A hybrid iterated local search metaheuristic for the flexible job shop scheduling problem

Dayan de C. Bissoli
Graduate School of Computer Science (PPGI)
Federal University of Espírito Santo (UFES)
Vitória, Brazil
dayan.bissoli@ufes.br

André R. S. Amaral
Graduate School of Computer Science (PPGI)
Federal University of Espírito Santo (UFES)
Vitória, Brazil
amaral@inf.ufes.br

***Abstract*—In the flexible job shop scheduling problem (FJSP) we have a set of jobs and a set of machines. A job is characterized by a set of operations that must be processed in a predetermined order. Each operation can be processed in a specific set of machines and each of these machines can process at most one operation at a time, respecting the restriction that before starting a new operation, the current one must be finished. Scheduling is an assignment of operations at time intervals on machines. The classic objective of the FJSP is to find a schedule that minimizes the completion time of the jobs, called makespan. Considering that the FJSP is an NP-hard problem, solution methods based on metaheuristics become a good alternative, since they aim to explore the space of solutions in an intelligent way, obtaining high-quality but not necessarily optimal solutions at a reduced computational cost. Thus, to solve the FJSP, this article describes a hybrid iterated local search (HILS) algorithm, which uses the simulated annealing (SA) metaheuristic as local search. Computational experiments with a standard set of instances of the problem indicated that the proposed HILS implementation is robust and competitive when compared with the best algorithms of the literature.**

***Index Terms*—flexible job shop scheduling, hybrid metaheuristic, iterated local search, simulated annealing**

I. Introduction

Scheduling is one of the most important problems in production planning and control systems. The classical job shop scheduling problem (JSP) is one of the most difficult problems in this area. The JSP aims to schedule a set of jobs by means of a set of machines so as to optimize some objective function (OF). Each job consists of a sequence of operations, which must be processed uninterruptedly on specific machines [1]–[3].

When considering real cases, the JSP has the limitation that each operation can only be processed on a single machine. Therefore, an extension called flexible job shop scheduling problem (FJSP) arises, in which it is possible that an operation of a given job be processed in alternative machines. Thus, in addition to having the characteristic scheduling problem (JSP), the FJSP also has a routing sub-problem in which one has to select an appropriate machine among those available to process each operation [4], [5].

Figure 1 shows an example of a productive environment based on the FJSP, where the arrows indicate the possible paths of the jobs through the machines, thus, illustrating the alternatives for processing the operations. The assignment of each operation to an alternative machine implies finding for each job a path which starts to the left of the figure and ends to the right. In addition to defining the assignments, it is still necessary to determine the sequences of the operations to be processed in each machine. The description of the environment represented in Figure 1 is intended to clarify how the FJSP jobs can go through alternative ways through the machines:

- Job 1 has 4 operations and its first operation can be processed alternately on machines 1 or 2. The second operation of job 1 must be processed on the machine 1. In this way, if the first operation is also processed on machine 1, re-circulation occurs. The third operation of job 1 must be performed on machine 3 and the last operation on machine 4.
- Job 2 has 3 operations, with only the second one having processing alternatives, machines 3 or 4.
- Job 3 also has 3 operations and its second operation can be processed either on machine 2 or machine 3.

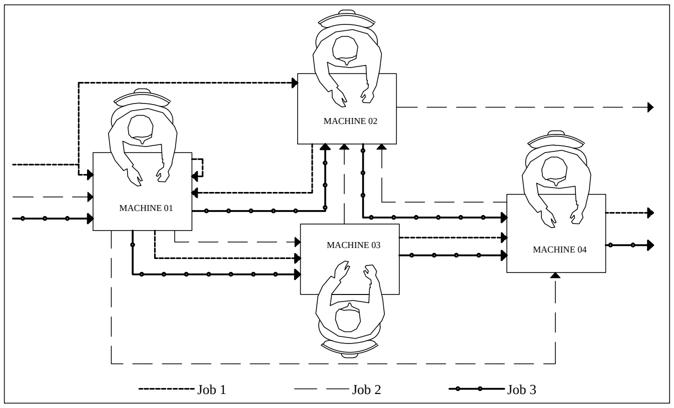

Fig. 1. Representation of a FJSP.

Because the FJSP is an NP-hard problem, the application of exact methods becomes limited [6]. In this way, in this work in order to solve the FJSP, considering makespan as the objective function, we implemented a hybrid algorithm (HILS) based on iterated local search (ILS) and using simulated annealing (SA) as a local search method. Computational experiments with a standard set of instances for the problem indicated that the proposed implementation is efficient and provides high-quality solutions to the FJSP.

This study is organized as follows. The next section presents the related literature. Next, the problem modeling and the proposed metaheuristic are described. Then, computational experiments are reported followed by the conclusions.

II. Related Literature

The FJSP was proposed by Brucker and Schlie [7], who developed an algorithm, which solves an instance with two machines in polynomial time. Since then, several approaches have been proposed to solve the FJSP. Most of the works present heuristic and/or metaheuristic approaches [5], [8]. Whereas [9] applied a *branch-and-bound* algorithm for the FJSP and compared it with two metaheuristics based on *Tabu Search* (TS) and *Simulated Annealing* (SA). More recently, Previero [10] has employed two strategies to solve the FJSP. The first one is an exact *Branch-and-Cut* (B & C) algorithm with a variant that aggregates valid inequalities of [11] to a model proposed by [12]. The second approach is based on two metaheuristics: *Local Branching* (LB) and *Diversification, Refining and Tight-refining* (DRT). The results were compared with [12] and with those of the solver Gurobi [13]. In general, metaheuristic-based approaches performed better in relation to exact methods and only for 3 out of 59 instances tested did metaheuristics not obtain the best results.

Considering heuristic methods, [14] have proposed hierarchical approaches, in which the assignments of the operations to machines (routing) and the scheduling of the operations (scheduling) were studied separately. However, most of the works in the literature consider the two subproblems simultaneously, such as [15] and [16], which applied tabu search to the FJSP (in [16] a new re-attribution procedure of the operations was proposed). [17] presented two neighborhood structures to improve the *Tabu Search* (TS) algorithm proposed by [16].

