS algorithm augmented with a variable neighborhood local search approach exploiting the concept of critical path. Gao et al. [48] investigated a discreet HS approach for the multi-objective FJSSP (MOFJSSP) with weighted linear combination of two minimization criteria: Makespan and mean of earliness and tardiness. Mainly, the approach uses a GA crossover operator for the improvisation of the operations permutation part of the coding scheme. As well, Gao et al. [49] proposes a discrete HS approach for the FJSSP with fuzzy processing time that makes uses of an operations permutation-based improvisation operator that work only on two harmonies at a time. In [50], the authors presented a study on the application of different metaheuristic approaches for the resolution of the MOFJSSP investigating as well a discrete operation permutation-based HS (OP-DHS) proposal. Mainly, introduced improvisation operator deals clearly with the discrete and permutation nature of FJSSP and exploits a large amount of solutions in HM. In succinct experiments compared to a Genetic and TS algorithm, OP-DHS approach depicted a promising search behavior for the resolution of MOFJSSP with weighted linear combination of three minimization criteria, including Makespan.

From the above presented recent literature concerning the application of HS algorithm to the FJSSP, it is worth noting that conceptually, with the exception of OP-DHS algorithm, none of the other approaches proposes an adaptation of HS algorithm and operators to FJSSP, considering both its discrete and permutation-based nature. Mainly, the hybrid HS-based resolution process presented in [47] is formulated in the continuous domain. Although they use a discrete improvisation operator to deal with the affectation part of the solution vector, approaches presented in [48] and [49] did not propose a conceptually suitable permutation-based improvisation operator that, with respect to the canonical formulation of the HS algorithm, exploits a large number of the solutions stored in HM. It is thus noticeable that OP-DHS approach proposed in [50] constitutes a fundamental development regarding the application of HS algorithm to the FJSSP considering that it proposes a strong adaptation of the canonical continuous domain HS algorithm to the discrete and permutation-based nature of the problem. However, considering limitations of the reported experimental procedure and results, it is clear that further investigations and improvements of the basic OP-DHS approach are important to give a fair insight and a rigorous assessment of HS algorithm performances in its permutation-based form for the resolution of FJSSP using usual criterion and comparative procedures. This fact constitutes the main motivation of the present research work. Moreover, this investigation is also motivated both by the representativeness of the FJSSP in combinatorial optimization area and the recently reported "infancy" of research works concerning the application of HS algorithm to discrete and scheduling problems [35, 46]. Mainly, academic and practical implications of presently reported extended investigation of the HS algorithm in its combined discrete and permutation declination are expected to go beyond the context of FJSSP.

In accordance with the above presented research context and motivations, this paper presents as a main contribution an Effective Operations Permutation-based Discrete Harmony Search (EOP-DHS) approach for the resolution of FJSSP with Makespan criterion. The proposed EOP-DHS algorithm largely adopts the discrete and permutation-based research process of the basic OP-DHS [50] and improves its search ability exploiting a Modified declination of the Intelligent Mutation operator (MIM) previously introduced in [14]. Integrated to the basic OP-DHS scheme MIM is probabilistically applied on each newly improvised harmony to balance maximum machines workload (sum of processing times of the most loaded machine); this is done by re-affecting a flexible operation belonging to a machine with maximum workload to another allowable machine. Thus, MIM is mainly expected to have both explorative and exploitative role in the research process. Actually as a mutation operator, MIM conceptually plays an explorative role. By balancing machines maximum workload it also acts as an exploitative operator because of, as noted in [51], Makespan and maximum workload are often positively correlated. Mainly, MIM operator essentially allows maintaining and enhancing the reciprocal balance of explorative and exploitative power of the developed effective HS framework for the resolution of FJSSP.

In order to assess its effectiveness and efficiency, the proposed EOP-DHS algorithm is experimented using an extensive set of benchmarking problems and compared to a wild spectrum metaheuristic approaches recently deployed for the resolution of FJSSP including existing declinations of HS algorithm. Comparative algorithms have been chosen to have as fair as possible representation of the general domain of metaheuristic approach appliance to FJSSP with makespan, which is too large to be considered entirely. Thus, different classes of algorithms with different sophistication level but proved effectiveness for most of them have been selected for comparative purpose (GA, TS, Biogeography-Based Optimization (BBO), Mimetic Algorithm (MA), Artificial Bee Colony (ABC), Parallel Variable Neighborhood Search (PVNS), Artificial Immune Algorithm (AIA), Particle Swarm-Optimization, and Hybrid of Artificial Immune and simulated annealing (HAISA).

Furthermore, in order to rigorously assess the effectiveness of the contribution, the integrated MIM operator is investigated, and the EOP-DHS algorithm is experimentally validated in comparison with its mutation-free declination, which is precisely the mono-objective version of the original OP-DHS algorithm with Makespan criterion. Wilcoxon non-parametric statistical procedure is used to support the conclusions drawn.

The paper presents a complementary experimental contribution by investigating the effectiveness of the adopted permutation-based HS scheme for the resolution of classical JSSP. The basic OP-DHS algorithm (without the integration of operations affectation search mechanisms) is experimented for the resolution of different hard benchmarking instances of classical JSSP. A main motivation of such investigation is to enrich existing knowledge by providing a consistent evaluation of the adopted operation permutation-based HS scheme for the resolution of hard permutation-based scheduling problems. This is achieved by alleviating the possible experimental bias that can be introduced by simultaneously considering both of affectation and sequencing related HS components, as have been done for FJSSP. Considering the simplicity of the experimented HS framework, the experimental comparative investigation does not consider existing superior and generally sophisticated metaheuristic approach existing in the literature for the classical JSSP but just a restraint sample of algorithms representing different classes of sophistication and effectiveness.

The reminder of this paper is organized as follows: in the Section 2 we formulate FJSSP. The canonical HS algorithm is presented in Section 3. Section 4 details the proposed effective operations permutation-based discrete harmony search framework. Section 5 reports and discusses the results of the experimental studies. Finally, Section 6 concludes this paper with some perspectives for future works.

2 FJSS problem formulation

The FJSS problem is defined as follow: given a set of n independent jobs $\mathrm{J} = \{\mathrm{J}_1, \mathrm{J}_2,\ldots, \mathrm{J}_\mathrm{n}\}$to be processed on a set of m machines $M = \{M_1, M_2,\ldots, M_m\}$. Each job J_i consists of a sequence of n_i ordered operations $\{O_{i1}, O_{i2},\ldots, O_{in_i}\}$. Each operation O_{ij} $(i = 1,2,\ldots,n;\ j=1,2,\ldots,n_i)$ has to be processed, with no interruption, by one machine out of a set of allowable machines for the operation $M_{ij} \subseteq M$. The processing time of an operation O_{ij} on machine $M_k \in M_{ij}$ is p_{ijk}. Each machine can only handle at most one operation at a time.

A solution to the problem consists in assigning both a machine and a starting time t_{ij} to each operation to minimize the makespan $\mathrm{C}_{\max}$, which is the maximum of completion time of jobs.

$$C_{max} = \mathrm{Max}_{1 \leq i < n}\ C_i \tag{1}$$

Where, C_i is the completion time of the last operation of job J_i.

From the given problem definition, we note that the JSS Problem is a particular case of the FJSS Problem where the set of allowable machines for each operation contains exactly one machine (i.e. $|M_{ij}|=1\ \forall i=1,2,\ldots,n;\ \forall j=1,2,\ldots,n_i$).

3 The harmony search algorithm

The conceptual inception of the HS technic is the analogy between the intelligent exploration of harmonies space process depicted during jazz band musical improvisations, and intelligent explorative search in meta-heuristic optimization technic [33]. Thus, the HS algorithm is a population-based evolutionary algorithm that mimics the improvisation process of music players. Accordingly, when a group of musicians seek to find musically pleasing melodies, they iteratively explores the harmonies search space by a creative process that is made up of three different stages: (*i*) playing any pitch from memory, (*ii*) modifying a pitch which exists in the memory, or (*iii*) improvising any pitch from the possible pitch range. The New Harmony is then integrated or not in musician's memories according to its audio-aesthetic significance.

According to the optimization point of view, a decision solution vector is represented by a harmony, and a population of solution vectors is represented by a harmonies memory. At each algorithm iteration a new harmony is generated (improvised) with the value of each decision variable composing the solution vector obtained using one of the following three options: (*i*) assigning a value from the memory, (*ii*) modifying an existing value in the memory, or (*iii*) assigning a possible random value. The generated harmony is then assessed for insertion in the harmonies memory using a predefined replacement policy based on its objective value.

This analogy give rise to a simple iterative optimization approach that consists of five steps:

Step 1: *Initializing the optimization problem and algorithm parameters*
Step 2: *Initialize the Harmonies Memory (HM)*
Step 3: *Improvising a new harmony from the HM*
Step 4: *Updating the HM*
Step 5: *Repeat Step 3 and 4 until the termination criterion is met.*

The continuous and basic form of the algorithm is succinctly presented in the following subsections.

3.1 Initializing the optimization problem and algorithm parameters

The continuous optimization problem is defined as follows:

$$minimize F(x), \tag{2}$$

Subject to $x_i \in \mathrm{X_i} = \{x_i(1), \ldots, x_i(k), \ldots, x_i(k_i)\}$

Where $F(x)$ is the objective function, and $x = \{x_1, \ldots, x_N\}$ is the optimal solution that has to be found by the harmony search algorithm. The set of candidate values for the variables are $x_i \in X_i = \{x_i(1), \ldots, x_i(k), \ldots, x_i(k_i)\}$.

The algorithm initialization phase considers the following parameters:

- Harmony memory size (HMS), (i.e. number of solution vectors in the harmonies memory)
- Harmony memory consideration rate (HMCR), where HMCR $\in$ [0, 1]
- Pitch adjusting rate (PAR), where PAR$\in$ [0, 1]
- Number of improvisations (NI) that is the stopping criterion.

