A clustering search metaheuristic for the bi-objective flexible job shop scheduling problem

Wagner A. S. Altoé, Dayan de C. Bissoli, Geraldo R. Mauri, André R. S. Amaral
Graduate School of Computer Science (PPGI)
Federal University of Espírito Santo (UFES)
Vitória, Brazil
altoe.wagner@gmail.com, dayan.bissoli@ufes.br, geraldo.mauri@ufes.br, amaral@inf.ufes.br

***Abstract*—The Flexible Job Shop Scheduling Problem (FJSP) is an extension of Job Shop Scheduling (JSP), which is closer to reality, allowing an operation of a given job to be processed by alternative machines. Considering that for some industries it is relevant to consider for more than one objective, the FJSP is treated in this study in a multiobjective way whit the following criteria: the last processing time of the last operation, called makespan, and the total tardiness. Therefore, it is proposed an algorithm based on the Clustering Search (CS) metaheuristic to generate solutions, and thus produce a set of non-dominated solutions in order to obtain the Pareto frontier, providing to decision maker a set of quality solutions. To evaluate the CS, we proposed a set of instances considering due dates for the jobs, to enable analysis of the bi-objective FJSP (BOFJSP). The CS results were competitive when compared to the literature, generating several non-dominated solutions.**

***Index Terms*—flexible job shop scheduling, multiobjective optimization, metaheuristic, clustering search, Pareto frontier.**

I. Introduction

In the current competitive environment, for an industry to keep up with market growth, it becomes necessary to efficiently schedule production. Industries must meet deadlines and minimize costs. So, organizations must plan their activities in such a way that resources are used efficiently [1].

The scheduling of activities in an industry consists of determining the sequence of tasks (jobs) in each machine in order to optimize some objective. In this category of problems we have a set of tasks, each one formed by a sequence of operations, and each operation is performed by a single specific machine. For a job to be processed, it must move through all machines in the pre-determined sequence. The purpose of this problem, called Job Shop Scheduling (JSP), comes down to designate the order of the operations for each job in the previously defined machines to optimize some criterion [2].

Considering the limitation of the JSP that each operation is processed by a specific machine, Brucker and Schlie [3] proposed the Flexible Job Shop Scheduling (FJSP). In this perspective, it is possible that some jobs can be processed by alternative machines, that is, the same job may have variations in the sequencing of their production [4].

The FJSP can have several different criteria as an objective function, the processing time of the last operation, called makespan, being the most studied. However other objectives can be evaluated, such as those related to the date of delivery, the sum of the delay in the delivery of products (total tardiness), among others [5], [6].

With the continuous need for quality improvement in the decision-making process, and the goal of making the model closer to the real operational context of the industries, comes the need to address the problem considering more than one objective. Such problems are called multiobjective problems [1], [7].

In multiobjective optimization problems, it is expected to satisfy all objectives. However, find this solution is not trivial, because the objectives tend to conflict with each other, that is, by improving one objective, it can lead to the worsening of the other (trade-off). Multiobjective problems approaches can be classified as follows [8]:

- Single objective: one objective is selected and the others are applied in the constraints of the problem;
- Linearization: the objective function is linearized, and for each objective is given a multiplier value, which is classified according to the importance of the objective. This approach uses strategic level knowledge to weight the objective function as needed;
- Pareto frontier: considers that the objectives of a multiobjective problem are usually conflicting, that is, by improving one objective, it can affect and / or worsen the other. There is no single solution that is optimal for all goals. In this situation, we seek for a set of non-dominated solutions that satisfy the problem constraints.

To better understand the Pareto frontier approach, Jozefowiez *et al.* [8] define the dominance concept as follows. A solution $y = (y_1, y_2, ...y_n)$ dominates a solution $z = (z_1, z_2, ...z_n)$ if and only if $\forall i \in \{1...n\}$, $y_i \leq z_i$ and $\exists i \in \{1...n\}$, $y_i < z_i$.

These classes of problems can be solved through exact methods and / or from mathematical programming models. However, such methods tend to be ineffective with instances that have a larger dimension, that is, they can not obtain an optimal solution, or even a feasible solution in reasonable computational time, which is common if we consider real FJSP applications [9]. Thus, it arises the motivation to use heuristic methods that are able to generate good quality solutions to these problems.

In order to extend the approaches made by Moreira *et al.* [10] and Lei [9], which addressed the JSP in a multiobjective way, minimizing simultaneously the objectives makespan and the total tardiness of jobs, in this paper a bi-objective version of the FJSP (BOFJSP) is treated, considering these same criteria.

Whereas BOFJSP is a problem that inherits the characteristics of the JSP, it is defined as NP-Hard [11]. Thus, as a solution method, we implemented the Clustering Search (CS) metaheuristic [12], considering that it has been showing good results for other combinatorial optimization problems, and that no application of CS to the FJSP has been found in literature so far [6], [13].