An algorithm called *Effective Genetic Algorithm* (eGA) was developed by Zhang *et al.* [18] who adopted different strategies for selection, crossover and mutation. The computational results show that the eGA leads to the same or better computational time and solution quality compared with other genetic algorithms.

More recently, [19] presented an algorithm based on SA, with a mechanism of partial scheduling and a cooling mechanism, which is a function of the standard deviation. It was found that SA with the proposed mechanism, called *SA-Partial Controlled* (SA-PC), converges faster to good solutions than without the proposed mechanism.

A metaheuristic that combines *Scatter Search* and *Path-Relinking* with TS (SSPR) was developed by [20] for the FJSP. The authors sought to combine the diversification characteristic provided by Scatter Search and Path-Relinking with the intensifying nature of TS. This method was compared with the ones in the literature and obtained the best known values for 58 out of 178 considered instances.

Ishikawa *et al.* [21] implemented a hybrid evolutionary method called *Hierarchical Multi-Space Competitive Distributed Genetic Algorithm* (HmcDGA). The method presented competitive results, but with a high computational cost. A *Quantum Particle Swarm Optimization* algorithm (QPSO) was proposed by [22] to solve the FJSP. The mutation operator, a technique that originates from genetic algorithms, is used as a resource to help avoid premature convergence and to diversify the solution. Another tool used by QPSO to provide greater diversity and reduce computational cost in the search is the use of chaotic numbers (logistic map) instead of random numbers.

Gao *et al.* [23] proposed a *Discrete Harmony Search* (DHS) algorithm for the FJSP with multiple objectives. Posteriorly, a variation of DHS, called *Effective Operations Permutation-Based Discrete Harmony Search* (EOPDHS) was applied to the FJSP by [24]. The authors used an operator called the Modified Intelligent Mutation, in order to probabilistically balance the the maximum working time of the machines during the general search process. The results were compared with the literature and indicated that the proposed algorithm is effective for the solution of the FJSP.

Li *et al.* [25] developed for the FJSP a *Hybrid Artificial Bee Colony* (HABC) algorithm based on the *Artificial Bee Colony* (ABC) and TS algorithms. The roulette method and crossover operator for bees were implemented with the aim of improving population initialization and exploration, respectively. On average, the results presented by HABC were better than the results presented by the other well-known algorithms used in the comparison.

III. Problem formulation

The FJSP can be formulated as follows [5]: We are given a set of n jobs to be processed on m machines; the set of machines is denoted by: $M = \{M_1, M_2, ..., M_m\}$; job i consists of a sequence of n_i operations: $(O_{i,1}, O_{i,2}, ..., O_{i,n_i})$; each operation must be executed to finish the job; the execution of each operation j of a job $i(O_{i,j})$ requires a machine of a given set $M_{i,j}$; the time for the operation $O_{i,j}$ to be processed on machine M_k is $p_{i,j,k}$. The following assumptions are considered:

1) All machines are available at time $t = 0$.
2) All jobs are available at time $t = 0$.
3) Each operation must be processed by only one machine at a time.
4) There are no precedence constraints between the operations of different jobs; thus, the jobs are independent of each other.
5) An operation that started processing can not be interrupted.
6) Transport time between jobs and machines and time to set up the machine for the processing of a particular operation are included in the processing time.

The FJSP can be categorized into two subproblems [5]:

1) A *routing* subproblem in which one has to select an appropriate machine among those available to process each operation.

2) A *scheduling* subproblem in which the operations assignments are sequenced on all selected machines to obtain a feasible scheduling that minimizes a predefined objective.

Figure 2 illustrates a disjunctive graph of a FJSP with 3 jobs and 4 machines, which is an abstraction of the productive environment shown in Figure 1. Vertices 1 through 10 represent the operations. Vertices 0 and 11 correspond, respectively, to artificial operations of start and end of the scheduling. The vertices are connected by conjunctive and disjunctive arcs. *Conjunctive arcs* consecutively connect the operations of each job, indicating the order of precedence to be respected. Arcs that do not have definite orientation are called *disjunctive arcs*, which link operations to be performed on the same machine. It is noticed that disjunctive arcs for different machines may be incident on a vertex, which characterizes flexibility.

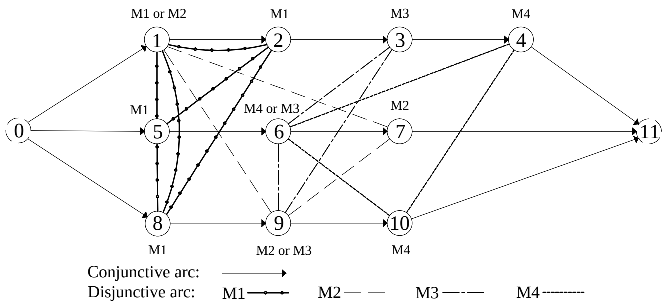

Fig. 2. Disjunctive graph: representation of a scheduling.

The FSJP graph defines a feasible solution if it is acyclic [16]. Figure 3 exemplifies a solution obtained by defining the orientations of the disjunctive arcs of the graph of Figure 2.

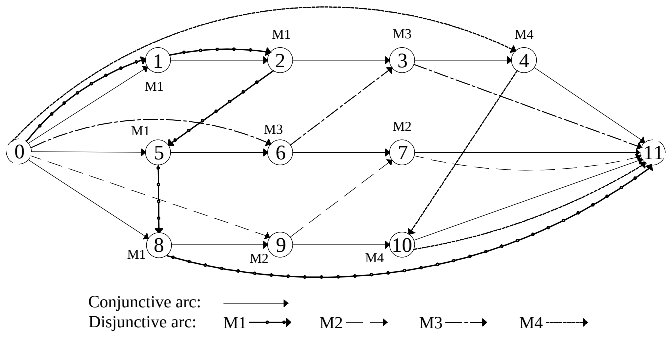

Fig. 3. FJSP solution.