3.2 Harmony memory representation and initialization

A matrix of solution vectors, extended with a column of calculated objective value for each corresponding harmony represents the Harmony Memory (HM). During initialization, the matrix is filled with randomly generated solution vectors. Accordingly, each solution value is randomly selected within the range of its set of allowable values. The solutions are evaluated and based on their objective function values, they are sorted with $\mathrm{a^{worst}}$ and $\mathrm{a^{Best}}$ the worst and best harmony vector respectively.

$$\mathrm{HM} = \left[\begin{array}{cccc|c} x_1^1 & x_2^1 & \cdots & x_N^1 & f(x^1) \\ x_1^2 & x_2^2 & \cdots & x_N^2 & f(x^2) \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ x_1^{HMS} & x_2^{HMS} & \cdots & x_N^{HMS} & f(x^{HMS}) \end{array}\right] \tag{3}$$

3.3 Improvising a new harmony from the HM

Harmony improvisation process embeds recombination operators of the approach. It principally associates at each iteration different intensification and explorative components for the generation of a new harmony. It is mainly expected to exploit a wide region of the search space during recombination, which is a noticeable particularity compared to other population-based evolutionary algorithms. Improvisation makes use of three recombination operators for the generation of a new harmony $\mathrm{x} = \{x'_1, x'_2, x'_3, \ldots, x'_N\}$: memory consideration, pitch adjustment, and randomization.

Applied with a certain probability determined by HMCR$\in$ [0, 1], harmony consideration forces the new values x'_i to be randomly inherited from corresponding values stored in HM:$\{x_i^1, x_i^2, \ldots\ldots, x'_{HMS}\}$. Otherwise, using a randomization component with an implicit probability of (1 – HMCR), value of x'_i is chosen according to its admissible range $\mathrm{X_i}$. The following equation summarizes memory consideration and random consideration steps:

$$x_i' \leftarrow \begin{cases} x_i' \in \{x_i{}^1, x_i{}^2, \ldots\ldots, x_{HMS}'\} & \text{w.p.HMCR} \\ x_i' \in X_i & \text{w.p. } (1 - \text{HMCR}) \end{cases} \tag{4}$$

Using a probability determined by the PAR parameter, pitch adjustment component examines every value chosen by harmony consideration to determine whether it should be adjusted or not. Adjusting decision for x'_i is given by (5), where *bw* is an arbitrary distance bandwidth that determines the nature and amount of change that can occur in x'_i It depends mainly on the considered optimization problem. The function rand() generates a random number $\in [0, 1]$.

$$x_i' \leftarrow \begin{cases} x_i' = x_i' \pm \text{rand}().\text{bw} & \text{w.p.PAR} \\ x_i' = x_i' & \text{w.p.} \quad (1 - \text{PAR}) \end{cases} \tag{5}$$

3.4 Updating the HM

Harmonies Memory updating mechanism is based on a replacement policy that can take different forms. The replacement policy mainly characterize the admissibility or not of the new improvised solution (New Harmony) in the Harmonies Memory and, if considered, which solution from the HM will be replaced. According to its common and basic form, the HM is updated if the new generated harmony vector is better than the worst harmony in the HM. In this case a^{new} will replace a^{worst} and become a new member of the HM. If a^{new} is worst than a^{worst} it is not considered. The following equation summarizes HM updating step:

$$a^{worst} \leftarrow \begin{cases} a^{new} & \text{if}: \; f(a^{new}) < f(a^{worst}) \\ a^{worst} & \text{if}: \; f(a^{worst}) \leq f(a^{new}) \end{cases} \tag{6}$$

3.5 Stopping criterion

If the predefined termination criterion is met - for example the number of improvisations -, return the best harmony vector a^{best}. Otherwise, go to step (3).

4 An effective operations permutation-based discrete HS algorithm for the FJSSP

In this section, the improved discrete permutation-based form of the canonical HS algorithm introduced in Section 3 is presented for the resolution of FJSSP with Makespan minimization objective. Mainly, for completeness this section highlights important conceptual and algorithmic features of the basic OP-DHS approach adopted in this work. Furthermore, this section details the integrated MIM operator. At the end of this section, the overall algorithm flow of the proposed EOP-DHS approach is presented.

4.1 Harmony coding and decoding

As reported by Wu and Wu [16] and Zhang et al. [15] different coding approaches have been used in FJSSP literature. In this work, the adopted representation from [15] codes the FJSSP solution vector by means of two parts that express independent assignment of machines to operations and the processing sequence of operations on machines. Accordingly, a Harmony representation for FJSSP is composed of two vectors: a Machine Selection Vector (MSV) and an Operation Sequence Vector (OSV). Table 1 depicts

Table 1 Example of a FJSSP instance

Jobs	Ops	Machines		
		M1	M2	M3
	O11	N/A	1	2
J1	*O12*	1	5	3
	O13	4	1	N/A
	O21	3	5	6
J2	*O22*	N/A	2	1
	O23	1	3	N/A
	O31	N/A	6	2
J3	*O32*	5	1	4
	O33	2	1	2
	O34	1	N/A	3

an example of a FJSSP instance. Numerical values indicate the execution time of each operation on the corresponding allowable machines. N/A means that the machine is Not Allowable for the corresponding operation. Figure 1 illustrates a harmony MS/OS vectors representation of a feasible solution for FJSSP of Table 1.

Integer-valued MSV indexed by p, expresses the assignment of each operation to a machine selected from the set of allowable machines for the corresponding operation. Its number of elements is equal to the total number of operations of all jobs (T_{op}) and each element MSV_p indicates a column index (a machine choice) in the matrix of alternative allowable machines/processing-time for the operation O_{ij} corresponding to P. As an example and as depicted in Fig. 1, the operation $O_{2,2}$ is assigned to machine 2 which correspond to the column index 1 in the matrix of alternative allowable machines for that operation. The second row in the matrix corresponds to the execution times of the operation $O_{2,2}$ for each of the possible machines.

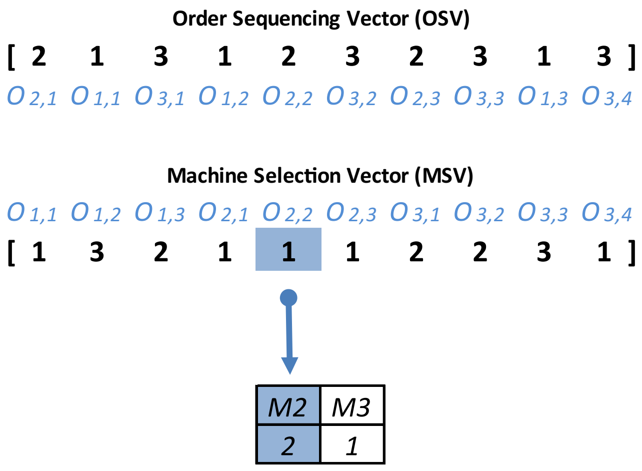

Order Sequencing Vector (OSV)
[2 1 3 1 2 3 2 3 1 3]
O2,1 O1,1 O3,1 O1,2 O2,2 O3,2 O2,3 O3,3 O1,3 O3,4
Machine Selection Vector (MSV)
O1,1 O1,2 O1,3 O2,1 O2,2 O2,3 O3,1 O3,2 O3,3 O3,4
[1 3 2 1 1 1 2 2 3 1]
M2 M3
2 1

Fig. 1 MS/OS coding of a Harmony

The Operations Sequence Vector (OSV) indexed by (s) expresses the processing sequence of all the operations in the system. With a length also equal to the total number of operations (T_{op}), it uses an "Operations Permutation-based representation" that defines all operations of a job with the same integer value and then interprets them according to the sequence of their appearance. Therefore, the index i of the job J_i appears n_i times in the vector, with n_i the number of operations of the job J_i. As an example, the OS part of the solution vector illustrated in Fig. 1 expresses the following sequence of operations execution: $(O_{2,1} \rightarrow O_{1,1} \rightarrow O_{3,1} \rightarrow O_{1,2} \rightarrow O_{2,2} \rightarrow O_{3,2} \rightarrow O_{2,3} \rightarrow O_{3,3} \rightarrow O_{1,3} \rightarrow O_{3,4})$.

The evaluation procedure is based on a decoding process that transforms Harmony solution vectors to a feasible schedule. In this study, a priority-based and machine's idle-time occupation decoding scheme such as described in [12, 13, 15] is used. It assists in a construction process that iteratively inserts each operation of OSV in its corresponding selected machine according to its priority, which is determined by its order of appearance. The corresponding machine and processing time are identified using the MSV, and the operation is inserted at the first allowable idle-time interval respecting precedence constraints, or at the end of the machine. The algorithm presented in Fig. 2 depicts the adopted decoding procedure and an example of a construction of a schedule structure (represented by a Gantt diagram) integrating machines idle-time occupation process is depicted in Fig. 3b. The considered Harmony is: [2 1 3 1 2 3 2 3 1 3]. During the construction process the operation $O_{3,2}$ is inserted at the allowable idle time interval in machine 2 before the operation $O_{2,2}$, generating a final Makespan of eight time unit. The utilization of a decoding process that does not integrate machines idle-time occupation strategy place the operation $O_{3,2}$ after the operation $O_{2,2}$ and generates a Makespan of 9 time unit (Fig. 3a).

4.2 Harmonies memory initialization

Expected to enhance evolutionary algorithms resolution process efficiency, different population initialization strategies have been adopted in the FJSSP literature. Such as the strategies presented in [14, 15]. These approaches generally consider different local and global search heuristic mechanisms for minimizing or balancing machine's workload during the generation of a new solution, introducing a number a supplementary algorithm parameters to be tuned. In this work, a simple random harmonies memory initialization approach is adopted such as to maintain the number of algorithm parameters as small as possible. Furthermore, this strategy allows the generation of an initial HM with high solutions diversity.