This paper is divided as follows: Section II presents some works related to the BOFJSP. The Section III details the methodology developed in this study. Computational experiments are described in Section IV. Section V presents the conclusions and then the references used in this work are listed.

II. Related Literature

For the classical FJSP, more recently, Cruz-Chávez *et al.* [14] presented an algorithm based on the Simmulated Annealing (SA) metaheuristic with a mechanism of partial scheduling and a cooling mechanism, which is function of the standard deviation. It was found that SA with the proposed mechanism, called SA-Partial Controlled (SAPC), converges faster to good solutions than without the proposed mechanism. Another metaheuristic that combines the methods Scatter Search and Path-Relinking with Tabu Search (SSPR) was developed by González *et al.* [15] for the FJSP. The authors sought to combine the diversification characteristic provided by Scatter Search and Path-Relinking, with the intensifying nature of TS. Both studies presented efficient results, providing new best known values for classic instances.

There are a large number of works in the literature that propose methods to solve different versions of the FJSP [6], [16]. However, with the aim to approximate the FJSP to the real cases of the industries, some authors conducted approaches focused on adding new objectives to the classic problem, that is, making it a multiobjective problem in order to provide an improvement in the quality of the decision-making process [9], [10], [17]–[19].

Shao *et al.* [20] have developed a hybrid algorithm combining the Particle Swarm Optimization (PSO) algorithm with the Simulated Annealing (SA) metaheuristic to solve the multi-objective FJSP. SA is used for local search in the proposed algorithm. The best solutions obtained by the SA are stored, and then submitted to a mechanism to search for non-dominated solutions.

Yuan and Xu [21] conducted an extensive literature review about multiobjective FJSP, which addresses the following objectives: makespan and related criteria to workload. As a method of solution for multi-objective FJSP, they proposed a Memetic Algorithm that is developed incorporating a new local search algorithm to the NSGA-II [22]. The NSGA-II is one of the well-known algorithms that aim to solve multiobjective problems [23].

Moreira *et al.* [10] proposed three versions of Genetic Algorithms to minimize the JSP considering the objectives: makespan and the total tardiness of the tasks. The results were compared with the literature. It was found that the proposed Genetic Algorithm with Path-Relinking obtained a greater number of non-dominated solutions in 80% of the tested problems.

To solve the JSP considering the objectives makespan and the total tardiness of the jobs, a method called Pareto Archive Particle Swarm Optmization (PAPSO) was developed by Lei [9]. The results were superior when compared with the SPEA2 method from [24].

Considering the multiobjective version for the JSP, which addresses the objectives makespan and the total tardiness of the jobs, as treated in [10] and [9], we did not find some application to the FJSP. So, it becomes interesting to study this approach.

III. Metodology

A. Problem Modeling

In order to approximate the FJSP to real-world cases, it is considered the model proposed by Özgüven *et al.* [25], with an adaptation in the respective model performed by Melo and Ronconi [5], to propose a new adaptation in the mixed integer linear programming model for the FJSP considering as objective function to simultaneously minimize the makespan and total tardiness of jobs, thus defining the bi-objective FJSP (BOFJSP):

Indices and sets
J: set of jobs;
M: set of machines;
O: set of operations;
i: jobs $(i, i^{'} \in J)$;
j: operations $(j, j^{'} \in O)$;
k: machines $(k \in M)$;
O_i: ordered set of operations of job i $(O_i \subseteq O)$;
O_{ij}: operation j of the job i $(O_{ij} \in O_i)$;
M_j: the set of alternative machines on which operation j can be processed, $(M_j \subseteq M)$;
$M_j \cap M_{j'}$: the set of machines on which operations j and $j^{'}$ can be processed.
P_{ij}: set of operations that must precede the operation O_{ij} in job.

Parameters
t_{ijk}: the processing time of operation O_{ij} on machine k;
d_i: due date of the job i;
L: a large number.

Decision variables
X_{ijk}: 1, if machine k is selected for operation O_{ij}; 0, otherwise;

S_{ijk}: the starting time of operation O_{ij} on machine k;
C_{ijk}: the completion time of operation O_{ij} on machine k;
$Y_{iji'j'k}$: 1, if operation O_{ij} immediatly precedes operation $O_{i'j'}$ on machine k; 0, otherwise
C_i: the completion time of job i;
T_i: tardiness of job i;
C_{max}: maximum completion time over all jobs (makespan).