A. Hybrid Iterated Local Search

To solve the FJSP, we developed an algorithm based on Iterated Local Search (ILS) [26]. The classical ILS considers that generating new initial solutions through the application of perturbations in the local optimal solution allows a better exploration of the neighborhood by local search. Algorithm 1 presents the generic pseudocode of HILS metaheuristic, using SA as local search.

Algorithm 1: HILS Pseudocode for FJSP.

```
input  : $ILS_{max}$
output : Solution $s$
1 $s \leftarrow$ InicitalSolution();
2 $s \leftarrow$ LS1($s$);
3 for $i \leftarrow 1$ to $ILS_{max}$ do
4     $s_i \leftarrow$ Pertubation($s, k_{max}$);
5     $s_i \leftarrow$ SimulatedAnnealing($s_i, SA_{max}$);
6     $s \leftarrow$ Acceptance ($s, s_i$);
7 end
```

After generating an initial solution s (line 1), HILS applies a local search to s (line 2), in order to get a local optimum solution and then starts the iterative process (lines 3-7). Until the limit of ILS_{max} iterations is reached, HILS performs a perturbation in the best solution s, to submit it to a local search algorithm and then evaluate it by an acceptance criterion. In this implementation, the parameter ILS_{max} is determined by a timeout. These steps are described below.

1) Initial Solution: The construction of the initial solution s is performed in a random way. The idea is based on randomly selecting the jobs to be added in s, and when there is flexibility, randomly choose one machine among the set of alternative machines.

2) Local Search 1: The Local Search 1 (LS1) implemented was based on the best improvement method [27] and is presented in Algorith 2. Its behavior consists of exploring the neighborhood of the current solution s. The strategy is divided into two stages (a) and (b):

(a) The first step (lines 3-13) consists of obtaining, at each iteration, a new solution s' when performing an exchange in the processing order of a job of a machine in the solution s. Thus, when selecting a certain machine, from the second job of the scheduling, the exchange is done with its neighbor on the left. After the exchange, the solution is reconstructed to the right of the exchange made. Then it is evaluated if there was improvement in the solution ($s' < s^*$). If so, the best solution is updated ($s^* \leftarrow s'$). After each exchange, s is returned ($s' \leftarrow s$) and the algorithm selects the next job on the right. The process repeats until all jobs of all machines of the solution s have been evaluated.

(b) At the end of step (a), the algorithm begins the exploration of the solution in the neighborhood due to flexibility (lines 15-34). The idea is for jobs that have flexibility to make all the possible exchanges between the alternative machines and for each exchange to explore the whole neighborhood of the machine inserted in the solution s' in the same way as in (a). To do this, select a job from the flexibility list (line 16), and in accordance with its alternative machine, the exchange is effected in s'. Then the solution is reconstructed as in (a). Subsequently, the algorithm evaluates whether $s' < s$, and if so, the best solution is updated ($s^* \leftarrow s'$).

At the end of each exchange, the solution is returned $s' \leftarrow s$ and select the next job $flexJobs[k+1]$ which has flexibility. This process is repeated until all flexibility is evaluated.

Algorithm 2: Local Search pseudocode.

```
input : $s$
1  $s^* \leftarrow s' \leftarrow s$;
2  $i \leftarrow 1$;
3  while $i <= m$ do
4      $j \leftarrow 2$;
5      while $j <= n_{m_i}$ do
6          exchange jobs $s'(m_i[j])$ and $s'(m_i[j-1])$ ;
7          rebuilds $s'$ ;
8          if $f(s') < f(s^*)$ then
9              $s^* \leftarrow s'$ ;
10         end
11         $s' \leftarrow s; j++$;
12     end
13 end
14 $k \leftarrow 1$;
15 while $k <= n_{flex}$ do
16     $j_{flex} \leftarrow flexJobs[k]$;
17     $i \leftarrow$ machine that will be allocated the $j_{flex}$ ;
18     destroy the solution $s'$ from right to left until the
         $j_{flex}$;
19     insert the $j_{flex}$ in machine $i$;
20     $j \leftarrow$ index of $j_{flex}$ in machine $i$ ;
21     rebuilds $s'$;
22     if $f(s') < f(s^*)$ then
23         $s^* \leftarrow s'; s \leftarrow s'$;
24     end
25     while $j <= n_{m_i}$ do
26         exchange jobs $s'(m_i[j])$ and $s'(m_i[j-1])$ ;
27         rebuilds $s'$ ;
28         if $f(s') < f(s^*)$ then
29             $s^* \leftarrow s'$;
30         end
31         $s' \leftarrow s; j++$;
32     end
33     $k++$;
34 end
35 return $s^*$
```

3) Local Search 2 - Simulated Annealing: In the proposed SA, we use three different ways to explore the neighborhood: N^1, N^2 and N^3, which are described below. The application of a local search (LS1) is another variation proposed in the classic SA.

The pseudocode of the proposed SA as local search is presented in Algorithm 3. SA starts at a temperature T_0, which is decremented according to the value of α (line 23), running until it reaches final temperature T_f. At each temperature there are SA_{max} iterations (lines 5-17). At each iteration of the SA a perturbation in the current solution takes place according to one of the three neighborhoods (N^1, N^2 or N^3), which are randomly selected, obtaining s' (lines 7-8). If s' is better than the overall solution, update s^* and apply the local search LS1 in this solution (lines 9-16). Otherwise, considering a given priority, the search continues from s' or s (lines 19-21). The SA_{max} parameter was configured based on [19], which defined the number of SA iterations as $I = 2 \times (m \times (n-1))$, being m the number of machines and n the number of jobs.