4.3 Harmony improvisation operator

According to HS optimization process, at each algorithm step a new harmony is improvised from HM using three recombination components: memory consideration, pitch adjustment, and random consideration. In this work, two dedicated discrete improvisation schemes fitting the two-part adopted FJSSP representation are adopted: Machine Selection and Operations Sequence vector Improvisation.

4.3.1 Machine selection vector improvisation

The integer coded machines selection vector presents no particular structural relationships between its components. Accordingly, the MSV improvisation scheme uses with slight modifications, related to the integer valued nature of the MSV, the basic operators presented in Section 3.3. Hence, applied with a certain probability determined by HMCR $\in [0,\ 1]$, the harmony consideration operator force the new values of machine selection index for the operation $O_{i,j}$ at the position p (MSV_p^{New}) to be inherited from a randomly selected historical values stored in the HM. Otherwise and using a random consideration with an implicit probability of $(1 - \mathrm{HMCR})$, the values of MSV_p^{New} is randomly chosen according to its possible range of values. Every component chosen by harmony consideration is examined for adjustment in its possible range of values, which is the number of allocation possibilities for that operation.

Figure 4 depicts a succinct example of an MSV improvisation process for the problem described in Table 1. As it is shown, the new final MSV part of the harmony is iteratively generated from the Machine selection part of the HM. Indeed, each element is generated using one of the possible rules with a certain probability: Consideration, Consideration + adjustment, or Randomization. Value adjustment and randomization are expected in the range of the possible allocations for each operation that is also depicted in Fig. 4.

4.3.2 Order sequence vector improvisation

As formerly presented in Section 4.1, the OSV uses an operations Permutation-based representation commonly adopted for metaheuristic approach to FJSSP. Therefore, ordinary HS improvisation operators are not suitable to this structurally constrained representation because of the possibility to generate unfeasible solutions. An operations permutation-based HS improvisation operator is adopted in this work in order to tackle this issue. This operator depicts two main conceptual features: It approaches the explorative behavior of the HS improvisation operator in its native formulation and avoids from the generation of an unfeasible OSV during recombination. Particularly, as for the basic

```
Procedure: Harmony Decoding
Input: Harmony MS/OS Vectors, Problem Data
Output: A schedule Structure

1- Initialize the Schedule Structure
2- For Each element $q$ of the $OSV$, with $q = [1, \dots, T_{op}]$ do
3-     Decode each integer $OSV_q$ to the corresponding Operation $O_{i,j}$
4-     Refer to $MSV$ to determine the integer element $MSV_p$ corresponding to the index of
       the array of alternative machines for the corresponding operation $O_{i,j}$ .
5-     Identify from the dataset the corresponding selected machine $M_k$ and processing
       time $d_{i,j,k}$
6-     If $O_{i,j}$ is the first operation of the Job $J_i$ Then
           Set $t_{i,j}$ the starting time of $O_{i,j}$ to 0
7-     Else
           Set $t_{i,j}$ to $tf_{i,j-1}$ the stop time of the predecessor operation $O_{i,j-1}$
8-     EndIf
9-     If $M_k$ did not process any operation Then
           Set $ts_k$ the stop time of the machine $M_k$ to 0
10-    Else
           Set $ts_k$ the stop time of the last operation on $M_k$
11-    EndIf
12-    If $ts_k \leq t_{i,j}$ Then
           Add $O_{i,j}$ to $M_k$ starting at $t_{i,j}$
13-    Ese If it exist (an idle-time interval between $t_{i,j}$ and $ts_k$) $\geq d_{i,j,k}$ Then
           Insert $O_{i,j}$ in $M_k$ starting at $t_{i,j} = Max(t_{i,j}, t_{idle})$ , where $t_{idle}$ is the starting
           time of the first allowable machine idle-time interval
14-    Else
           Add $O_{i,j}$ to $M_k$ starting at $ts_k$
15-    EndIf
16- EndFor
```

Fig. 2 Harmony decoding procedure

HS algorithm, for the generation of one OSV, this operator exploits with some probability a large amount of the information contained within the HM and is expected to increase its exploitation ability with the increase of the number of jobs. The operator processes according to two consecutive conceptual stages (Procedure in Fig. 5):

The first Stage concerns the generation of a matrix of N intermediates OSVs, each one corresponding to a specific Job. Accordingly, for each job from the vector of randomly sorted set of all jobs, the job-based consideration scheme, if used, copies all the operations of the considered job from a randomly selected OSV from the HM in their corresponding positions in the empty intermediate OSV. Next, if applied to the intermediate OSV the Operation-Based adjustment scheme adjusts randomly the position of a randomly selected operation in the vector. If consideration and

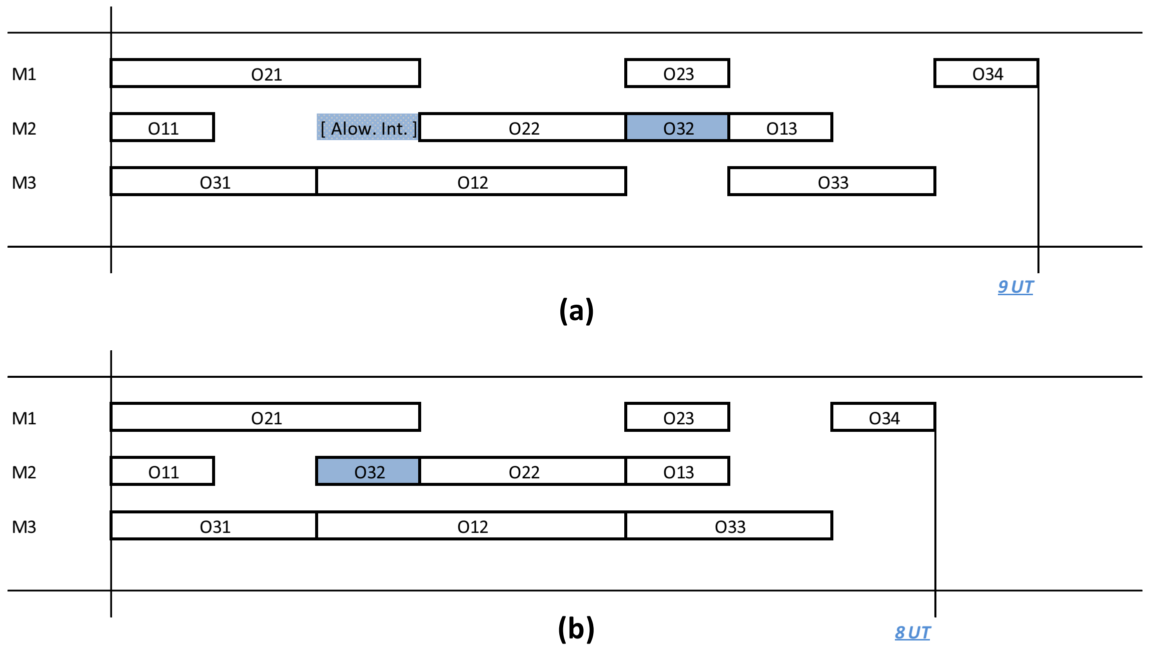
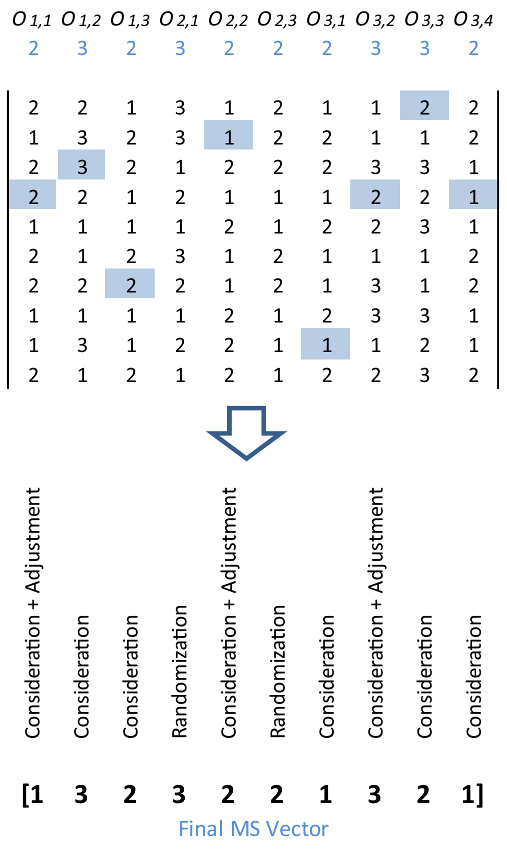


Fig. 3 Example of a decoding procedure without (**a**) and with allowable machine idle time interval occupation strategy (**b**)

adjustment are not of concerns, a random intermediate OSV is obtained by applying the Job-Based consideration scheme to a newly generated OSV. In the second stage, the final OSV is generated from the matrix of intermediates OSVs. Essentially, the components of the final OSV are filled considering the values of non-empties components of each column of the matrix of intermediates OSVs sequentially. The values of non-empties components of each column are considered according to their order of apparition from the first to the last line of the matrix.



