2013 IEEE Congress on Evolutionary Computation
June 20-23, Cancún, México

An algorithm based on Evolution Strategies for makespan minimization in hybrid flexible flowshop scheduling problems

Eduardo C. de Siqueira*, Marcone J. F. Souza†, Sérgio R. de Souza*, Moacir F. de França Filho*
and Carolina Gil Marcelino*
*Centro Federal de Educação Tecnológica de Minas Gerais
Belo Horizonte, Minas Gerais, Brazil
Email: eduardosiqueira@dppg.cefetmg.br, sergio@dppg.cefetmg.br, franca@des.cefetmg.br, carolimarc@gmail.com
†Departamento de Computação, Campus Universitário
Universidade Federal de Ouro Preto
Ouro Preto, Minas Gerais, Brazil
Email: marcone@iceb.ufop.br

***Abstract*—This work presents an algorithm based on Evolution Strategies for solving hybrid flexible flowshop scheduling problems. The construction of the initial solution is made by two methods: random NEH heuristic and Iterated Greedy Search metaheuristic. The main search mechanism in the search space of the developed algorithm is mutation, which is based on two parameters, one related to the probability of applying each type of mutation and the other related to the number of times each type of mutation is applied. After carried out the mutation, a portion of the parent population is refined by a local search procedure based on the Iterative Improvement Insertion method. These two versions of the proposed algorithm were tested and the results are compared with the ones from literature. The results showed better solutions for a subset of instances of the problem.**

I. Introduction

Scheduling problems arise in various economic sectors, especially in industrial production. The problem is to define a sequence of tasks to be performed on a set of machines, satisfying a set of operational constraints and taking into account a certain objective, which can be the minimization of the finish time of all tasks, the minimization of the time delay in completing tasks, among others.

There are several studies about scheduling problems. In spite of this, a strong gap between theory and practice has always remained in these works. Many developed models of scheduling problems do not incorporate various operational constraints, and this fact limits their applications in real situations. However, there is a recent trend in developing approaches for solving real problems [1].

This paper considers the real problem described in [1], which treats a variation of the Hybrid Flowshop problem (HFS). A set of tasks passes through a number of stages and, at each stage, there is a set of unrelated parallel machines. The main characteristic of a flowshop is the processing flow in the machines be the same, i.e, all tasks follow the same sequence of stages.

However, in the treated problem, the stages may not be all executed. This variant, named as Hybrid and Flexible Flowline problem (HFFL), is a generalization of HFS problem as well as Flexible Flowline problems. In this case, the optimization criterium is to minimize the maximum completion time, known as makespan.

HFFL problem belongs to NP-Hard class problem [1]. Metaheuristics have been used to solve such problems by providing quality solutions with low computational cost. Therefore, this paper proposes an algorithm based on Evolution Strategies (ES) [2], [3] for solving it. Evolution Strategies are iterative procedures that perform the search for the solution space through individuals populations that represent potential solutions to the optimization problem. These methods were developed for application to numerical optimization problems, and are characterized by their robustness and efficiency. Each iteration is called a generation, by analogy with biological systems. In each generation, recombination and mutation operators act on the population, allowing exploration of the search space. The principle of survival of the fittest is implemented by the selection mechanism, which ensures the survival of individuals representing best quality solutions to the problem. The use of this class of algorithms is motivated by its successful application in solving various optimization problems such as those reported in [4] and [5]. The adaptation proposed here for ES algorithm is inspired by [4].

The remain of this article is organized as follows: Section II shows a short literature review concerning flowshop problems. In Section III, the problem to be dealt is formulated and examples are included. The proposed algorithm for solving the treated problem is described in Section IV. Section V shows the found computational results. Section VI closes the article, pointing conclusions and future works.

978-1-4799-0454-9/13/$31.00 ©2013 IEEE

II. Literature review

This section presents a literature review concerning the addressed problem. Hybrid and Flexible Flowline (HFFL) problem is a generalization of Hybrid Flowshop (HFS) problem. HFS, in turn, is a different problem from Flexible Flowshop and from Flexible Flowline, since, for these last two problems, the machines available at each stage are identical. HFS, on the other hand, does not have this constraint. Some stages may have only one machine, but at least one stage must have a parallels machines group. These machines are usually different according [6].