$$\min \quad \sum_{i\in J} T_i \tag{1}$$
$$\min \quad C_{max} \tag{2}$$
$$s.t. \quad \sum_{k\in M_j} X_{ijk} = 1, \forall i \in J, \forall j \in O_i \tag{3}$$
$$S_{ijk} \leq X_{ijk} \times L, \forall i \in J, \forall j \in O_i, \forall k \in M_j \tag{4}$$
$$C_{ijk} \leq X_{ijk} \times L, \forall i \in J, \forall j \in O_i, \forall k \in M_j \tag{5}$$
$$C_{ijk} \geq S_{ijk} + t_{ijk} - (1 - X_{ijk}) \times L, \forall i \in J, \forall j \in O_i, \forall k \in M_j \tag{6}$$
$$S_{ijk} \geq C_{i'j'k} - (Y_{iji'j'k}) \times L, \forall i < i', \forall j \in O_i, \forall j' \in O_{i'}, \forall k \in M_j \cap M_{j'} \tag{7}$$
$$S_{i'j'k} \geq C_{ijk} - (1 - Y_{iji'j'k}) \times L, \forall i < i', \forall j \in O_i, \forall j' \in O_{i'}, \forall k \in M_j \cap M_{j'} \tag{8}$$
$$\sum_{k\in M_j} S_{ijk} \geq \sum_{k'\in M_{j'}} C_{ij'k'}, \forall i \in J, \forall j \in O_i, \forall O_{ij'} \in P_{ij} \tag{9}$$
$$C_i \geq \sum_{k\in M_j} C_{ijk}, \forall i \in J, \forall j \in O_i \tag{10}$$
$$T_i \geq C_i - d_i, \forall i \in J \tag{11}$$
$$X_{ijk} \in \{0,1\}, \forall i \in J, \forall j \in O_i, \forall k \in M_j \tag{12}$$
$$S_{ijk} \geq 0, \forall i \in J, \forall j \in O_i, \forall k \in M_j \tag{13}$$
$$C_{ijk} \geq 0, \forall i \in J, \forall j \in O_i, \forall k \in M_j \tag{14}$$
$$Y_{iji'j'k} \in \{0,1\}, \forall i < i', \forall j \in O_i, \forall j' \in O_{i'}, \forall k \in M_j \cap M_{j'} \tag{15}$$
$$C_i \geq 0, \forall i \in J \tag{16}$$
$$T_i \geq 0, \forall i \in J \tag{17}$$

The objective function (1) aims to minimize the total tardiness and objective function (2) minimizes the makespan. The constraint (3) ensures that each operation will have only one machine to process it, while the constraints (4) and (5) certify that if an operation is not allocated to a machine, the start and end times of processing for that machine are equal to zero. The constraint (6) ensures that the processing end time of an operation assigned to a machine will be greater than or equal to the sum of its start time plus the processing time of this machine. Constraints (7) and (8) limit a machine from processing more than one operation simultaneously. Constraint (9) guarantees the precedence of operations, and the constraint (10) certifies that the completion time of the job is greater than or equal to the time of completion of the operations. The constraint (11) certifies that the job will not be completed before having completed all operations, and the constraints (12) to (17) define the domain of the variables. Finally, the constraints (12) to (15) treat the integrity of the variables.

Based on the mathematical model presented, the complexity of the problem is highlighted because it is a problem with more than one objective and, considering the conflicting characteristics of the objectives to with each other, a single solution tends to prioritize one objective over the other. Thus, obtaining the Pareto frontier becomes an interesting alternative to solve the problem, since it is possible for the decision maker to evaluate a set of quality solutions and thus to define which objective to prioritize, according to your convenience.

B. Proposed instances

Considering the characteristic of the proposed model, which considers the criterion of minimize the total tardiness besides the makespan, we not find in the literature a set of instances that allow perform computational experiments. Consequently, there is a need to propose new instances, considering due dates, to evaluate the method proposed in this work. Thus, the proposed instances are described in this section.

We adapted the instances proposed by Brandimarte [26], because he did not define due dates. The instances were adapted in a similar way to the dates generated by Armentano and Scrich [27] and Moreira *et al.* [10], as shown in Equations (18), (19) and (20).

$$d_i = \left\lfloor (\beta \times \sum_{i=1}^{M} \bar{t}_{ij}) \right\rfloor \tag{18}$$
$$\beta_1 = ((J \times M)/1000) + 0,5 \tag{19}$$
$$\beta_2 = \beta_1 + 1 \tag{20}$$

The parameters β_1 and β_2 are used to determine the due date of jobs, being β_2 a more relaxed way to generate the due dates. The due date of the job i is referenced in d_i, being $\bar{t}_{ij}$ the average processing time of operations of the job i. To avoid rounding or difference problems when considering decimal places, we have set truncate the due date values.

C. Clustering Search metaheuristic

The Clustering Search metaheuristic (CS) is a generic definition of Evolutionary Clustering Search (ECS) [28]. The main difference between the methods is that ECS uses an evolutionary algorithm to generate solutions, while CS can use any metaheuristic [12], [13].