$O_{1,1}$	$O_{1,2}$	$O_{1,3}$	$O_{2,1}$	$O_{2,2}$	$O_{2,3}$	$O_{3,1}$	$O_{3,2}$	$O_{3,3}$	$O_{3,4}$
2	3	2	3	2	2	2	3	3	2
2	2	1	3	1	2	1	1	2	2
1	3	2	3	1	2	2	1	1	2
2	3	2	1	2	2	2	3	3	1
2	2	1	2	1	1	1	2	2	1
1	1	1	1	2	1	2	2	3	1
2	1	2	3	1	2	1	1	1	2
2	2	2	2	1	2	1	3	1	2
1	1	1	1	2	1	2	3	3	1
1	3	1	2	2	1	1	1	2	1
2	1	2	1	2	1	2	2	3	2
Consideration + Adjustment	Consideration	Consideration	Randomization	Consideration + Adjustment	Randomization	Consideration	Consideration + Adjustment	Consideration	Consideration

[1 3 2 3 2 2 1 3 2 1]

Final MS Vector

Fig. 4 Example of MSV Improvisation Process

In Fig. 6 the OSV improvisation process for the problem described in Table 1 is illustrated by an applicative example. Actually, the randomly generated Job^{Rand} vector in this example is [2, 1, 3].

4.4 Modified intelligent mutation operator

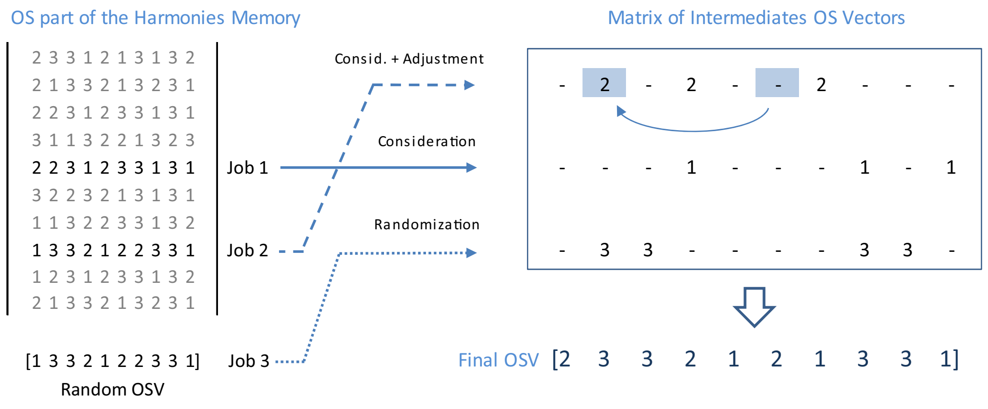
Inspired by the Intelligent Mutation operator proposed in [14] and also used in [19], the Modified Intelligent Mutation (MIM) operator is applied with a certain rate to the newly improvised harmony. The operator mainly balances maximum machines workload by reassigning a random flexible operation from the machine with the maximum workload to another randomly chosen allowable machine. Main difference between the MIM and the original version of the operator is that it considers systematically the obtained solution as a current one. Actually, the original operator used in [14] evaluates the fitness of the generated solution and consider it only if it minimize Makespan comparatively to the altered individual.

MIM operator processes according to three steps summarized in Fig. 7. Its utilization implies the introduction of a new algorithm parameter that is the Modified Intelligent Mutation Rate (MIMR).

```
Procedure: Operations Permutation-based OSV Improvisation
Input: Harmony Memory, Consideration rate, Adjustment rate
Output: New OSV
Stage 1:
1- Initialize $\mathrm{OSV}^{\mathrm{New}}$ the New OS Vector
2- Initialize $Job^{rand}$: the vector of randomly sorted set of jobs
3- Initialize $OSV^{Inter}$: the Matrix of Intermediates OS Vectors
4- For Each element $r$ of $Job^{Rand}$, with $r = [1, \ldots, N]$ do
5-     Identify the current Job: $Job_r^{Rand}$
6-     Initialize $OSV_r^{Inter}$ : the $r^{th}$ intermediate OS vector with empty positions
7-       If Consideration Then
8-           Select $OSV^{Mem}$ : the OS part of a random Harmony from the Memory
9-           Copy from $OSV^{Mem}$ the values equal to $Job_r^{Rand}$ in their respective positions in
             $OSV_r^{Inter}$
10-          If Adjustement Then
11-                  Select a random non-empty position in $OSV_r^{Inter}$
12-                  Insert the corresponding value to another randomly selected empty position
13-          End If
14-      Else
15-          Generates $OSV^{\mathrm{rand}}$: a randomly generated OS vector
16-          Copy from $OSV^{\mathrm{rand}}$ the values equal to $Job_r^{Rand}$ in their respective positions in
             $OSV_r^{Inter}$
17-      End If
18-   Add $OSV_r^{Inter}$ to $OSV^{Inter}$
19- End For
Stage 2:
20- For Each column $c$ of $\mathrm{OSV}^{\mathrm{New}}$, with $c = [1, \ldots, T_{op}]$ do
21-     For Each line $l$ of $OSV^{Inter}$, with $l = [1, \ldots, N]$ do
22-         If $OSV_{(c,l)}^{Inter}$ is non empty Then
                Add element at $OSV_{(c,l)}^{Inter}$ to $\mathrm{OSV}^{\mathrm{New}}$
23-         End If
24-     End For
25- End For
```

Fig. 5 OSV Improvisation procedure

Fig. 6 Example of OSV Improvisation procedure



Step 1: Identify the Machine M^{max} with the maximum workload

Step 2: Select, if it exist, OP^{max} a random operation from M^{max} with a flexibility > 1

Step 3: Reassigns in MSV randomly OP^{max} to a different allowable machine.

Fig. 7 Conceptual steps of the modified intelligent mutation operator

4.5 Framework of the proposed EOP-DHS algorithm

As described in Fig. 8, the overall framework of the proposed EOP-DHS algorithm to FJSSP consists in six steps. Actually, the five steps canonical HS framework is extended with the application of MIM operator and modified according to the presented discrete and permutation form of the adopted operators. Besides, we note that despite the introduction of a new algorithm parameter that is MIMR, the overall number of HS algorithm parameters is maintained to five. This is because the presented approach does not use an adjustment bandwidth as for the canonical HS algorithm.

5 Experimental study

In order to assess the effectiveness of the proposed EOP-DHS algorithm this section provides an experimental evaluation of its performances and a comparative study with different stat of the art metaheuristic approaches to FJSSP, considering significant benchmarking instances of different dimensions. The comparative algorithms have been mainly chosen such as to give a wide exemplification of the spectrum of approaches previously investigated for the resolution of FJSSP, and they represent different levels of algorithmic complexity and effectiveness. Additionally, the proposed enhanced EOP-DHS approach integrating the MIM operator is experimentally compared to the basic OP-DHS algorithm (the mutation-free EOP-DHS) and the Wilcoxon non-parametric statistical test is used in order to characterize achieved improvement. Finally, the experimental examination is extended with supplementary results related to the performances of the adopted operations-permutation based HS framework for the resolution of the classical JSSP comparatively to a sample of metaheuristic algorithms previously deployed for the resolution of the classical JSSP.

Step 1: Initialize the problem and algorithm parameters
Initializes Scheduling Instances, Harmonies Memory Size (HMS), Memory Consideration Rate (MCR), Pitch Adjustment rate (PAR), Modified Intelligent Mutation Rate (MIMR), Number of Improvisations (NI).

Step 2: Initialize Harmonies Memory (HM)
Initialize the HM with a randomly generated harmonies, Evaluates each harmony and determines the Best and Worst Harmonies

Step 3: Improvise a New Harmony
Improvise a New MSV using the Integer value-based Improvisation scheme, Improvise a New OSV using the Operations Permutation-Based Improvisation scheme

Step 4: Apply MIM Operator
Apply MIM Operator to the newly generated Harmony

Step 5: Update HM
Replace if better the Worst Harmony in the HM with the newly improvised one, Update Best and Worst Harmonies

Step 6: Repeat Step 3, 4 and 5 until the termination criterion is met.
Repeat Improvisation, *Intelligent Global Adjustment and HM updating for the predefined number of Improvisation.*

Fig. 8 Overall framework of the proposed EOP-DHS algorithm for the FJSSP

5.1 EOP-DHS approach performances and comparative results

The first experimentation round in this subsection concerns the assessment of the performances of the proposed EOP-DHS approach comparatively to several previously published state of the art metaheuristic proposals to FJSSP. Thus, the ten BRdata test instances of Brandimarte [5] are first considered and in order to enhance approach efficiency, algorithm parameters are set empirically as follow: MCR = 0.97, PAR = 0.01, MIMR = 0.5 and NI = 500 000. HMS parameter is adopted as a free dimensioning parameter and is determined empirically for each instance of the problem. Besides, the algorithm is coded in Java language and run on an I7 PC with 3.30 GHz and 8 GB of RAM memory. Regarding the stochastic nature of the proposed metaheuristic approach, 30 runs are realized for the resolution of each problem instance.