Second [7], the first researches about HFS scheduling problems arose around 1970. One of the first works on HFS in modeling a production system is [8], which studies a synthetic fiber industry under this approach. [9] show that HFS with the objective of makespan minimization is a NP-Hard problem. Therefore, as HFFL is a generalization of HFS, then so is HFFL.

In [10], a survey on applications of evolutionary computation to real-world problems in industries of metals, paper and chemistry has been presented. In [11], the solution of a scheduling in a real-life flowshop with sequence-dependent setup times of jobs and different relative weights for each job has been considered. The objetive is to minimize the maximum weighted tardiness of a job and the total weighted tardiness of jobs. The proposed heuristic consists in a modified Simulated Annealing algorithm, which incorporates a perturbation scheme.

Different authors address the HFS problem. In [12], genetic algorithms has been proposed for solving an HFS problem with unrelated parallel machines at each stage, sequence-dependent setup times and machine eligibility. The results were found from a extensive computacional tests performed with a set of 1320 random instances and instances with real data. The proposed genetic algorithm has shown significantly efficient for solving this class of problem.

Six different mathematical models were presented in [13] for solving a generalization of the permutation flowshop scheduling problem. The considered objective was the makespan minimization. The models were tested with small-sized instances having 4 to 16 tasks and none of them was able to achieve the optimal solution in acceptable processing time for all instances.

In [14], an unrelated parallel machine problem with machine and job sequence dependent setup times is considered. In the studied problem, the setup time depends on the machine and job sequence, as well as on a number of resources assigned. The objectives are the minimization of a linear combination of total completion time and the total number of resources assigned. The results showed optimal solutions for instances until 8 tasks and 5 machines.

In [15] is presented a review of exact, heuristic and meta-heuristic algorithms for solving HFS. A Lagrangian relaxation method with cut generation is proposed in [16] for solving the hybrid flowshop scheduling problem and to minimize the total weighted tardiness. In [17], the HFS scheduling problem in a flexible multistage batch production system is tacked via an batch agregation and splitting heuristic, with the objective of makespan minimization and increase the productive capacity utilization. In [18], a clonal selection algorithm is proposed for solving an HFS problem. The objective, in this case, is the minimization of the sum of the total earliness and tardiness penalties.

In [19], an Hybrid Flowshop problem with two stages and recirculation is solved via constructive heuristics. The same class of problem is treated in [20], however, there is unrelated parallel machines in the second stage in this case. An adaptive randomized list scheduling heuristic is proposed for solving the addressed problem. The objective, in these two articles, is the makespan minimization.

A new mathematical model for HFS is presented in [21] with a processor assignment that minimizes makespan and cost of assigning a number of machines to stages. [22] studies HFS problem with unrelated machines, sequence-dependent setup time, availability constraints, and limited buffers applied in a production environment of a television assembly line for inserting electronic components. Genetic algorithms were used in these two articles for solving the treated problems, both with makespan minimization as the objective.

An heuristic algorithm for solving an Hybrid Flowshop problem with machines elegibility has been proposed in [23]. The objectives to be dealt are minimization of sum of the completion time of groups of jobs and the minimization of sum of the differences between completion time of jobs and delivery time of the group containing that job. The same problem is studied in [24], where a Particle Swarm Optimization (PSO) is proposed for solving it.

A mathematical model for a new hybrid and flexible scheduling problem, named Hybrid and Flexible Flowline (HFFL) problem is showed in [1]. With this model, HFFL problems up to 15 tasks, 3 stages and 2 machines in each stage has been solved at optimality. Constructive heuristics for solving more large instances of this problem (until 100 jobs) are proposed.

In [25], two important questions, related to HFFL, are highlight: how to determine the sequence at each stage and the distribution of tasks on the machines at each stage. An ILS-based algorithm is proposed for solving this problem, with the objective of makespan minimization. Other works treating HFFL are [26], [27] and [28]. These articles had solved this problem using genetic algorithms.