In CS, a cluster is defined by a set of three elements $C = \{c, v, r\}$: the best solution of the cluster is its center c; the number of solutions associated with each cluster defines the volume v; and the parameter r indicates the number of attempts without improvement after applying a local search [29].

When the volume v reaches the parameter λ, it becomes promising and then local search is applied. The parameter r

is incremented when better solutions are not obtained. If the inefficiency index r reaches the value r_{max}, a perturbation occurs in the center of the cluster c, in order to diversify the search.

As a solution-generating metaheuristic for CS, we used Simulated Annealing (SA) [30]. The SA is based on an analogy with thermodynamic to optimize solutions to combinatorial optimization problems.

SA performs perturbations in the current solution in an iterative way, seeking better neighbor solutions. An important feature of SA is the fact that it allows worse solutions as a form of diversification of the search. In doing this, SA tends to escape from local optimum, seeking a global optimum in later iterations. The probability of accepting solutions that are worse than the current one is decreased by the temperature.

Algorithm 1 shows the pseudocode of the proposed CS, using SA as the solution generator. SA runs until the timeout $Time_{max}$ expires. At each temperature, SA_{max} iterations are done and then the current SA solution is sent to the CS (lines 4-21).

The method is initialized by creating the number γ of clusters, with their respective characteristics of maximum volume λ and index of inefficacy r_{max}. Then a initial solution is created and defined as the best solution. Next, the SA is started, where at each iteration a neighborhood is chosen randomly between N^1 or N^2 to generate a neighbor solution of the current solution (lines 9-10). A neighbor solution is accepted if it is better than the current solution (lines 11-16) or with a given probability (lines 17-19), otherwise.

The SA temperature is decreased (line 21) and if the maximum number of clusters is not reached, the current SA solution is assigned to the center of a new cluster (lines 22-24). After all γ clusters have been initialized, the new solutions generated by the SA are now included in existing clusters (line 25 ahead).

Next, the assimilation process begins, which is defined based on the similarity between the current SA solution and the center of all clusters. The current SA solution is inserted in the cluster whose center is most similar to it. The algorithm then verifies if the cluster has become promising and if the local search or a perturbation will be applied or if a new best solution has been found (lines 26-45).

In order to perform the bi-objective analysis, initially the CS generates solutions considering the objective functions in a separated way. Thus, first the CS is applied to the BOFJSP considering the makespan criterion, and then, the CS is applied considering the criterion total tardiness. In both cases, in the search process, when a new best solution is found, it is stored in a list of solutions ($listSol$). After the CS is finished, the process of removing the dominated solutions from the list of solutions ($listSol$) is started, resulting on the set of non-dominated solutions.

The parameters used in the proposed CS algorithm are:

- γ: Maximum number of clusters;
- λ: Maximum volume of clusters;
- r_{max}: Limit to the index of ineffectiveness of clusters;
- T_0: Initial temperature of the SA;
- α: SA cooling rate;
- T_f: Final temperature of the SA;
- SA_{max}: Maximum number of iterations per temperature of the SA; and
- $Time_{max}$: SA execution time limit.

1) 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 assignment (e.g. by moving an operation from one machine to another), or by changing a sequence (e.g. by shifting an operation or swapping two operations) [31], [32]. Thus, two ways of exploring the neighborhood were used: N^1 and N^2, 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 1 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 $O_{4,1}$.

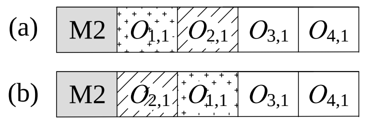

Fig. 1. Exchange move: N^1.

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 2 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.