Table 2 compares the EOP-DHS algorithm to six algorithms previously investigated on BRdata; ABC algorithm of Wang et al. [21], PVNS algorithm (PVNS) of Yazdani et al. [18], MA of Kobti [26], GA of Pezzella et al. [14], the hybrid Tabu Search with Public Critical Block-based neighborhood structure (TSPCB) of Li et al. [9], and BBO of Rahmati et al. [17]. The first three columns report respectively the name of the instance, the corresponding number of jobs and machines and the best-known solution obtained so far for that instance (*BKS*). The following columns report best-reported solutions sets for the different algorithms and both the reported computational results of the ABC algorithm and the obtained computational results for the proposed EOP-DHS. For the ABC and the EOP-DHS algorithms, the results involve four metrics for each problem: the best-obtained Makespan (*BCm*), the average Makespan (A*vCm*), the standard deviation of the Makespan (*SDV*), and the average computational time in seconds (*AvT(s)*). Besides, to characterize their global relative performances, the number of best solution obtained (*#BCm*) and Mean

Table 2 Results of the proposed EOP-DHS and comparison with different metaheuristic approaches on BRdata

Inst.	$N \times M$	BKS	BBO	TSPCB	GA	MA	PVNS	ABC				EOP-DHS				
								BCm	*AvCm*	*SDV*	*AvT(s)*	*BCm*	*AvCm*	*SDV*	*AvT(s)*	*HMS*
Mk01	10×6	(40)	**40**	**40**	**40**	**40**	**40**	**40**	40,0	0,00	3,2	**40**	40,0	0,00	0,1	100
MK02	10×6	(26)	28	**26**	**26**	**26**	**26**	**26**	26,5	0,50	35,6	**26**	26,7	0,34	2,7	100
MK03	15×8	(204)	**204**	**204**	**204**	**204**	**204**	**204**	204,0	0,00	1,2	**204**	204,0	0,00	0,0	100
MK04	15×8	(60)	66	62	**60**	**60**	**60**	**60**	61,2	1,36	38,9	**60**	60,5	0,84	5,5	200
MK05	15×4	(172)	173	**172**	173	**172**	173	**172**	173,0	0,14	19,3	**172**	172,7	0,45	13,7	200
MK06	10×15	(57)	64	65	63	59	60	60	64,5	1,75	66,6	60	61,1	0,87	23,3	100
MK07	20×5	(139)	144	140	**139**	**139**	141	**139**	141,4	1,20	131,8	**139**	140,7	1,10	41,6	300
MK08	20×10	(523)	**523**	**523**	**523**	**523**	**523**	**523**	523,0	0,00	2,3	**523**	523,0	0,00	0,8	100
MK09	20×10	(307)	310	310	311	**307**	**307**	**307**	308,8	1,63	91,2	**307**	308,1	1,66	36,3	200
MK10	20×15	(196)	230	214	212	216	208	208	212,8	2,43	237,1	207	211,5	2,59	227,5	200
#BCm			3	5	6	7	6	8				8				
MRD(%)			5,3	2,82	2,06	1,37	1,34	1,14				1,09				

Makespan values in bold indicate that this is the best known solution obtained

Relative Deviation metric (*MRD*) are reported for each algorithm. An algorithm *MRD* from the *BKS* set is defined as follows:

$$MRD\ (\%) = \frac{1}{p}\sum_{i=1}^{p} Rdv_i$$

With $Rdv_i\ (\%) = \frac{1}{C_i}\left(C_i - C_i^{bks}\right) \times 100$

Where Rdv_i, represents the relative deviation for the considered instance i, p denotes the number of tested instances, C_i denotes the Makespan of the considered instance, and C_i^{bks} denotes the best-known Makespan so far for the instance i.

Table 2 results indicate that the proposed EOP-DHS algorithm to FJSSP depicts a superior behavior comparatively to a wild spectrum of metaheuristic approaches previously investigated for the FJSSP. Actually, the algorithm dominates the six algorithms in term of MRD from the best-known solution set, where it achieves a minimum MRD of 1.09%. Concerning the *#BCm* indicator, both the proposed EOP-DHS and the ABC approaches achieved the best result among the seven approaches with eight best solutions obtained out of the 10 test problems. It is noticeable that ABC algorithm depicts an interesting competitive behavior compared to the proposed EOP-DHS both in term of solution quality and robustness to initial conditions (appreciated using *AvCm* and *SDV* metrics). Because of the difference in computing environments, it can be simply appreciated that both two approaches depict a comparable and acceptable computational efficiency. Hence, considering the nature of the two approaches and the fact that EOP-DHS algorithm is not equipped with any sophisticated initialization scheme or critical path-based local search technique as with ABC approach, one can easily support the contextual dominance of EOP-DHS algorithm to ABC. In the other side, the appreciated superiority of the proposed EOP-DHS approach to the overall set of considered algorithms, amongst them different highly competitive algorithms such as PVNS, MA, and GA, confirm that the EOP-DHS algorithm is effective and robust for the resolution of the FJSSP on the well-known BRdata set.

The second experimentation round concerns a comparative investigation of performances of the proposed EOP-DHS to FJSSP on Fdata benchmarking set [4]. Thus, EOP-DHS results concerning the ten medium size Fdata problems are considered comparatively to the results obtained by three effective and competitive metaheuristic proposals from the literature: Artificial Immune Algorithm (AIA) of Bagheri et al. [19], Hybrid of Artificial Immune and Simulated Annealing approach (AISA) of Roshanaei et al. [24], and Particle Swarm-based approach of Teekeng et al. [23] (EPSO). Table 3 reports for each algorithm and problem instance the best-found solution and relative deviation from the BKS. Besides, the table reports previously introduced performance metrics for each algorithm (MRD, #BCm). Algorithm parameters are set empirically as follow: MCR = 0.97, PAR = 0.01, MIMR = 0.5, NI = 500 000 and HMS = 100.

It can be noticed in Table 3 that the search ability of the proposed EOP-DHS approach is also confirmed using a different dataset and comparatively to different effective and sophisticated metaheuristic approaches to the FJSSP. Indeed, the EOP-DHS algorithm obtained the best performances among the four algorithms with nine BKSs out of the ten considered problems and a superior MRD of 0.025%.

The third and final experimentation round of this subsection concerns the comparison of the proposed EOP-DHS approach to existing declinations of HS algorithm to the

Table 3 Results of EOP-DHS and comparison with existing FJSSP metaheuristic approaches on Fdata

Inst.		n × m	BKS	AIA		AISA		EPSO		EOP-DHS	
				BCm	*Rdv (%)*	*BCm*	*dv (%)*	*BCm*	*Rdv (%)*	*BCm*	*Rdv (%)*
MFJS	1	5 × 6	468	**468**	0,00	**468**	0,00	**468**	0,00	**468**	0,00
MFJS	2	5 × 7	446	448	0,45	**446**	0,00	**446**	0,00	**446**	0,00
MFJS	3	6 × 7	466	468	0,43	**466**	0,00	**466**	0,00	**466**	0,00
MFJS	4	7 × 7	554	**554**	0,00	**554**	0,00	**554**	0,00	**554**	0,00
MFJS	5	7 × 7	514	527	2,53	**514**	0,00	**514**	0,00	**514**	0,00
MFJS	6	8 × 7	634	635	0,16	**634**	0,00	**634**	0,00	**634**	0,00
MFJS	7	8 × 7	879	**879**	0,00	**879**	0,00	**879**	0,00	**879**	0,00
MFJS	8	9 × 8	884	**884**	0,00	894	1,13	**884**	0,00	**884**	0,00
MFJS	9	11 × 8	1055	1088	3,13	1088	3,13	1059	0,38	**1055**	0,00
MFJS	10	12 × 8	1196	1267	5,94	**1196**	0,00	1205	0,75	1199	0,25
#BCm				4		8		8		9	
MRD(%)					1,26		0,43		0,11		0,025

Makespan values in bold indicate that this is the best known solution obtained

FJSSP on BRdata: the basic Operations permutation-based discrete HS (OP-DHS) approach of Gaham et al. [50], the Discrete HS (DHS) algorithm of Gao et al. [48], and the Hybrid HS (HHS) algorithm of Yuan et al. [47]. Indeed, in Table 4 the first and second columns indicate the name for each test instance and its corresponding BKS. The following columns report best solutions obtained by the three algorithms and the corresponding relative deviation from the BKS for each problem. MRD and *#BCm* metrics are also included for each algorithm.

Reported results show that the proposed EOP-DHS algorithm clearly outperforms existing discrete adaptations of the HS algorithm to the FJSSP both in term of MRD and *#BCm*. The EOP-DHS algorithm achieve also competitive performances compared to the hybrid critical path-based local search continuous domain HHS approach with a same *#BCm* and a difference in MRD that is inferior to 0.5%, what is in our sense an expected result considering the nature of the two algorithms. These succinctly presented results confirm that the EOP-DHS is of an outperforming searching ability comparing to stat of the art discrete harmony search approach to the FJSSP.

5.2 Performance analysis of the modified intelligent mutation operator

In order to assess the effectiveness of the contribution, particularly the effect of the integration of MIM operator into the research process, the EOP-DHS algorithm is experimentally validated in this subsection in comparison with its Mutation-Free declination (EOP-DHS$_{MF}$) (that is essentially the basic OP-DHS algorithm to the mono-objective FJSSP with Makespan). Taking into account the motivation of the following experimentation that is self-contained

Table 4 Results of EOP-DHS and comparison with existing HS-based FJSSP Metaheuristics proposals

Inst.	BKS	OP-DHS		DHS		HHS		EOP-DHS	
		BCm	*Rdv (%)*	*BCm*	*Rdv (%)*	*BCm*	*Rdv (%)*	*BCm*	*Rdv (%)*
Mk01	40	**40**	0,00	**40**	0,00	**40**	0,00	**40**	0,00
MK02	26	**26**	0,00	28	7,69	**26**	0,00	**26**	0,00
MK03	204	**204**	0,00	**204**	0,00	**204**	0,00	**204**	0,00
MK04	60	61	1,67	**60**	0,00	**60**	0,00	**60**	0,00
MK05	172	173	0,58	**172**	0,00	**172**	0,00	**172**	0,00
MK06	57	61	7,02	67	17,54	58	1,75	60	5,26
MK07	139	140	0,72	143	2,88	**139**	0,00	**139**	0,00
MK08	523	**523**	0,00	**523**	0,00	**523**	0,00	**523**	0,00
MK09	307	**307**	0,00	309	0,65	**307**	0,00	**307**	0,00
MK10	196	214	9,18	212	8,16	205	4,59	207	5,61
#BCm		5		5		8		8	
MRD (%)			1,92		3,69		0,63		1,09

Makespan values in bold indicate that this is the best known solution obtained and the fact that previous presented experimentations of the EOP-DHS algorithm have been carried out using the Harmony size as a free dimensioning parameter, an alternative and uniform experimental design is adopted in this part. Thus, the EOP-DHS$_{MF}$ and the EOP-DHS algorithms are experimented using both BRdata and Fdata benchmarking sets and with the following parameters: HMS = 200, MCR = 0.97, PAR = 0.01, and NI = 300 000. The MIMR is naturally set to zero for the EOP-DHS$_{MF}$ and to 0.5 for the EOP-DHS. Besides, 50 runs are realized by each algorithm for the resolution of each problem instance and Wilcoxon signed rank test is adopted to measure the statistical relevance of the results. The confidence level for all tests is set to 95% (corresponding to $\alpha = 0.05$). The Wilcoxon signed rank test [52] is a non-parametrical statistical procedure generally used to perform pairwise performances comparison of evolutionary algorithms [53]. Particularly, the test aims at the detection of significant differences between two sample means, that is, the behavior of two algorithms over a certain number of runs. The Wilcoxon signed ranks test have been for example used in several studies concerning the application of metaheuristic approach for the resolution of manufacturing scheduling problems [54, 55].