III. Problem formulation

Let a set of tasks $N = \{1, 2, 3, ..., n\}$, a set of stages $M = \{1, 2, 3, \cdots, m\}$ and a set M_i of unrelated parallel machine for processing these tasks. Each task can pass through each stage. The following characteristics are considered [1]:

1) F_j: set of stages visited by j task, $1 \leq F_j \leq m$;
2) p_{ilj}: processing time of task j on machine l and stage i;

3) rm_{il}: release time of machine l at stage i. It represents the start time of the processes in the machine. No task can be started on the machine before release time;
4) E_{ij}: set of eligible machines for task j on stage i;
5) lag_{ilj}: lateness time between the end of task j on machine l of stage i and the start of the next stage of task j;
6) S_{iljk}: setup time of machine l on stage i, when task k is processed after task j. There is an associated binary value, named A_{iljk}, which indicates if the setup time is anticipatory, i.e, A_{iljk} is equal to 1 (one) if the preparation can be done before the task be released in earlier stage, and is equal to 0 (zero), otherwise.

The objective to be satisfied is the makespan minimization, i.e., the minimization of finish time of the last task of the system.

To exemplify the problem, we consider an instance with five tasks and two stages. Each stage have three machines. Tables I to V show the data of the example and are discussed in the sequel.

Table I shows which machines are eligible for each task at each stage. In this table, j indicates a task and line i represents each of the two stages. For this table, it turns out that the first task can be executed on machines 2 and 3, at stage 1, and on machine 4, at stage 2. It is also verified that the tasks 1, 2 and 5 visit all stages, while tasks 3 and 4 skip stages 1 and 2, respectively.

Table II shows the release time for each machine at each stage. In this table, line rm_{il} indicates the release time for each machine l at stage i. For example, the release time of machine 1 at stage 1 is equal to 20 and the release time of machine 5 at stage 2 is equal to 29.

Table III shows the processing time of each task at each machine while Table IV shows the lag times. In these two tables, j indicates a job, each line i represents each stage and each line l represents each machine. It is turns out that the processing time of task 1 on machine 2 is equal to 8 and the lag time is null. On the other hand, the processing time of the same task, when processed on machine 3, is equal to 13 and the lag time is equal to 9. The processing time at a non eligible machine is null.

Table V shows the setup times and the associated binary-value. The setup times are sequence-dependent. If setup time is anticipatory, the associated binary-value is equal to 1; otherwise, this value is 0. In this table, j and k represent a task, the line i represents each one of the two stages and the data of each machine are separated by commas. For example, the setup time between tasks 1 and 4 on machine 3 is 10 and non-anticipatory.

Figure 1 shows a possible scheduling for this instance. Note the makespan is equal to 107.

Table I
ELIGIBILITY

	i	1	2
j	1	{2,3}	{4}
	2	{3}	{4,5,6}
	3	-	{5}
	4	{1,3}	-
	5	{1,2}	{4,5,6}

Table II
RELEASE TIME

i	1			2		
l	1	2	3	4	5	6
rm_{il}	20	5	33	38	29	45

IV. METHODOLOGY

In this section, we present an algorithm based on Evolutionary Strategies (ES) for solving the treated problem. A portion of the initial population is generated by means of an adaptation of NEH construction heuristic. The other portion is built based on the metaheuristic Iterated Greedy Search (IGS). The algorithm makes use of Iterative Improvement Insertion (III) [29] procedure for refining the part of individuals generated during the evolutionary process. At the end of the generations, a local search method with full representation in individuals, proposed in [30], is applied.

This section is organized as follows: Subsection IV-A shows how an individual is represented. Subsection IV-B shows the used mutation operators. Subsection details the basic Evolutionary Strategies algorithm, while Subsections IV-D and IV-E show the NEH and IGS methods. Subsection IV-F brings the explanation of the functioning of the local search method with full representation.

A. Solution Representation

An individual ind of the problem is represented by a double (π, M) and by two vectors of mutation parameters, named $prob$ and a. In this representation, π is the task sequence for processing and M is the machines matrix. All stages have the same π task sequence, named permutational representation, and, for each task, there is a column of the M matrix showing in which machine a task is processed in each stage. If this task is not performed at a given stage, a null value is assigned to it.