Algorithm 1: CS Pseudocode for BOFJSP.

```
input  : $\gamma$, $\lambda$, $r_{max}$, $T_0$, $T_f$, $\alpha$, $SA_{max}$,$Time_{max}$, $s$
1  $cluster \leftarrow 0; r_i \leftarrow 0; v_i \leftarrow 0; \forall i = 1...\gamma; qtd \leftarrow 1;$
2  $listSol[qtd] \leftarrow s^* \leftarrow s; SA_{maxIni} \leftarrow SA_{max};$
3  while $Time < Time_{max}$ do
4      $T \leftarrow T_0;$
5      while $T > T_f$ do
6          $iter \leftarrow 0$ ;
7          while $iter < SA_{max}$ do
8              $iter \leftarrow iter + 1$ ;
9              $k \leftarrow random[1, 2]$ ;
10             $s^{'} \leftarrow N^k(s);$
11             if $f(s^{'}) < f(s)$ then
12                 $s \leftarrow s';$
13                 if $f(s^{'}) < f(s^*)$ then
14                     $s^* \leftarrow s^{'}; listSol[qtd] \leftarrow s^*; qtd + +;$
15                 end
16             end
17             else
18                 $s \leftarrow s^{'}$, with probability $e^{\frac{-(f(s^{'})-f(s))}{T}}$;
19             end
20         end
21         $T \leftarrow T \times \alpha;$
22         if $cluster < \gamma$ then
23             $cluster + +; v_{cluster} + +; c_{cluster} \leftarrow s;$
24         end
25         else
26             $i \leftarrow armin_{\{i \in \{1...\gamma\}\}}\{D_i\};$
27             $v_i + +; c_i \leftarrow best(c_i, s);$
28             if $v_i = \lambda$ then
29                 $v_i \leftarrow 1;$
30                 if $r_i = r_{max}$ then
31                     $r_i \leftarrow 0;$
32                     $c_i \leftarrow N^{(random[1,2])}(c_i);$
33                 end
34                 else
35                     LocalSearch($c_i$);
36                     if $c_i$ improved then
37                         $r_i \leftarrow 0;$
38                     end
39                     else
40                         $r_i + +;$
41                     end
42                 end
43                 $s^* \leftarrow best(s^*, c_i);$
44                 $listSol[qtd] \leftarrow s^*; qtd + +;$
45             end
46         end
47     end
48 end
49 return listSol
```

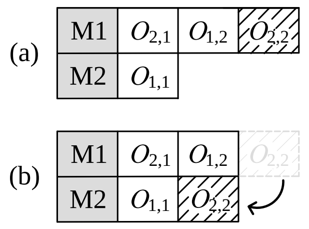

Fig. 2. Exchange move: N^2.

2) Similarity: The objective of this step is to identify which cluster has its center most similar to the current solution (line 27 of the Algorithm 1). This similarity is obtained by calculating the *Hamming Distance* (D_i) [33] to count the number of positions that the corresponding entries are different. At the end of the comparison, the solution is assigned to the cluster that is the least distant and, then, the cluster volume v_i is increased. Next, the solution of the cluster center is compared with the solution generated by the SA and if the solution generated by the SA is better, it becomes the center of the cluster.

Figure 3 presents two distinct solutions s_1 and s_2. This structure refers to the order of processing of jobs on the machines. The solutions differ in the machine that processes the last job ($O_{2,2}$). The Hamming distance between the two solutions is calculated and its value equals one.

s_1

Operation	$O_{1,1}$	$O_{1,2}$	$O_{2,1}$	$O_{2,2}$
Machine	1	2	1	2

s_2

Operation	$O_{1,1}$	$O_{1,2}$	$O_{2,1}$	$O_{2,2}$
Machine	1	2	1	1

Fig. 3. Representation of the Hamming distance calculation between two solutions s_1 and s_2: $D_i(s_1, s_2) = 1$.

3) Local Search: Soon after the CS identifies the promising areas, a local search (LS) is carried out, in order to refine the solution. This LS helps in quickly conversion to a local optimum. As LS was implemented an algorithm based on the best improvement method [34]. The pseudocode of the LS is presented in the Algorithm 2. The LS is divided into two stages. First takes place an intensification in the neighborhood of the current solution s by changing the sequence of the operations (lines 3-13). Then the flexibility is explored, changing the assignment of the operations in the current solution s (lines 14-34). At the end of the search, it returns the best solution found.

4) Selection of non-dominated solutions: After completion of the CS algorithm, begins the process of selecting the non-dominated solutions. This process consists of searching in the solutions vector $listSol$, with the objective of compare, at each iteration, the solution s_i with the other solutions s_j, verifying if s_i dominates s_j. If the solution s_i is dominated by s_j, then

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 $s^*$
```

Algorithm 3: Pseudocode of the selection of the non-dominated solutions.