Table 5, presents obtained results. The first column reports the name of each instance. Its corresponding Best and average Makespan value obtained over 50 runs for both the EOP-DHS$_{MF}$ and the EOP-DHS algorithms are reported in the following four columns. Sixth column reports the sum of ranks for the problems in which the EOP-DHS algorithm outperformed the EOP-DHS$_{MF}$ (R+). The sum of ranks for the opposite (R -) and the numbers of ties cases (Ties) among the 50 runs are respectively reported in the seventh and the eighth columns. The last column reports the corresponding *p-value* that indicates the statistical relevance of obtained results. The attached signs "+" or "−" indicate that the proposed EOP-DHS algorithm performs significantly better or worse than EOP-DHS$_{MF}$ algorithm. The sign "=" denotes that there is no significant difference between the two declinations. Additional indicators; numbers of "=", "−" and "+" cases over the twenty benchmarking instances are also reported in the table.

It can be observed from corresponding columns in Table 5 that EOP-DHS algorithm shows better performances compared to its mutation-free version considering average Makespan value among the 50 runs. This clearly indicates that by acting on maximum machines workload the integration of MIM operator improves the search ability of the adopted HS framework for the resolution of FJSSP. This conclusion is confirmed by the conducted statistical test. Actually, statistical results indicate that EOP-DHS algorithm performs significantly better than the EOP-DHS$_{MF}$ algorithm on 13 out the 20 considered instances. Besides, it is important to note that the number of "−" indicating that the EOP-DHS$_{MF}$ algorithm performs significantly better than EOP-DHS algorithm is equal to zero.

5.3 EOP-DHS approach performances discussion and extended experimental investigation

Globally considered, presented results indicate that the proposed EOP-DHS algorithm is a simple and highly competitive algorithm for the resolution of FJSSP on well-known BRdata and Fdata problem instances. Actually, the approach at least competes effectively or outperforms the entire considered comparative set of recently deployed metaheuristic approaches for the FJSSP. In order to permit a fair consideration of this result, it is important to relativize the conceptual and algorithmic simplicity of the proposed EOP-DHS algorithm to the sophistication of most of the considered stat of the art comparative approaches. Indeed, these approaches often incorporate in addition to a specific initialization scheme, domain-specific local search components, sophisticated critical path-based neighborhood search, or the two mechanisms in the same time. This fact is clearly susceptible to introduce a significant implementation complexity of these approaches. Besides, the utilization of dedicated heuristic initialization procedures generally introduces a number of additional algorithm parameters that have to be tuned, which is a complex task in the context of non-deterministic optimization.

According to our appreciation, the effective search behavior of the proposed EOP-DHS algorithm is clearly in line with its developed operational procedures that maintain a correct balance between its explorative and exploitative capabilities overall the search process: first, the EOP-DHS algorithm exploits a solutions decoding scheme that generates an active schedule at each evaluation step. Secondly, the EOP-DHS algorithm does not use any specific initialization scheme, what foster a high diversity of the generated initial population and avoid from a guided convergence of the algorithm to a specific region of the search space. Besides, this strategy allows a free-parameters HM initialization scheme that avoids from any inconsistent behavior of the search process due to a poor choice of these parameters. Thirdly, by using an improvisation operator that probabilistically exploits at each iteration a large number of the solutions stored in the HM, the algorithm guaranties, as it have been noted in [34, 35], an appropriate balance between exploration and exploitation ability of the search space. Finally, exploitation and exploration abilities of the EOP-DHS approach are enhanced by the use of MIM operator that exploits the correlation that often exist between maximum machines workload and Makespan criteria. Actually, as indicated in [51], maximum machines workload and Makespan are generally linearly correlated what implies a minimization of the Makespan when reducing maximum

Table 5 Results of the OP-DHS and comparison with the EOP-DHS on BRdata and Fdata

Inst.	OP-DHS		EOP-DHS		R+	R −	Ties	p-Value	
	AV(Cm)	BCm	AV(Cm)	BCm					
MFJS1	468,14	**468**	468,04	**468**	35	10	41	9,60E-02	/=
MFJS2	464,57	**446**	454,92	**446**	767,5	135,5	8	8,17E-05	/+
MFJS3	483,55	**466**	474,35	**466**	666	154	9	1,00E-03	/+
MFJS4	576,8	**554**	565,04	**554**	969.5	111,5	3	2,77E-06	/+
MFJS5	530	**514**	525,2	**514**	350.5	244,5	15	3,80E-01	/=
MFJS6	641,69	**634**	635,2	**634**	129,5	23,5	36	6,95E-03	/+
MFJS7	929,14	**879**	906,02	**879**	988	338	1	4,00E-03	/+
MFJS8	936,9	**884**	925,31	**884**	775,5	449,5	3	1,00E-01	/=
MFJS9	1142,12	1099	1118,96	1059	985,5	190,5	2	7,08E-05	/+
MFJS10	1285,24	1251	1274,35	1215	814	220	5	1,25E-03	/+
Mk01	40,65	**40**	40	**40**	153	0	33	2,93E-04	/+
MK02	27,18	27	26,73	**26**	604	26	15	2,21E-06	/+
MK03	204	**204**	204	**204**	0	0	50	1,00E+00	/=
MK04	62,67	**60**	61,67	**60**	499	167	14	8,00E-03	/+
MK05	173,92	173	172,98	**172**	378	0	23	8,30E-06	/+
MK06	64,31	61	63,92	61	476	265	12	1,40E-01	/=
MK07	143,92	140	142,04	140	928,5	106,5	5	4,15E-06	/+
MK08	523	**523**	523	**523**	0	0	50	1,00E+00	/=
MK09	307,06	**307**	307	**307**	1	0	49	1,00E+00	/=
MK10	220,12	213	217,63	211	920	115	5	4,00E-06	/+
Number of /=									**7**
Number of /-									**0**
Number of /+									**13**

Makespan values in bold indicate that this is the best known solution obtained

machines workload. The MIM operator exploits this fact and acts probabilistically as an intensification operator. Besides by altering the solution generated by the improvisation operator, MIM operator behaves probabilistically also as an explorative search component.

Comparatively to the EOP-DHS approach, different metaheuristic algorithms have been investigated in this work. Considering BRdata benchmarking set as a basis for the following analysis (Table 2), the population-based BBO algorithm presented in [17] showed relatively poor performances comparatively to the overall set of considered approaches. According to our appreciation, this fact is probably due, among others, to the adopted population initialization approach. Actually, in BBO algorithm the affectation part of the solutions are initialized using dedicated rules that mostly foster the generation of solutions with minimum global workload (total sum of processing times for all machines), where it is showed in [51] that for FJSSP, global workload and Makespan are generally inversely correlated. On the other side, TSPCB approach [9] uses a sophisticated initial solution generation scheme based on different rules particularly articulated around machines workloads balancing and minimization. Besides, an efficient neighborhoods structure for machine assignment is used. However, the approach fails to obtain effective solutions for Mk04, MK06, and Mk10 problems and showed a lack of exploration ability, what is generally a difficult task within the context of trajectory-based metaheuristic approaches, particularly for hard and computationally expensive problems (such as MK04, MK06, MK07. MK09, Mk10). We note also that the TSPCB approach does not exploit any active schedule-based solutions decoding scheme. Moreover, GA of Pizzella [14] exploits a sophisticated initial population generation scheme based on global workload balancing and an Intelligent Mutation Operator (IMO) that acts probabilistically on each solution to balance maximum workload. GA showed relatively good search ability for the considered dataset but fail in getting high quality solutions for MK06, MK09, and MK10 problems, which is probably due to a lack of exploration ability of the algorithm. Possible reasons for that, lies in the fact that IMO is used only as an intensification operator and obtained solution are considered only if they improve solutions Makespan. Besides, as for TSPCB algorithm, we also note that the GA do not exploit any active schedule-based solutions decoding scheme.

Above presented analysis of the performances of various metaheuristic approaches previously applied to the FJSSP considered mainly approaches with a relatively simples search procedures. For comparative purpose, different other highly sophisticated resolution processes have been considered. Actually, MA [26], PVNS [18], and ABC approach [21] achieved globally high quality results for BRdata benchmarking instances and obtained an MRD (mean relative deviation from the best known solution set in Table 5) that is inferior to 2%. This appreciated superior search ability is certainly due to various facts related to operational procedures embedded within these approaches and proves that they clearly achieved a proper balance between exploitation and exploration abilities of the search space. However, some search components are of particular interest; in ABC approach a continuous updating mechanism of the population is proposed to enrich the searching behavior and avoid the premature convergence of the algorithm. MA uses a novel Priority-based fitness function (PBFF) during selection. Accordingly, when ties occur between individuals, a priority scheme is considered to direct the search process considering the lower maximum workload. Authors stressed that the solution which has lower maximum workload is more likely to be the direction to the optimal solution. Besides, a maximum workload-based strategy is also considered in PVNS within the adopted Neighborhood structures. Conceptually, proposed EOP-DHS approach for FJSSP adopts a great part of these search components in an effective and simple optimization process.