Table III
PROCESSING TIME

	i	1			2		
	l	1	2	3	4	5	6
j	1	0	8	13	21	0	0
	2	0	0	15	45	46	44
	3	0	0	0	0	13	0
	4	12	0	32	0	0	0
	5	18	16	0	16	19	36

Table IV
LAG TIME

	i	1		
	l	1	2	3
j	1	0	0	9
	2	0	0	8
	3	0	0	0
	4	2	0	-6
	5	0	-5	0

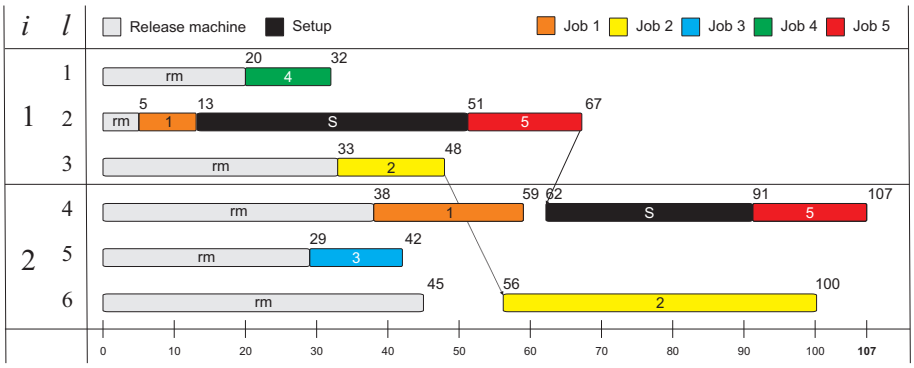

Figure 1. Gantt chart. $C_{max} = 107$

Table V
SETUP TIME AND A_{iljk} VALUES

	i	1				
	k	1	2	3	4	5
j	1	-,-,-	-,-,5(0)	-,-,-	-,-,10(0)	-,38(0),-
	2	-,-,10(0)	-,-,-	-,-,-	-,-,38(0)	-,-,-
	3	-,-,-	-,-,-	-,-,-	-,-,-	-,-,-
	4	-,-,46(0)	-,-,21(0)	-,-,-	-,-,-	43(0),-,-
	5	-,6(0),-	-,-,-	-,-,-	25(0),-,-	-,-,-
	i	2				
j	1	-,-,-	33(1),-,-	-,-,-	-,-,-	29(0),-,-
	2	30(1),-,-	-,-,-	-,6(1),-	-,-,-	8(1),39(1),27(1)
	3	-,-,-	-,45(1),-	-,-,-	-,-,-	-,30(0),-
	4	-,-,-	-,-,-	-,-,-	-,-,-	-,-,-
	5	41(0),-,-	47(0),7(0),38(1)	-,48(0),-	-,-,-	-,-,-

The $prob$ and a vectors contain a number of positions equal to the number of different types of mutation, and each position of $prob$ stores the probability of applying a type of mutation, whereas a stores the intensity of this mutation.

Figure 2 shows the double (π, M) for an individual of the example presented in Figure 1.

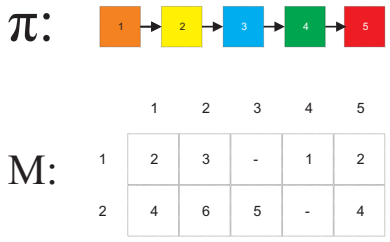

Figure 2. Representation of a portion of an individual

Figure 2 shows that the task sequence is 1, 2, 3, 4 and 5, i.e., the task 1 precedes task 2, which precedes tasks 3 and 4 and task 5 is the last of the sequence. By matrix M, task 1 is processed on machine 2 at stage 1 and on machine 4 at stage 2. The null value in line 1 and column 3 of matrix M shows task 3 is not performed in stage 1.

Each stage contains a sequence of tasks for local search with full representation. For this showed example, the task sequence for stage 1 is 1, 2, 4 and 5; for stage 2 is 1, 2, 3 and 5.