Algorithm 3: Pseudocode of SA with local search.

```
input : $T_0$, $T_f$, $\alpha$, $SA_{max}$, Solution $s$
output: Solution $s^*$
1  $s^* \leftarrow s$;
2  $T \leftarrow T_0$;
3  while $T > T_f$ do
4      $iter \leftarrow 0$ ;
5      while $iter < SA_{max}$ do
6          $iter \leftarrow iter + 1$ ;
7          $k \leftarrow random[1,3]$ ;
8          $s' \leftarrow N^k(s)$;
9          if $f(s') < f(s)$ then
10             $s \leftarrow s'$;
11             if $f(s') < f(s^*)$ then
12                 $s^* \leftarrow s'$;
13                 $s' \leftarrow$ LS1 ($s'$) ;
14                 if $f(s') < f(s^*)$ then
15                     $s^* \leftarrow s'$;
16                 end
17             end
18         end
19         else
20             $s \leftarrow s'$, with probability $e^{\frac{-(f(s')-f(s))}{T}}$;
21         end
22     end
23     $T \leftarrow T \times \alpha$;
24 end
25 return $s^*$
```

4) Neighborhood Structures: In order to search for better solutions, SA has the characteristic of exploring the neighborhood in an iterative way, making exchanges between neighbors. In the FJSP context, moves are performed by changing the routing (e.g. by moving an operation from one machine to another), or by changing a scheduling (e.g. by shifting an operation or exchanging two operations) [28], [29]. Thus, three ways of exploring the neighborhood were used: N^1, N^2 and N^3, which are described below.

a) Neighborhood Structure N^1: The strategy used in the neighborhood structure N^1 is based on exchanging positions between operations on a machine. For this, we randomly select a machine and then randomly select an operation among the ones allocated in the machine. Only the first operation on the selected machine does not enter the raffle. This is due to how the exchange is performed: when selecting an operation, the exchange move is always performed with its nearest neighbor on the left.

In the Figure 4 an N^1 exchange move is shown. Initially, Machine 2 (M2) was randomly selected. M2 processes, in the following order, the operations $O_{1,1}$, $O_{2,1}$, $O_{3,1}$ and $O_{4,1}$. Then, a new raffle is carried out, and the operation $O_{2,1}$ is selected. Then the exchange is made with the operation located to its left. Therefore, the updated sequence of operations processed in M2 is as follows: $O_{2,1}$, $O_{1,1}$, $O_{3,1}$ and $O_{4,1}$.

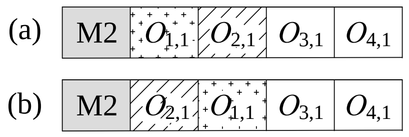

Fig. 4. Exchange move: N^1.

b) Neighborhood Structure N^2: The strategy used in the neighborhood structure N^2 is based on exploiting the flexibility. The method seeks to exchange an operation that has flexibility from one machine to another. For this, a raffle is initially realized to select an operation among those with flexibility. The machine on which the operation is allocated is identified. Then, randomly select another machine that can process it and then, exchange the operation from the current machine to the selected machine. The solution is repaired, in order to maintain its viability.

Figure 5 presents an N^2 exchange move. Initially the operation $O_{2,2}$ was randomly selected among those that have flexibility. It was identified that the $O_{2,2}$ operation is allocated on the M1 machine. It was found that the $O_{2,2}$ operation can be processed alternately in several machines, among which the machine M2 was randomly selected. In this way the $O_{2,2}$ operation was exchanged from machine M1 to machine M2.

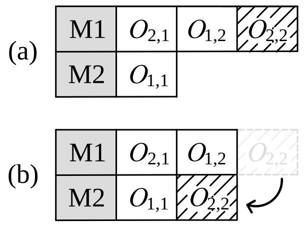

Fig. 5. Exchange move: N^2.

c) Neighborhood Structure N^3: The neighborhood structure N^3 is an extension of N^1, and is based on performing the exchange of operations considering the order in which they were inserted in the solution. For this, an auxiliary vector is used, which stores the history of the sequence of insertion of operations in the machines to build the solution. The N^3 exchange has the following characteristic:

1) In the data structure that stored the order of insertion of the operations in the machines, an operation X is randomly selected;
2) If the operation Y to the left of X does not belong to the same job as X, then the positions of X and Y are exchanged in the auxiliary vector;
3) Otherwise, operation Y is designated as operation X. Return to step 2.

Figure 6 illustrates a N^3 move when precedence is violated. The $O_{2,2}$ operation was randomly selected. In the auxiliary structure, which stored the order of insertion of the operations, a search was made and it was verified that the operation $O_{2,1}$ is on its left. It has been verified that the operations belong to the same job and there will be a violation of the precedence restriction if the exchange was made. In this way, the next operation to the left of $O_{2,2}$ was selected: $O_{2,1}$. As in this case there is no breach in precedence, the exchange was made between $O_{2,1}$ and $O_{1,1}$.

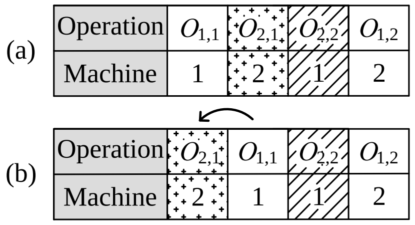

Fig. 6. N^3 exchange move - when the order of precedence is violated.

5) Perturbation: As perturbation strategy, is performed k_{max} exchanges movements in s. In order not to make the deterministic search, at each iteration a random selection of N^1, N^2 or N^3. The Algorithm 4 shows the pseudocode of this step.

Algorithm 4: Perturbation of the HILS to the FJSP.

```
  input  : $s, k_{max}$
  output : Solution $s$
1 for $i \leftarrow 1$ to $k_{max}$ do
2     $k \leftarrow random[1, 3]$ ;
3     $s \leftarrow N^k(s)$;
4 end
```

6) Acceptance Criteria: For HILS accept an intermediary solution s_i, it must be better than the current solution s. Thus, it is evaluated whether $f(s_i) < f(s)$, and if so, the solution s is discarded and the search continues in s_i. Otherwise, the search continues in s.

IV. Computational Experiments

A. Experimental Setup

To illustrate the effectiveness of the HILS algorithm described in this article, we consider traditional sets of FJSP instances found in the literature: HUdata [15], BRdata [14], DPdata [16] and BCdata [30], providing a total of 178 instances with different sizes and flexibility (the average number of possible machines per operation).