Finally, in order to provide a fair consideration of the performances of the proposed EOP-DHS algorithm for the resolution of the FJSSP, an extended experimental investigation have been endeavored on series of 168 benchmarking problems with actual complexity: The data set from Dauzére-Pérés and Paulli [56] (DPdata), the data set from Barnes and Chambers [57] (BCdata), and the data set from Hurink et al. [58] that contains three sets of benchmark problems: Hurink Edata, Hurink Rdata, and Hurink Vdata.

Table 6 reports computational performance of the proposed EOP-DHS algorithm comparatively to different metaheuristic approaches deployed for the resolution of the FJSSP and tested using the adopted extended benchmarking data sets: The recent and highly superior Hybrid genetic and tabu search Algorithms (HA) of Li and Gao [59], the hybrid Genetic Algorithm (hGA) of Al-Hinai et al. [60], and previously investigated Hybrid HS (HHS) algorithm of Yuan et al. [47], PVNS algorithm (PVNS) of Yazdani et al. [18], and the GA of Pezzella et al. [14]. Particularly, Table 6 reports for each approach and dataset the MRD from the best-known lower bound obtained over 50 runs and the Global Relative Deviation (GRD) computed for all datasets.

Obtained results clearly confirm previously appreciated highly competitive behavior of the EOP-DHS algorithm for the resolution of FJSSP, extending the conclusions to realistically dimensioned and computationally expensive datasets. Indeed, the approach obtained a GRD from the best-known lower bound of just 6.67% that represent a significant performance comparatively to the superior HA, and show a clear superior search behavior comparatively to other approaches.

5.4 Classical JSSP experimental investigation

Final experimentation subsection, deals with the investigation of the adopted operations permutation-based HS framework initially proposed in [50] for the resolution of the classical JSSP with Makespan criterion. Thus, 27 problems of a generally affirmed intractability taken from the OR-Library [61] are considered: 2 problems from the Fisher and Thompson instances (ft10, ft20) and 25 problems from Lawrence instances (LcA16-LA40). Algorithm parameters are set empirically as follow: MCR = 0.97, PAR = 0.08, NI = 500 000 and HMS = 100.

Table 7 reports computational achievement of OP-DHS algorithm in term of best obtained Makespan (BCm), Average Makespan (*AvCm*) and average computational time (*Av*(*T*)) for 30 independent runs (last three columns). For each problem, the name, the dimension, and the BKS are reported in the first three columns. For performances comparison purpose, Table 5 reports also best obtained results for three metaheuristics approaches from the literature (columns 4, 5 and 6): The Bat Algorithm (BA) approach investigated in [32], the Hybrid Genetic Algorithm/Local Search algorithm (Hybrid) proposed in [62] and the New Island Model Genetic Algorithm (NIMGA) presented in [29]. In addition, Average Makespan and average computational time for the NIMGA approach are reported in columns 7 and 8.

Table 7 results show that OP-DHS algorithm to the JSSP depicts either a superior or a highly competitive searching behavior comparatively to the investigated approaches. Indeed, achieving a *BCm* and an *MRD* respectively equal to 12 and 1.03%, OP-DHS algorithm clearly outperforms the BA that obtains respectively 0 and 1.57% for the considered measures. We note that the basic BA has been chosen as a comparative algorithm because, not using any complementary improvement operators, it is considered from the same algorithmic class of EOP-DHS approach. The results also show that OP-DHS algorithm outperform GA/TS approach that attains an MRD of 3,84% and a BCm equal to 5. The GA/TS approach represents an example of a sophisticated optimization method using a hybrid of a genetic algorithm and a Tabu Search. Besides, comparatively to the effective NIMGA, obtained average Makespan and computational

Table 6 Results of EOP-DHS on extended datasets and comparison with different metaheuristic proposals

Data Set	N. Prob.	HA (%)	EOP-DHS (%)	PVNS (%)	hGA (%)	GA (%)
DPdata	18	1,82	3,43	5,11	6,83	7,63
BCdata	21	22,38	23,74	26,66	24,74	29,56
Hdata						
Edata	43	2,21	3,01	3,86	3,92	6
Rdata	43	1,16	2,64	1,88	3,68	4,42
Vdata	43	0,08	0,52	0,42	0,8	2,04
GRD		5,53	6,67	7,59	7,99	9,93

time results of the EOP-DHS approach reflect a superior behavior both in term of robustness and efficiency for the resolution of JSSP. Indeed, EOP-DHS results are clearly superior to those of NIMGA for both average Makespan and computational time. It is important to mention that the parallel three populations-based NIMGA have been implemented in C++ and tested using a computing environment close to the one used for this study (a PC with 3.40 GHz Intel(R) Core(TM) i7-3770 CPU and 8.00GB).

Thus, considering obtained results and the admitted intractability of most of the investigated benchmarking instances, it can be affirmed that the OP-DHS approach

Table 7 Results of EOP-DHS for the JSSP and comparison with different metaheuristic proposals

Inst.	N × M	B K S	BA	GA/TS	NIMGA			EOP-DHS		
					BCm	*AvCm*	*Av(T)*	*BCm*	*AvCm*	*Av(T)*
ft10	10 × 10	*930*	951	953	**930**	951,85	76,15	937	943,95	38,56
ft20	20 × 5	*1165*	1177	1192	1173	1186,55	84,62	**1165**	1185,05	34,2
la16	10 × 10	*945*	965	959	946	954,05	50,07	**945**	968,55	35,07
la17	10 × 10	*784*	794	792	**784**	784,75	29,67	**784**	784,25	8,41
la18	10 × 10	*848*	858	857	**848**	853,5	32,78	**848**	854,65	21,52
la19	10 × 10	*842*	852	860	**842**	846,55	14,2	850	866,35	35,14
la20	10 × 10	*902*	912	907	907	909,65	73,69	907	907,5	10,23
la21	15 × 10	*1046*	1066	1097	1058	1086,25	100,33	1062	1073,6	23,84
la22	15 × 10	*927*	944	980	937	956,25	101,42	935	942,9	18,58
la23	15 × 10	*1032*	1042	**1032**	**1032**	1032,2	24,56	**1032**	1032,15	6,41
la24	15 × 10	*935*	970	1001	947	970,45	72,34	967	969,3	14,59
la25	15 × 10	*977*	989	1031	992	1013,7	99,19	992	1002,45	39,88
la26	20 × 20	*1218*	1228	1295	**1218**	1240,1	119,01	**1218**	1221,65	85,74
la27	20 × 20	*1235*	1256	1306	1269	1296,2	131,83	1265	1279,01	74,63
la28	20 × 20	*1216*	1227	1302	1247	1265,45	123,18	1227	1243,2	61,27
la29	20 × 20	*1152*	1184	1280	1241	1261,8	124,61	1191	1225,65	44,28
la30	20 × 20	*1355*	1365	1406	**1355**	1357,45	84,2	**1355**	1355	23,19
la31	30 × 10	*1784*	1794	**1784**	**1784**	1784	6,12	**1784**	1784	14,7
la32	30 × 10	*1850*	1871	**1850**	**1850**	1850	9,66	**1850**	1850	32,41
la33	30 × 10	*1719*	1739	**1719**	**1719**	1719	9,98	**1719**	1719	51,93
la34	30 × 10	*1721*	1731	1758	**1721**	1723	55,24	**1721**	1721	62,17
la35	30 × 10	*1888*	1919	**1888**	**1888**	1888,55	27,1	**1888**	1888	14,1
la36	15 × 15	*1268*	1291	1357	1293	1316,45	110,98	1291	1298,9	56,42
la37	15 × 15	*1397*	1425	1494	1439	1460,2	138,27	1437	1448,05	50,06
la38	15 × 15	*1196*	1223	1338	1222	1266,65	138,48	1228	1241,2	38,13
la39	15 × 15	*1233*	1256	1343	1259	1275,35	137,2	1253	1260,5	71,49
la40	15 × 15	*1222*	1252	1311	1246	138,78	1252	1258,75	53,85	
#BCm			0	5	12			12		
MRD(%)			1,57	3,84	1,13			1,03		

Makespan values in bold indicate that this is the best known solution obtained depicts a highly competitive behavior both in term of effectiveness and efficiency for the resolution of the JSSP with Makespan criterion. That particular and complementary result confirms specially the effectiveness of the adopted HS framework and used improvisation scheme for the resolution of hard combinatorial scheduling problems.

6 Conclusion and future works

In this paper, an Effective Operations Permutation-based Discrete Harmony Search approach was proposed to solve the FJSSP with Makespan criterion. Proposed EOP-DHS approach adopts an integrated "affectation-sequencing" two-part representation of the solution harmony and a dedicated improvisation operator particularly adapted to the integer-valued and operations permutation-based used coding scheme. A complementary search operator, the Modified Intelligent Mutation (MIM) is integrated to the adopted framework in order to enhance its overall search ability by probabilistically balancing maximum machine workload during the overall search process. MIM operator allows essentially maintaining and enhancing the reciprocal equilibrium of diversification and intensification abilities of the proposed EOP-DHS algorithm. For performances assessment purpose, EOP-DHS approach has been experimented on a set of 188 test problems and compared with a representative spectrum of metaheuristic resolution approaches recently formulated for the FJSSP. Obtained and discussed results indicate that the proposed algorithm is effective for the resolution of the FJSSP. In addition, a complementary experimental procedure proving the effectiveness of the adopted permutation-based HS scheme for the resolution of hard combinatorial optimization problems exemplified by the JSSP has been carried on.

Enhancement of the approach considering its efficiency will be considered in future works. Furthermore, an extended examination of the adopted Permutation-based improvisation framework for the resolution of other hard combinatorial problems will be of a notable usefulness.