B. Mutation Description

The exploration of the solution space of the problem is based on two types of mutation:

- Reallocation in the Sequence (MRS): this type consists on to choose a task and change it to a new position in the sequence;
- Block Reallocation in the Sequence (MBRS): this type consists on to choose a block of adjacent tasks and change it to a new position in the sequence;

In this article, we consider blocks of 2, 3 and 4 tasks and, in consequence, MBRS type is divided in three subtypes of mutation, associated to the number of tasks considered in the block. Hence, there are a total of four types of mutation to be realized.

C. Evolutionary Strategies

In this article an algorithm, based on Evolutionary Strategies (ES), is proposed for solving the treated problem. The basic pseudocode of ES is presented in Algorithm 1.

This ES algorithm is divided in two phases. In the first phase (lines 1 to 7 of algorithm 1), an initial population of μ individuals are generated, being half of the individuals created by the Greedy-Random NEH method, and the other half built based on IGS metaheuristic.

The second phase of the Algorithm 1 (lines 8 to 20) is for creating a population of γ children through mutation. The Iterative Improvement Insertion method (III) is applied to κ children in the sequel, for $\kappa < \gamma$. This method is not applied to all individuals in the population of children in view of its high computational cost. After these procedures, surviving individuals are selected to the next generation among the population of parents and children. These steps are repeated until a stopping criterion is met. Each ES module is detailed below.

In lines 3 to 5, an individual ind of the population is created from the application of vectors $prob$ and a to the sequence (individual) s. The mutation probability vector $prob$ of each individual is initialized with a real number randomly chosen in the interval $[0, 1]$ for each of four types of mutation. The vector of mutation intensity a of this individual is initialized with an integer randomly chosen in the set $\{1, \cdots, 8\}$.

At line 12, the mutation parameters vectors are changed. A mutation, according to a normal distribution with standard deviation σ_{real} and zero average is applied to probability vector $prob$. On the other hand, to the intensity vector a is applied a mutation following a binomial distribution with standard deviation $\sigma_{binomial}$ and zero average.

The individual mutation (line 13) operates as follows. A random real number in the interval $[0, 1]$ is generated and the probability condition $prob_l$ for this number is verified. In affirmative case, a_l tasks (or task blocks) are removed from current solution and inserted in a vector π_R. After, to rebuilt the solution, each task (or task block) of the current solution is inserted in the best possible position of the scheduling.

In line 17, the refinement of κ parents by III method is performed. This refinement consists on to select a task from the current solution and insert it in the best possible position to scheduling. If the current solution is improved, it is updated, and the procedure restarted. These steps are repeated until all tasks are considered and no improvement is possible in the solution. The order of task selection is random.

As shown in line 19, the next generation of the algorithm is composed by μ individuals with the lowest values of makespan among the population of parents and children.

After stopping criterion is reached, the local search method with full representation of individuals in the surviving population is applied. This representation is considered here in order to better explore the search space, since the permutation representation, while achieving good results, restricts the search space. Finally, the best individual of the final population is returned.

Algorithm 1 Evolutionary Strategies (ES)

Require: μ, γ, κ, $StoppingCriterion$
Ensure: $BestIndividual$
1. **for** $w \leftarrow 1$ to μ **do**
2. $\pi_w \leftarrow$ generateSolution();
3. Initialize the vector of probability of mutation $prob_w$;
4. Initialize the vector of intensity of mutation a_w;
5. Find ind_w by applying $prob_w$ and a_w vectors to π_w;
6. $Pop_w \leftarrow ind_w$;
7. **end for**
8. **repeat**
9. **for** $w \leftarrow 1$ to γ **do**
10. $y \leftarrow$ random integer number between 1 and μ;
11. $ind_w \leftarrow Pop_y$;
12. $ind_w \leftarrow$ Parameter_mutation(ind_w, σ_{real}, $\sigma_{binomial}$);
13. $Child_w \leftarrow$ Individual_mutation(ind_w);
14. **end for**
15. **for** $w \leftarrow 1$ to κ **do**
16. $y \leftarrow$ random integer number between 1 and γ;
17. $Child_y \leftarrow$ III($Child_y$); {Local search}
18. **end for**
19. $Pop \leftarrow$ selection(Pop, Children, μ);
20. **until** *StoppingCriterion* be satisfied
21. **for** $w \leftarrow 1$ to μ **do**
22. $ind_w \leftarrow$ LocalSearchFullRepresentation(ind_w);
23. **end for**
24. **return** BestIndividualPop();