The HILS algorithm was implemented in C++ language and compiled with the Visual C++ compiler. The experiments were run on an Intel Core i5-750 2.67GHz computer with 8GB of RAM, using Windows 7 Ultimate 64-bit operating system. The HILS algorithm was applied 10 times for each instance, and the best and the average value of the makespan value found was recorded.

In order to calibrate HILS, tests were performed with three small, medium and large instances (mk01, 10a and mk10, respectively). The tests were based on the methodology used in [31], in which some fair values for the parameters are used based on the development of the SA, then some values are kept fixed, while others parameters are changed. The Table I presents the final result of the calibration process.

TABLE I
DEFINITION OF HILS PARAMETERS.

Parameter	Defined value	Values tested
T_0	100	100, 1000 and 10000
α	0.998	-
T_f	0.001	-
SA_{max}	I	I, $I \times 10$ and $I \times 100$
k_{max}	$I \times 0.1$	10, $I \times 0.1$ and 100
$Time_{max}$	1800	600, 1200 and 1800

In order to achieve greater diversification and intensification of the search, at the end of each iteration of HILS, the parameters SA_{max} and k_{max} have their values doubled (limited to 30000). Thus, with each new iteration of HILS, the solution s will suffer a greater number of perturbations, which tends to diversify the search, and when necessary, SA will perform a greater number of iterations at each temperature, intensifying the search.

B. Computational results and comparison to the literature

Is presented in Table II the summary of the results of HILS algorithm for the HUdata set of instances, across each of the three subsets of 43 instances: edata, rdata and vdata. For each subset of instances, Table II shows the average value of the best makespan (MS), the average makespan obtained in ten runs of the algorithms for each instance (Av.) and the respective average time required for HILS to obtain the makespan in ten runs, in seconds (T(s)).

TABLE II
SUMMARY OF RESULTS: HUDATA.

	TS [17]			HILS		
Set	*MS*	*Av.*	*T(s)*	*MS*	*Av.*	*T(s)*
HUdata/edata	1003.8	1004.8	2.8	1003.2	1004.9	289.3
HUdata/rdata	909.9	911.4	4.2	909.0	910.7	362.3
HUdata/vdata	895.6	895.9	4.3	895.4	895.7	243.2
Mean	936.4	937.4	3.7	935.9	937.1	298.3

For the HUdata instance set, the results are subdivided into $edata$, $vdata$ and $rdata$. The results of HILS are compared with those generated by the tabu search (TS) metaheuristic [17], considering that only in that work, among the ones evaluated, the results for the HUdata the set of instances are detailed. The HILS algorithm obtained a new best known solution (BKS) for several instances. The results of the HILS algorithm for the instances that generated a new BKS are reported in Tables III, IV and V. The tables show, for each instance (I), the best makespan value (MS) obtained in ten runs of the algorithms, the average of the MS in the 10 executions (Av.), the average time required to get the best MS, in seconds (T(s)) and the standard deviation (SD). Considering that the good results generated by TS are in almost two decades without improvement, even with the FJSP being quite approached in that period, it is emphasized that were obtained in HUdata set of instances, 28 new BKS. In order to do so, it was decided to explore the ILS diversification / intensification power for a longer period of time, which made it possible to achieve new BKS.

TABLE III
NEW BKS GENERATED BY HILS FOR THE SET $edata$.

	TS [17]				HILS			
I	*MS*	*Av.*	*T(s)*	*SD*	*MS*	*Av.*	*T(s)*	*SD*
la24	909	911.6	3.9	2.6	**908**	909.6	218.5	1.6
la26	1125	1127.0	5.5	3.4	**1113**	1118.4	1251.6	4.2
la27	1186	1188.8	9.3	3.3	**1182**	1186.3	633.5	3.2
la28	1149	1149.0	3.4	0.0	**1147**	1147.7	926.8	0.9
la29	1118	1120.6	5.5	2.9	**1116**	1118.6	775.7	2.3
la30	1204	1213.2	9.2	6.5	**1201**	1204.8	776.0	2.4
la31	1539	1540.6	9.6	0.9	**1536**	1540.9	388.5	2.1
la36	1162	1163.2	8.1	1.8	**1160**	1165.1	272.4	2.6
la38	1144	1146.6	6.9	2.6	**1143**	1156.1	1012.9	4.7
la40	1150	1151.6	7.8	2.6	**1146**	1149.4	1183.1	3.2

TABLE IV
NEW BKS GENERATED BY HILS FOR THE SET $rdata$.

	TS [17]				HILS			
I	*MS*	*Av.*	*T(s)*	*SD*	*MS*	*Av.*	*T(s)*	*SD*
la03	478	478.2	1.4	0.5	**477**	477.8	374.5	0.4
la15	1090	1090.0	1.8	0.0	**1089**	1089.9	73.0	0.3
la22	760	763.6	5.1	2.1	**757**	760.5	1051.0	2.2
la23	842	845.2	6.5	3.0	**840**	843.8	640.1	2.9
la24	808	813.8	4.1	3.4	**806**	810.6	814.5	2.3
la25	791	794.4	3.4	2.2	**787**	792.9	614.2	2.7
la27	1091	1092.6	7.5	1.3	**1088**	1090.3	711.4	1.6
la28	1080	1081.6	7.5	1.1	**1079**	1080.6	517.8	1.0
la29	998	998.6	4.0	1.3	**997**	998.0	488.2	0.9
la33	1499	1500.0	11.5	1.0	**1498**	1498.9	322.8	0.3
la37	1077	1080.6	9.5	2.7	**1068**	1078.5	810.9	5.1
la38	962	968.0	9.3	3.7	**958**	969.5	671.5	8.1
la40	970	974.0	6.1	3.4	**966**	970.1	807.7	4.4

TABLE V
NEW BKS GENERATED BY HILS FOR THE SET $vdata$.