```
input  : $listSol, qtd$
1  $s^* \leftarrow s' \leftarrow s$;
2  $i \leftarrow 1$;
3  while $i < qtdm$ do
4      $j \leftarrow i + 1$;
5      while $j <= qtd$ do
6          if dominates($listSol[i]$, $listSol[j]$) then
7              $listSol[j] \leftarrow listSol[qtd]$;
8              $qtd \leftarrow qtd - 1; j \leftarrow j - 1$;
9          end
10         else
11             if dominates($listSol[j]$, $listSol[i]$) then
12                 $listSol[i] \leftarrow listSol[qtd]$;
13                 $qtd \leftarrow qtd - 1; j \leftarrow i$;
14             end
15         end
16         $j \leftarrow i + 1$;
17     end
18 end
```

TABLE I
COMBINATION OF VALUES USED IN PARAMETER CALIBRATION.

SA						
Experiment	*1*	*2*	*3*	*4*	*5*	*6*
SA_{max}	1000	3000	5000	8000	10000	120000
α	0.95	0.96	0.97	0.975	0.98	0.985
T_0	100	200	400	600	800	1000
T_c	0.001	0.0015	0.0018	0.0019	0.0011	0.0112
CS						
γ	5	10	15	20	25	30
λ	5	10	15	20	25	30
r_{max}	1	4	6	8	10	12

s_i is removed from the vector. The pseudocode for selecting the non-dominated solutions is described in the Algorithm 3.

IV. COMPUTATIONAL EXPERIMENTS

A. Experimental setup

The experiments were conducted on a computer with Intel Core i5 750 2.67 GHz processor, 8GB of RAM and operating system Windows 7. The algorithm was implemented in C++ language and compiled with Visual C++ compiler.

The values for the CS and SA parameters were empirically defined and adjusted according to the results obtained in the experiments. The combination of parameters and their respective values are set out in Table I.

Tests were performed considering instances of different complexity: mk01, mk05 and mk10. The stopping criterion was set as 180 seconds without the algorithm presenting new improvement in the solution. At the end of the tests, the combination of parameters that obtained the best results was experiment number 3 of Table I.

B. Computational results of the CS for FJSP

To evaluate the CS, it was applied in the BRdata [26] instance set, and also to the set of proposed instances. For the BRdata set the CS was executed 5 times for each instance. To analyze the efficiency of the algorithm, in Table II, a comparison is made with the best values obtained by the Simmulated Annealig Partial Controlled (SAPC) [14] and the Scatter Search with Path-Relinking (SSPR) [15], considering the classical criterion of minimize the makespan. Table II is described below: The first column lists the names of the instances, followed by their characteristics (number of jobs $\times$ number of machines); The other columns list the best values found by each algorithm, followed by the time, in seconds, necessary to find the solution. The best makespan values obtained are indicated in bold.

TABLE II
COMPARISON OF THE CS RESULTS WITH THE LITERATURE CONSIDERING THE CRITERION MINIMIZE THE MAKESPAN.

I	$\|J\| \times \|M\|$	SAPC	T(s)	SSPR	T(s)	CS	T(s)
Mk01	10×6	**40**	3200	**40**	11	**40**	1
Mk02	10×6	28	4600	**26**	15	**26**	28
Mk03	15×8	216	6786	**204**	24	**204**	2
Mk04	15×8	**60**	5467	**60**	19	**60**	7
Mk05	15×4	**168**	6781	172	57	173	38
Mk06	10×15	59	3783	**57**	40	58	33
Mk07	20×5	147	7526	**139**	84	140	110
Mk08	20×10	524	8792	**523**	83	**523**	4
Mk09	20×10	**307**	4563	**307**	52	**307**	7
Mk10	20×15	197	7865	**196**	94	200	189
Mean		174.6	5936.3	172.4	47.9	173.1	41.9

When analyzing the Table II, it can be seen that the metaheuristic CS proposed in this work found the best known solution for six of the ten instances considered. For instances in which CS did not found the best results, it obtained a very close value, which demonstrates the quality of the results. When analyzing the average time to obtain the solutions, it is noticed that the CS proved to be very efficient. However it is important to emphasize that this is a comparison in which the algorithms were performed in computers with different configurations, so a direct comparison becomes limited.

C. Computational results of the CS for BOFJSP

The CS results for the BOFJSP are subdivided into two sets of instances. The results of the first instance set, which considers the parameter β_1, are listed in Table III. The results generated from the set of instances that considers the parameter β_2 are shown in Table IV. Both Tables are presented as follows. The first column indicates the name of the instance; the second column shows the number of non-dominated solutions (#NDS) found; and, the MK and TT columns refer to the respective makespan values and total tardiness of the non-dominated solutions found.