D. Greedy-Random NEH Procedure

The Greedy-Random NEH procedure is responsible for generating a portion of individuals for the initial population of the proposed algorithm. This procedure is an adaptation of the NEH procedure proposed in [31] and operates as follows. Initially all tasks are inserted in the List of Candidates (LC). In the first iteration two tasks $j \in$ LC are randomly selected and a subprocedure that seeks the best possible sequencing between these two tasks is run. The chosen task is performed on the machine that can finish it soon. In the second iteration these two tasks are removed from LC and a single task from LC is then random chosen. The Greedy-Random NEH procedure is applied again to verify the best position for insertion of the chosen task. The process continues with the removal of the chosen task of LC and the application of NEH-like procedure until all tasks have been sequenced.

E. Iterated Greedy Search

The Iterated Greedy Search (IGS), whose pseudocode is shown in Algorithm 2, is used for generating a portion of individuals to compose the initial population of the proposed algorithm. In the sequel the algorithm is discussed.

In line 1, an initial solution is generated by the NEH method [31]. This method operates as follows. Initially, the average processing of each stage for each task is calculated, i.e., if a task could pass thought two machines at stage i and these machines consume the processing times P_{ij1} and p_{ij2}, then to this task j is assigned the average time, given by $\bar{p}_{ij} = \frac{p_{ij1}+p_{ij2}}{2}$. Following, the sum of these average processing times in all stages is calculated for each task j, i.e, the average processing time of task j, given by $\bar{p}_j = \sum_i \bar{p}_{ij}$, is calculated. In the first iteration, the two tasks with larger $\bar{p}_j$ are selected and a procedure that seeks the best possible sequencing between these two tasks is performed. In this scheduling the chosen task is performed on the machine capable of complete it as soon as possible. This sequencing is used as a basis to insert the third task. In the second iteration, the third task with larger $\bar{p}_j$ is chosen and sequenced in the best position possible. The process continues until all tasks have been sequenced.

The Iterative Improvement Insertion (III) method is applied as a local search method in line 2.

The phases of destruction, reconstruction and acceptance of the solution are repeated until the stopping criterion is met (lines 3 to 16). The destruction phase (lines 6 to 8) consists on removing d (input parameter) tasks of the current solution and insert them in a vector π_R. In order to rebuild the solution (lines 9 to 11), each task of π_R is inserted into the best possible position to scheduling. Then the III local search method is applied again. After that, if the solution generated is better

Algorithm 2 Iterated Greedy Search (IGS)

Require: d, $Temperature$, $StoppingCriterion$
Ensure: π
1. $\pi \leftarrow$ constructionNEH();
2. $\pi \leftarrow$ III(π); {Local Search}
3. **repeat**
4. $\pi' \leftarrow \pi$;
5. $\pi_R \leftarrow \emptyset$;
6. **for** $w \leftarrow 1$ to d **do**
7. Remove randomly a task of π' and insert it in π_R;
8. **end for**
9. **for** $w \leftarrow 1$ to d **do**
10. Remove the task $\pi_R(w)$ and insert it into the best possible position of π';
11. **end for**
12. $\pi' \leftarrow$ III(π'); {Local Search}
13. **if** $C_{\max}(\pi') < C_{\max}(\pi)$ **then**
14. $\pi \leftarrow \pi'$;
15. **end if**
16. **until** *StoppingCriterion* be satisfied;
17. **return** π

than the current solution, then the current solution is updated.

F. Local search with full representation

If the sequencing is analyzed, we note that the makespan is determined by a critical path. For example, as shown in Figure 1, we observe the start of the last operation of the task 5 is determined by the end of the operation of the same task in the previous stage and the start of this operation is determined by the end of the first operation of task 1, creating a path operations. An operation is the execution of a task on a stage. When this path determines the makespan, it is named critical path.