	TS [17]				HILS			
I	*MS*	*Av.*	*T(s)*	*SD*	*MS*	*Av.*	*T(s)*	*SD*
la21	806	807.6	4.7	1.5	**805**	805.5	846.3	0.7
la22	739	739.8	6.4	0.8	**736**	736.8	1057.6	0.6
la23	815	816.0	5.7	1.2	**813**	813.8	1082.1	0.9
la24	777	779.0	6.8	1.6	**776**	777.0	836.6	0.7
la25	756	756.4	6.3	0.6	**754**	755.0	1099.0	0.8

Considering the BRdata dataset, the Table VI shows a comparison of the results of the proposed HILS algorithm with some recent and relevant works found in the literature: HABC [25], QPSO [22], DHS [23], best results from [10], SA-PC

[19], HmcDGA [21], SSPR [20], eGA [18] and TS [17]. The results of the Table VI are presented as follows: column (I) lists the name of the instances and the next columns indicate the best makespan values found by each method. At the end of the Table VI the general mean of the best makespan values of each method is presented, as well as the number of times each method reached the best known solution (#BKS). The BKS values for each instance are bolded.

When analyzing the Table VI, we realize that no algorithm reaches the BKS for each instance. The SSPR [20] has a slightly better performance, obtaining the best average value of the makespan and reaching 8 BKS for 10 instances. On the other hand, Previero [10] presented the highest mean, 6.5% worse than SSPR, and reached only 3 BKS. The proposed HILS reached 6 of 10 BKS, with a deviation of only 0.17% of the mean result of the SSPR. However, when analyzing the general mean of each algorithm, it is verified that its performances are very close.

The results of the HILS algorithm for the DPdata set are compared with the literature and are presented in Table VII. It should be noted that not all references run tests on the DPdata dataset and therefore such references are not listed in the Table VII. It is important to note that no algorithm always reaches a BKS. The eGA algorithm [18] obtained the best average of the best values achieved in the set of DPdata instances. The proposed HILS algorithm obtained only one BKS, but presented a deviation of only 0.93% of the mean obtained by the eGA.

Table VIII presents the results of the proposed algorithms for the BCdata set. A comparison is made with the best results from Previero [10], SSPR [20], eGA [18] and TS [17]. As for the set DPdata, not all references conducted experiments with set BCdata. On average, the results of the algorithms in this comparison are very close. The SSPR obtained the best results, reaching 20 of the 21 BKS and the best average value. The HILS algorithm presented the same number of BKSs as eGA and TS, which on average presented slightly better values.

The Tables VI to VIII presented only the best makespan values found by the algorithms for each instance for the reason that the works present different methodologies of presentation of the results. In this way, as the intention is to present a broad comparison between well known algorithms found in the literature, we opted for this presentation format.

The detailed results of the HILS algorithm for the data sets BRdata, DPdata and BCdata are shown, respectively, in the Tables IX, X and XI. For each instance (I), the tables show the best makespan value (MS) found in 10 executions, followed by the average value of MS (Av.), the average computational time, in seconds (T(s)) and the standard deviation (SD) of the MS values obtained in the ten executions of the algorithm.

V. Conclusions

This study considers the flexible job shop scheduling problem (FJSP) and presents a hybrid iterated local search algorithm, which uses the simulated annealing metaheuristic as local search (HILS).

To evaluate the performance of the HILS, an extensive set of instances obtained from the literature was used and the results were compared with those of the best algorithms in the literature. The computational experiments showed that the proposed HILS algorithm is competitive when compared to the literature. In particular, HILS was able to improve the best known solution (BKS) for 28 of the 178 FJSP instances considered.

Acknowledgment

The first author acknowledges support of FAPES (Espírito Santo Research Foundation), Grant 68199325/2014.

References

[1] M. Laguna and R. Marti, "Grasp and path relinking for 2-layer straight line crossing minimization," *INFORMS Journal on Computing*, vol. 11, no. 1, pp. 44–52, 1999.

[2] I. Kacem, S. Hammadi, and P. Borne, "Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 32, no. 1, pp. 1–13, Feb 2002.

[3] J.-Q. Li, Q.-K. Pan, and M. F. Tasgetiren, "A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities," *Applied Mathematical Modelling*, vol. 38, no. 3, pp. 1111 – 1132, 2014.

[4] N. B. Ho and J. C. Tay, "Genace: an efficient cultural algorithm for solving the flexible job-shop problem," in *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, vol. 2, June 2004, pp. 1759–1766 Vol.2.

[5] I. A. Chaudhry and A. A. Khan, "A research survey: review of flexible job shop scheduling techniques," *International Transactions in Operational Research*, vol. 23, no. 3, pp. 551–591, 2016. [Online]. Available: http://dx.doi.org/10.1111/itor.12199

[6] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Mathematics of Operations Research*, vol. 1, no. 2, pp. 117–129, 1976. [Online]. Available: https://doi.org/10.1287/moor.1.2.117

[7] P. Brucker and R. Schlie, "Job-shop scheduling with multi-purpose machines," *Computing*, vol. 45, no. 4, pp. 369–375, Dec 1990.

[8] D. Cinar, Y. I. Topcu, and J. A. Oliveira, *A Taxonomy for the Flexible Job Shop Scheduling Problem*. Cham: Springer International Publishing, 2015, pp. 17–37.

[9] P. Fattahi, M. Saidi Mehrabad, and F. Jolai, "Mathematical modeling and heuristic approaches to flexible job shop scheduling problems," *Journal of Intelligent Manufacturing*, vol. 18, no. 3, pp. 331–342, Jun 2007.

[10] W. D. Previero, "Estratégias de resolução para o problema de job-shop flexível," Ph.D. dissertation, Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 9 2016, doutorado em Ciência da Computação.