From the Tables III and IV it is verified that the CS obtained a total of 94 non-dominated solutions for the 20 considered instances. It can be seen that the increase in parameter value β_2 in relation to the parameter β_1 interfered with the results. The instances generated from the parameter β_1 obtained better results, that is, 6 non-dominated solutions more (12% better), and lower mean values of the objective functions, being 3 % better in the makespan criterion and 19% better in the total tardiness criterion.

The Pareto frontier graph was generated considering the results of the Instance Mk10-β_1, which is shown in Figure 4. A set of non-dominated solutions was obtained from the set of solutions generated by CS. This perspective allows the decision-maker to more comprehensively assess the impact of each criterion in the solution.

V. CONCLUSIONS

The present study addressed the Flexible Job Shop Scheduling in a Bi-Objective way (BOFJSP). The model used is based on an adaptation of the models proposed by Özgüven *et al.* [25] and Melo and Ronconi [5], considering two criteria in the objective function, to minimize the makespan and the total tardiness of jobs, simultaneously.

TABLE III
NON-DOMINATED SOLUTIONS FOR THE PROPOSED SET OF INSTANCES BASED ON β_1.

I	#NDS	MK	TT
Mk01-β_1	4	40	319
		41	245
		42	224
		43	218
Mk02-β_1	3	26	167
		31	155
		32	133
Mk03-β_1	3	204	1592
		221	1589
		223	1584
Mk04-β_1	6	60	683
		69	602
		70	573
		72	534
		73	532
		81	518
Mk05-β_1	5	173	2233
		184	1894
		187	1746
		189	1569
		191	1545
Mk06-β_1	3	59	492
		67	471
		68	453
Mk07-β_1	6	140	2251
		141	2199
		142	2183
		158	1534
		162	1399
		165	1388
Mk08-β_1	3	523	6022
		526	5976
		536	5876
Mk09-β_1	9	307	5486
		308	5480
		320	4265
		329	4113
		330	4112
		333	4051
		344	4017
		345	4003
		346	3917
Mk10-β_1	8	200	3525
		201	3517
		202	3485
		227	3183
		228	3118
		238	3081
		239	3068
		242	3060
Mean	5.0	193.4	2287.6

An algorithm based on the metaheuristic Clustering Search (CS) was implemented to solve the BOFJSP. CS is a hybrid algorithm that combines a solutions-generating metaheuristic and a clustering process. The Simulated Annealing metaheuristic was used as solution generator method for the proposed CS.

The results obtained by the CS considering the makespan criterion were competitive when compared to the literature, finding the best solution for 6 of 10 classic problems consid-

TABLE IV
NON-DOMINATED SOLUTIONS FOR THE PROPOSED SET OF INSTANCES BASED ON β_2.

I	#NDS	MK	TT
Mk01-β_2	3	40	210
		42	147
		43	139
Mk02-β_2	3	26	28
		28	10
		29	9
Mk03-β_2	3	204	1134
		207	1130
		221	1117
Mk04-β_2	4	60	604
		61	579
		70	414
		71	378
Mk05-β_2	8	173	2190
		174	2151
		176	2135
		177	2106
		182	1540
		186	1523
		187	1393
		190	1387
Mk06-β_2	4	59	338
		63	311
		67	307
		68	283
Mk07-β_2	6	140	1699
		142	1667
		153	976
		154	884
		156	866
		169	845
Mk08-β_2	4	523	6066
		524	5917
		527	5894
		528	5468
Mk09-β_2	4	307	4881
		308	4874
		323	3402
		326	3377
Mk10-β_2	5	202	2866
		229	2532
		233	2498
		236	2468
		237	2348
Mean	4.4	186.8	1843.0

ered. For the bi-objective version of the FJSP, the CS algorithm found a total of 94 non-dominated solutions in a set of 20 proposed instances, which demonstrates its ability to provide good solutions to aid in the decision-making process in the industries.

ACKNOWLEDGMENT

This research was partly supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq (processes 454569/2014-9 and 301725/2016-0), Fundação de Amparo à Pesquisa e Inovação do Espírito Santo - FAPES (processes 68199325/2014 and 73290475/2016) and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES.

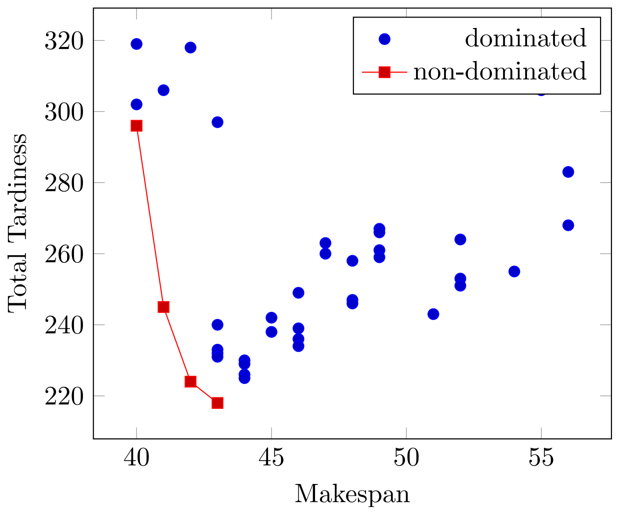

Fig. 4. Pareto frontier - instance Mk10-β_1.

REFERENCES

[1] 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.
[2] F. A. Rodammer and K. P. White, "A recent survey of production scheduling," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 18, no. 6, pp. 841–851, Nov 1988.
[3] P. Brucker and R. Schlie, "Job-shop scheduling with multi-purpose machines," *Computing*, vol. 45, no. 4, pp. 369–375, Dec 1990.
[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] E. L. d. Melo and D. A. P. Ronconi, "Regras de prioridade eficientes que exploram características do Job Shop Flexível para a minimização do atraso total," *Production*, vol. 25, pp. 79 – 91, 03 2015.
[6] 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.
[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr 2002.