References

1. Brucker P, Schlie R (1990) Job-shop scheduling with multi-purpose machines. Computing 45(4):369–375
2. Garey M, Johnson D, Sethi R (1976) The complexity of flowshop and jobshop scheduling. Math Oper Res 1:117–129
3. Jain A, Meeran S (1999) Deterministic job-shop scheduling: past, present and future. Eur J Oper Res 113(2):390–434
4. Fattahi P, Saidi-Mehrabad M, Jolai F (2007) Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. J Intell Manuf 18:331–342
5. Brandimarte P (1993) Routing and scheduling in a flexible job shop by tabu search. Ann Oper Res 41(3):157–183
6. Dauzère-Pérès S, Paulli J (1997) An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. Ann Oper Res 70:281–306
7. Mastrolill M, Gambardella L (2000) Effective neighborhood functions for the flexible job shop problem. J Sched 3:3–20
8. Saidi-Mehrabad M, Fattahi P (2007) Flexible job shop scheduling with tabu search algorithms. Int J Adv Manuf Technol 32(5-6):563–570
9. Li J, Pan QK, Suganthan P, Chua T (2011) A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem. Int J Adv Manuf Technol 52(5-8):683–697
10. Kacem I, Hammadi S, Borne P (2002) Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling. IEEE Trans Syst Man Cybern 32:1–13
11. Gen M, Gao J, Lin L (2009) Multistage-based genetic algorithm for flexible job-shop scheduling problem. In: Intelligent and evolutionary systems. Springer Berlin Heidelberg, pp 183–196
12. Gao J, Gen M, Sun L, Zhao X (2007) A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems. Comput Ind Eng 53(1):149–162
13. Gao J, Sun L, Gen M (2008) A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. Comput Oper Res 35:2892–2907
14. Pezzella F, Morganti G, Ciaschetti G (2008) A genetic algorithm for the flexible job-shop scheduling problem. Comput Oper Res 35(10):3202–3212
15. Zhang G, Gao L, Shi Y (2011) An effective genetic algorithm for the flexible job-shop scheduling problem. Expert Syst Appl 38(4):3563–3573
16. Wu X, Wu S (2015) An elitist quantum-inspired evolutionary algorithm for the flexible job-shop scheduling problem. J Intell Manuf, pp 1–17. https://doi.org/10.1007/s10845-015-1060-6
17. Rahmati SH, Zandieh M (2012) A new biogeography-based optimization (BBO) algorithm for the flexible job shop scheduling problem. Int J Adv Manuf Technol 58(9–12):1115–1129
18. Yazdani M, Amiri M, Zandieh M (2010) Flexible job-shop scheduling with parallel variable neighborhood search algorithm. Expert Syst Appl 37(1):678–687
19. Bagheri A, Zandieh M, Mahdavi I, Yazdani M (2010) An artificial immune algorithm for the flexible job-shop scheduling problem. Futur Gener Comput Syst 26(4):533–541
20. 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
21. Wang L, Zhou G, Xu Y, Wang S, Liu M (2012) An effective artificial bee colony algorithm for the flexible job-shop scheduling problem. Int J Adv Manuf Technol 60(1-4):303–315
22. Nouiri M, Bekrar A, Jemai A, Niar S, Ammari A (2015) An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. J Intell Manuf, pp 1–13. https://doi.org/10.1007/s10845-015-1039-3
23. Teekeng W, Thammano A, Unkaw P, Kiatwuthiamorn J (2016) A new algorithm for flexible job-shop scheduling problem based on particle swarm optimization. Artif Life Robot 21(1):18–23
24. Roshanaei V, Azab A, ElMaraghy H (2013) Mathematical modelling and a meta-heuristic for flexible job shop scheduling. Int J Prod Res 51(20):6247–6274
25. Chaudhry I, Khan (2016) A research survey: review of flexible job shop scheduling techniques. Int Trans Oper Res 23(3):551–591
26. Kobti Z (2012) A memetic algorithm for job shop scheduling using a critical-path-based local search heuristic. Memet Comput 4(3):231–245

27. Çaliş B, Bulkan S (2015) A research survey: review of AI solution strategies of job shop scheduling problem. J Intell Manuf 26(5):961–973
28. Zhao F, Zhang J, Zhang C, Wang J (2015) An improved shuffled complex evolution algorithm with sequence mapping mechanism for job shop scheduling problems. Expert Syst Appl 42(8):3953–3966
29. Kurdi M (2016) An effective new island model genetic algorithm for job shop scheduling problem. Comput Oper Res 67:132–142
30. Asadzadeh L (2015) A local search genetic algorithm for the job shop scheduling problem with intelligent agents. Comput Ind Eng 85:376–383
31. Wang X, Duan H (2014) A hybrid biogeography-based optimization algorithm for job shop scheduling problem. Comput Ind Eng 73:96–114
32. Dao T, Pan T, Pan J (2015) Parallel bat algorithm for optimizing makespan in job shop scheduling problems. J Intell Manuf, pp 1–12
33. Geem ZW, Kim JH, Loganathan G (2001) A new heuristic optimization algorithm: harmony search. Simulation 76(2):60–68
34. Manjarres D, Landa-Torres I, Gil-Lopez S, Del Ser J, Bilbao M, Salcedo-Sanz S, Geem Z (2013) A survey on applications of the harmony search algorithm. Eng Appl Artif Intell 26(8):1818–1831
35. Kong X, Gao L, Ouyang H, Li S (2015) A simplified binary harmony search algorithm for large scale 0–1 knapsack problems. Expert Syst Appl 42(12):5337–5355
36. Lee K, Geem Z, Lee S, Bae KW (2005) The harmony search heuristic algorithm for discrete structural optimization. Eng Optim 37:663–684
37. Mahdavi M, Fesanghary M, Damangir E (2007) An improved harmony search algorithm for solving optimization problems. Appl Math Comput 188:1567–1579
38. Moh'd Alia O, Mandava R (2011) The variants of the harmony search algorithm: an overview. Artif Intell Rev 36(1):49–68
39. Zou D, Gao L, Wu J, Li S, Li Y (2010) A novel global harmony search algorithm for reliability problems. Comput Ind Eng 58(2):307–316
40. Pandi V, Panigrahi B (2011) Dynamic economic load dispatch using hybrid swarm intelligence based harmony search algorithm. Expert Syst Appl 38(7):8509–8514
41. Pan QK, Suganthan PN, Liang JJ, Tasgetiren MF (2011) A local-best harmony search algorithm with dynamic sub-harmony memories for lot-streaming flow shop scheduling problem. Expert Syst Appl 38(4):3252–3259
42. Wang L, Pan QK, Tasgetiren M (2011) A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem. Comput Ind Eng 61(1):76–83
43. Wang L, Pan Q, Tasgetiren M (2010) Minimizing the total flow time in a flow shop with blocking by using hybrid harmony search algorithms. Expert Syst Appl 37(12):7929–7936
44. Chen J, Pan QK, Wang L, Li J (2012) A hybrid dynamic harmony search algorithm for identical parallel machines scheduling. Eng Optim 44(2):209–224
45. Gao K, Pan QK, Li JQ (2011) Discrete harmony search algorithm for the no-wait flow shop scheduling problem with total flow time criterion. Int J Adv Manuf Technol 56(5-8):683–692
46. Liu L, Zhou H (2013) Hybridization of harmony search with variable neighborhood search for restrictive single-machine earliness/tardiness problem. Inf Sci 226:68–92
47. Yuan Y, Xu H, Yang J (2013) A hybrid harmony search algorithm for the flexible job shop scheduling problem. Appl Soft Comput 13(7):3259–3272
48. Gao K, Suganthan P, Pan QK, Chua T, Cai T, Chong C (2016) Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives. J Intell Manuf 27(2):363–374
49. Gao K, Suganthan P, Pan QK, Tasgetiren M (2015) An effective discrete harmony search algorithm for flexible job shop scheduling problem with fuzzy processing time. Int J Prod Res 53(19):5896–5911
50. Gaham M, Bouzouia B, Achour N, Tebani K (2016) Meta-heuristics approaches for the flexible job shop scheduling problem Metaheuristics for Production Systems. Springer International Publishing, pp 285–314
51. Chiang T, Lin H (2013) A simple and effective evolutionary algorithm for multiobjective flexible job shop scheduling. Int J Prod Econ 141(1):87–98
52. Wilcoxon F (1945) Individual comparisons by ranking methods. Biom Bull 1(6):80–83
53. Derrac J, García S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol Comput 1(1):3–18
54. Lu C, Xiao S, Li X, Gao L (2016) An effective multi-objective discrete grey wolf optimizer for a real-world scheduling problem in welding production. Adv Eng Softw 99:161–176
55. Lu C, Gao L, Li X, Xiao S (2017) A hybrid multi-objective grey wolf optimizer for dynamic scheduling in a real-world welding industry. Eng Appl Artif Intell 57:61–79
56. Dauzère-Pérès S, Paulli J (1997) An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. Ann Oper Res 70:281–306
57. Barnes JW, Chambers JB (1996) Flexible job shop scheduling by tabu search. Graduate program in operations research and industrial engineering. Technical Report ORP 9609. University of Texas, Austin. http://www.cs.utexas.edu/users/jbc/
58. Hurink J, Jurisch B (1994) Tabu search for the job-shop scheduling problem with multi-purpose machines. Oper Res Spektrum 15(4):205–215
59. Li X (2016) An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. Int J Prod Econ 174:93–110
60. Al-Hinai N (2011) An efficient hybridized genetic algorithm architecture for the flexible job shop scheduling problem. Flex Serv Manuf J 23(1):64–85
61. Beasley J (1990) OR-Library: distributing test problems by electronic mail. J Oper Res Soc 41(11):1069–1072
62. Ombuki B, Ventresca M (2004) Local search genetic algorithms for the job shop scheduling problem. Appl Intell 21(1):99–109