[11] D. Applegate and W. Cook, "A computational study of the job-shop scheduling problem," *ORSA Journal on Computing*, vol. 3, no. 2, pp. 149–156, 1991. [Online]. Available: https://doi.org/10.1287/ijoc.3.2.149

[12] E. G. Birgin, P. Feofiloff, C. G. Fernandes, E. L. de Melo, M. T. I. Oshiro, and D. P. Ronconi, "A milp model for an extended version of the flexible job shop problem," *Optimization Letters*, vol. 8, no. 4, pp. 1417–1431, Apr 2014.

[13] GUROBI, *Gurobi Optimizer Reference Manual*, 2016. [Online]. Available: https://www.gurobi.com

[14] P. Brandimarte, "Routing and scheduling in a flexible job shop by tabu search," *Annals of Operations Research*, vol. 41, no. 3, pp. 157–183, Sep 1993.

[15] J. Hurink, B. Jurisch, and M. Thole, "Tabu search for the job-shop scheduling problem with multi-purpose machines," *Operations-Research-Spektrum*, vol. 15, no. 4, pp. 205–215, Dec 1994. [Online]. Available: https://doi.org/10.1007/BF01719451

[16] S. Dauzère-Pérès and J. Paulli, "An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search," *Annals of Operations Research*, vol. 70, no. 0, pp. 281–306, 1997.

TABLE VI
COMPARISON OF THE HILS RESULTS WITH THE LITERATURE FOR THE SET BRDATA

I	HILS	EOPDHS	HABC	QPSO	DHS	Previero	SA-PC	HmcDGA	SSPR	eGA	TS
Mk01	40	40	41	**37**	40	40	40	40	40	40	40
Mk02	**26**	26	**26**	**26**	28	**26**	28	**26**	**26**	**26**	**26**
Mk03	**204**	**204**	206	**204**	**204**	**204**	216	**204**	**204**	**204**	**204**
Mk04	**60**	**60**	62	**60**	**60**	**60**	**60**	**60**	**60**	**60**	**60**
Mk05	173	172	176	173	172	176	**168**	175	172	173	173
Mk06	58	60	60	64	67	65	59	60	**57**	58	58
Mk07	**139**	**139**	140	**139**	143	150	147	144	**139**	144	144
Mk08	**523**	**523**	526	**523**	**523**	524	524	**523**	**523**	**523**	**523**
Mk09	**307**	**307**	**307**	**307**	309	341	**307**	311	**307**	**307**	**307**
Mk10	197	207	208	205	212	250	197	217	**196**	198	198
Mean	172.7	173.8	175.2	173.8	175.8	183.6	174.6	176.0	172.4	173.3	173.3
#BKS	6	5	2	7	3	3	3	4	8	5	5

TABLE VII
COMPARISON OF THE HILS RESULTS WITH THE LITERATURE FOR THE SET DPDATA.

I	HILS	QPSO	Previero	SSPR	eGA	TS
1a	**2505**	**2505**	2762	**2505**	2516	2518
2a	2231	2230	2424	**2229**	2231	2231
3a	2229	2229	2355	**2228**	2232	2229
4a	2503	**2498**	2670	2503	2515	2503
5a	2215	**2207**	2376	2211	2208	2216
6a	2202	**2170**	2254	2183	2174	2203
7a	2286	2264	2595	2274	**2217**	2283
8a	2067	2073	2265	**2064**	2073	2069
9a	2064	2066	2378	**2062**	2066	2066
10a	2277	2205	2585	2269	**2189**	2291
11a	2064	**2050**	2229	2051	2063	2063
12a	2029	2019	2492	**2018**	2019	2034
13a	2259	2253	2708	2248	**2194**	2260
14a	2169	2167	2448	**2163**	2167	2167
15a	2163	2165	3287	**2162**	2165	2167
16a	2255	2252	2595	2244	**2211**	2255
17a	2151	2134	2696	2130	**2109**	2141
18a	2134	2123	3164	2119	**2089**	2137
Mean	2211.3	2200.6	2571.3	2203.5	2191.0	2212.9
#BKS	1	5	0	8	6	0

TABLE VIII
COMPARISON OF THE HILS RESULTS WITH THE LITERATURE FOR THE SET BCDATA.

I	HILS	Previero	SSPR	eGA	TS
mt10c1	**927**	**927**	**927**	928	928
mt10cc	910	**908**	**908**	910	910
mt10x	922	**918**	**918**	**918**	**918**
mt10xx	**918**	**918**	**918**	**918**	**918**
mt10xxx	**918**	**918**	**918**	**918**	**918**
mt10xy	906	**905**	**905**	906	906
mt10xyz	858	**847**	**847**	**847**	**847**
setb4c9	**914**	**914**	**914**	919	919
setb4cc	909	**907**	**907**	909	909
setb4x	**925**	**925**	**925**	**925**	**925**
setb4xx	**925**	**925**	**925**	**925**	**925**
setb4xxx	**925**	**925**	**925**	**925**	**925**
setb4xy	**910**	**910**	**910**	916	916
setb4xyz	905	**902**	905	905	905
seti5c12	1174	1174	**1170**	1174	1174
seti5cc	1136	1136	**1135**	1136	1136
seti5x	1202	1205	**1198**	1201	1201
seti5xx	1198	1200	**1197**	1199	1199
seti5xxx	1203	1200	**1194**	1197	1197
seti5xy	1136	1136	**1135**	1136	1136
seti5xyz	1128	1129	**1125**	**1125**	**1125**
Mean	997.6	996.6	995.5	997.0	997.0
#BKS	8	14	20	8	8

TABLE IX
DETAILED HILS RESULTS FOR THE SET BRDATA.

	HILS			
I	*MS*	*Av.*	*T(s)*	*SD*
Mk01	40	40.0	2.2	0.0
Mk02	26	26.8	30.2	0.4
Mk03	204	204.0	2.6	0.0
Mk04	60	60.1	13.6	0.3
Mk05	173	173.0	115.4	0.0
Mk06	58	59.4	582.4	0.7
Mk07	139	140.3	1119.9	0.5
Mk08	523	523.0	5.1	0.0
Mk09	307	307.0	9.9	0.0
Mk10	197	198.6	1346.3	1.2
Mean	172.7	173.2	322.8	0.3

TABLE X
DETAILED HILS RESULTS FOR THE SET DPDATA.