[8] N. Jozefowiez, F. Semet, and E.-G. Talbi, "From single-objective to multi-objective vehicle routing problems: Motivations, case studies, and methods," in *The Vehicle Routing Problem: Latest Advances and New Challenges*, B. Golden, S. Raghavan, and E. Wasil, Eds. Boston, MA: Springer US, 2008, pp. 445–471.
[9] D. Lei, "A pareto archive particle swarm optimization for multi-objective job shop scheduling," *Computers & Industrial Engineering*, vol. 54, no. 4, pp. 960 – 971, 2008.
[10] M. C. O. Moreira, J. E. C. Arroyo, T. O. Januário, and P. L. Oliveira Júnior, "Um algoritmo evolutivo para o problema de job shop scheduling bi-objetivo problem," in *XL SBPO - Simpósio Brasileiro de Pesquisa Operacional*, João Pessoa/PB, 2008, pp. 1446–1457.
[11] 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.
[12] A. C. M. Oliveira and L. A. N. Lorena, "Hybrid evolutionary algorithms and clustering search," in *Hybrid Evolutionary Algorithms*, A. Abraham, C. Grosan, and H. Ishibuchi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 77–99.
[13] A. C. A. M. Oliveira, A. A. Chaves, and L. A. N. Lorena, "Clustering search," *Pesquisa Operacional*, vol. 33, pp. 105 – 121, 04 2013.
[14] 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.

[15] 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.

[16] D. Cinar, Y. I. Topcu, and J. A. Oliveira, "A taxonomy for the flexible job shop scheduling problem," in *Optimization, Control, and Applications in the Information Age: In Honor of Panos M. Pardalos's 60th Birthday*, A. Migdalas and A. Karakitsiou, Eds. Cham: Springer International Publishing, 2015, pp. 17–37.

[17] D. Lei, "Multi-objective production scheduling: a survey," *The International Journal of Advanced Manufacturing Technology*, vol. 43, no. 9, p. 926, Oct 2008.

[18] L. Wang, G. Zhou, Y. Xu, and M. Liu, "An enhanced pareto-based artificial bee colony algorithm for the multi-objective flexible job-shop scheduling," *The International Journal of Advanced Manufacturing Technology*, vol. 60, no. 9, pp. 1111–1123, Jun 2012.

[19] M. A. F. Pérez and F. M. P. Raupp, "A newton-based heuristic algorithm for multi-objective flexible job-shop scheduling problem," *Journal of Intelligent Manufacturing*, vol. 27, no. 2, pp. 409–416, Apr 2016.

[20] X. Shao, W. Liu, Q. Liu, and C. Zhang, "Hybrid discrete particle swarm optimization for multi-objective flexible job-shop scheduling problem," *The International Journal of Advanced Manufacturing Technology*, vol. 67, no. 9, pp. 2885–2901, Aug 2013.

[21] Y. Yuan and H. Xu, "Multiobjective flexible job shop scheduling using memetic algorithms," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 1, pp. 336–353, Jan 2015.

[22] K. Deb and D. Kalyanmoy, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[23] E. Ahmadi, M. Zandieh, M. Farrokh, and S. M. Emami, "A multi objective optimization approach for flexible job shop scheduling problem under random machine breakdown by evolutionary algorithms," *Computers & Operations Research*, vol. 73, pp. 56 – 66, 2016.

[24] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization," in *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, K. C. Giannakoglou, D. T. Tsahalis, J. Périaux, K. D. Papailiou, and T. Fogarty, Eds. Athens, Greece: International Center for Numerical Methods in Engineering, 2001, pp. 95–100.

[25] C. Özgüven, L. Özbakir, and Y. Yavuz, "Mathematical models for job-shop scheduling problems with routing and process plan flexibility," *Applied Mathematical Modelling*, vol. 34, no. 6, pp. 1539 – 1548, 2010.

[26] 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.

[27] V. A. Armentano and C. R. Scrich, "Tabu search for minimizing total tardiness in a job shop," *International Journal of Production Economics*, vol. 63, no. 2, pp. 131 – 140, 2000.

[28] A. C. M. Oliveira, "Algoritmos evolutivos híbridos com detecção deregiões promissoras em espaços de busca contínuo e discreto," Ph.D. dissertation, Instituto Nacional de Pesquisas Espaciais - INPE, São José dos Campos, 2004, doutorado em Computação Aplicada.

[29] G. M. Ribeiro, G. Laporte, and G. R. Mauri, "A comparison of three metaheuristics for the workover rig routing problem," *European Journal of Operational Research*, vol. 220, no. 1, pp. 28 – 36, 2012.

[30] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

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

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

[33] R. W. Hamming, "Error detecting and error correcting code," *Bell Labs Technical Journal*, vol. 29, no. 2, pp. 147 – 160, 1950.

[34] 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.