	HILS			
I	*MS*	*Av.*	*T(s)*	*SD*
1a	2505	2517.6	906.5	5.1
2a	2231	2234.2	570.2	1.7
3a	2229	2230.0	745.5	0.5
4a	2503	2508.1	745.2	3.8
5a	2215	2221.4	731.3	2.8
6a	2202	2206.4	896.3	2.4
7a	2286	2295.2	1136.2	7.1
8a	2067	2069.5	1250.1	1.8
9a	2064	2066.3	602.3	1.2
10a	2277	2286.3	1480.3	6.9
11a	2064	2065.8	1099.5	1.3
12a	2029	2037.0	1262.0	3.5
13a	2259	2262.6	1111.9	4.0
14a	2169	2171.2	169.7	1.4
15a	2163	2165.6	874.1	1.3
16a	2255	2261.5	1443.6	4.0
17a	2151	2155.8	67.2	3.6
18a	2134	2141.7	828.3	3.6
Mean	2211.3	2216.5	884.5	3.1

[17] M. Mastrolilli and L. M. Gambardella, "Effective neighbourhood functions for the flexible job shop problem," *Journal of Scheduling*, vol. 3, no. 1, pp. 3–20, 2000.

[18] G. Zhang, L. Gao, and Y. Shi, "An effective genetic algorithm for the flexible job-shop scheduling problem," *Expert Systems with Applications*, vol. 38, no. 4, pp. 3563 – 3573, 2011.

[19] M. A. Cruz-Chávez, M. G. Martínez-Rangel, and M. H. Cruz-Rosales, "Accelerated simulated annealing algorithm applied to the flexible job shop scheduling problem," *International Transactions in Operational Research*, pp. 1–19, 2015.

TABLE XI
DETAILED HILS RESULTS FOR THE SET BCDATA.

	HILS			
I	*MS*	*Av.*	*T(s)*	*SD*
mt10c1	927	927.0	24.6	0.0
mt10cc	910	910.0	138.5	0.0
mt10x	922	923.8	702.5	2.1
mt10xx	918	920.1	364.0	3.8
mt10xxx	918	921.5	691.0	4.7
mt10xy	906	907.7	925.7	0.7
mt10xyz	858	858.4	280.6	0.8
setb4c9	914	918.0	395.2	3.7
setb4cc	909	913.6	886.4	3.4
setb4x	925	930.0	404.2	4.3
setb4xx	925	929.0	555.6	3.6
setb4xxx	925	930.6	446.1	4.9
setb4xy	910	918.8	712.6	4.7
setb4xyz	905	908.1	476.5	3.4
seti5c12	1174	1174.6	751.8	1.0
seti5cc	1136	1137.5	934.0	1.3
seti5x	1202	1207.4	870.8	2.7
seti5xx	1198	1208.0	1008.6	4.3
seti5xxx	1203	1207.3	1140.2	2.6
seti5xy	1136	1141.2	848.5	4.4
seti5xyz	1128	1130.3	680.7	2.7
Mean	997.6	1001.1	630.4	2.8

[20] M. A. González, C. R. Vela, and R. Varela, "Scatter search with path relinking for the flexible job shop scheduling problem," *European Journal of Operational Research*, vol. 245, no. 1, pp. 35 – 45, 2015.

[21] S. Ishikawa, R. Kubota, and K. Horio, "Effective hierarchical optimization by a hierarchical multi-space competitive genetic algorithm for the flexible job-shop scheduling problem," *Expert Systems with Applications*, vol. 42, no. 24, pp. 9434 – 9440, 2015.

[22] M. R. Singh and S. Mahapatra, "A quantum behaved particle swarm optimization for flexible job shop scheduling," *Computers & Industrial Engineering*, vol. 93, no. Supplement C, pp. 36 – 44, 2016.

[23] K. Z. Gao, P. N. Suganthan, Q. K. Pan, T. J. Chua, T. X. Cai, and C. S. Chong, "Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives," *Journal of Intelligent Manufacturing*, vol. 27, no. 2, pp. 363–374, Apr 2016.

[24] M. Gaham, B. Bouzouia, and N. Achour, "An effective operations permutation-based discrete harmony search approach for the flexible job shop scheduling problem with makespan criterion," *Applied Intelligence*, Aug 2017.

[25] X. Li, Z. Peng, B. Du, J. Guo, W. Xu, and K. Zhuang, "Hybrid artificial bee colony algorithm with a rescheduling strategy for solving flexible job shop scheduling problems," *Computers & Industrial Engineering*, vol. 113, no. Supplement C, pp. 10 – 26, 2017.

[26] H. R. Lourenço, O. C. Martin, and T. Stützle, "Iterated local search," in *Handbook of Metaheuristics, volume 57 of International Series in Operations Research and Management Science*. Kluwer Academic Publisher, 2002, pp. 321–353.

[27] P. Hansen and N. Mladenović, "First vs. best improvement: An empirical study," *Discrete Applied Mathematics*, vol. 154, no. 5, pp. 802 – 817, 2006, iV ALIO/EURO Workshop on Applied Combinatorial Optimization.

[28] P. Brucker, "The job-shop problem: Old and new challenges," in *in Proceedings of the MISTA Conference 2007*, 2007, pp. 15–22.

[29] P. Brucker and S. Knust, *Complex Scheduling*, 2nd ed., ser. GOR-Publications. Springer-Verlag Berlin Heidelberg, 2012.

[30] J. Barnes and J. Chambers, "Flexible job shop scheduling by tabu search." 1996, report Series: ORP96-09. Graduate program in operations research and industrial engineering. The University of Texas at Austin.

[31] G. M. Ribeiro, G. R. Mauri, and L. A. N. Lorena, "A simple and robust simulated annealing algorithm for scheduling workover rigs on onshore oil fields," *Computers & Industrial Engineering*, vol. 60, no. 4, pp. 519 – 526, 2011.