Second [30], it is very unlikely to improve the makespan value moving an operation that is not on the critical path. Therefore, local search is only applied to operations on the critical path.

This local search method starts with the task having completion time equal to the value of the makespan in the current solution. In Figure 1, this operation is given by task 5 on machine 4. This operation is reallocated in all possible positions of sequencing on al eligible machines. If the solution is improved, the current solution is updated, and the procedure restarts. If there is no improvement, the next operation on the critical path is considered. This operation may be the operation of the preceding stage of the same task or a previous operation of another task on the same machine. When two or more tasks determine the makespan or the start time of another task on the critical path, the local search is interrupted. In this case, the makespan value can not be improved by relocating a single operation. The procedure ends when no more tasks on the critical path.

Table VI
FAMILY OF INSTANCES

Família de Instâncias	n	m	m_i
15_2_1	15	2	1
15_2_3	15	2	3
15_3_1	15	3	1
15_3_3	15	3	3
50_4_2	50	4	2
50_4_4	50	4	4
50_8_2	50	8	2
50_8_4	50	8	4

Table VII
ALGORITHM PARAMETERS

	μ	γ	κ	d	T
ES	30	60	10	-	-
IG	-	-	-	2-8	0.5

V. EXPERIMENTAL RESULTS

The proposed algorithms were implemented in C++ with IDE Borland C++ Builder 6. The computational tests were performed at a desktop computer with Intel Core 2 Quad, 2.67GHz, 4 GB RAM memory, in Windows 7 32 bits operational system.

Eight families of instances, from [1], were used to perform the computational tests, each one with 12 instances. The main characteristics of these instances are shown in Table VI, where n is the number of machines; m is the number of stages; and m_i is the number of machines at each stage.

Table VII shows the parameter values adopted in the algorithm. In this table, μ, $gamma$ and κ columns represent the parameter values for ES algorithm and d and T columns shows the parameter values for IGS algorithm. These values were defined empirically.

The proposed algorithm was compared with the literature best results presented in [27] and [30] in relation three aspects: (i) solutions generated in construction phase; (ii) solutions generated during the iterations (generations) of the ES algorithm; (iii) solutions generated after application of local search with full representation. To evaluate the first aspect, the gap of the best solutions is calculated as well as the gap of the average solutions generated by the construction algorithms for each group of instances in relation to optimal values (in the case of instances involving 15 tasks) or best literature values (in the case of instances of 50 tasks). To evaluate the second aspect, the gap of the best solutions is again calculated and the gap of average solutions in the generations of the based on ES algorithm. To evaluate the third aspect, the gap of generated solutions after the application of local search with full representation.

The algorithm run 10 times for each instance. The execution was interrupted if the algorithm remained during 40 iterations without improving the solution.

Table VIII shows the comparison results of the proposed algorithm with the three aspects mentioned for the families of instances with 15 tasks and Table IX shows the comparison results for the families of instances with 50 tasks. These

Table VIII
GAP OF PROPOSED ALGORITHM FOR FAMILIES OF INSTANCES WITH 15 TASKS

Instance	Best Literature	Construction			Generations			Full representation			Improve	
		Best	Average	Time(ms)	Best	Average	Time(ms)	Best	Average	Time(ms)	Best	Average
15_2_1	1734.75	0.27	0.44	522.85	0.20	0.21	5894.00	0.14	0.18	5910.23	0.09	0.33
15_2_3	618.75	2.20	17.44	1475.50	1.12	2.30	6998.93	0.68	1.87	7890.03	4.25	11.70
15_3_1	1830.92	0.36	0.60	762.38	0.30	0.31	5515.41	0.27	0.29	5555.73	0.11	0.31
15_3_3	693.75	10.99	14.46	2248.85	1.46	2.31	11343.88	1.41	2.31	11343.88	7.30	9.01
Average	-	3.45	8.23	-	0.77	1.28	-	0.62	1.16	-	2.94	5.34

Table IX
GAP OF PROPOSED ALGORITHM FOR FAMILIES OF INSTANCES WITH 50 TASKS

Instance	Best Literature	Construction			Generations			Full representation			Improve	
		Best	Average	Time(ms)	Best	Average	Time(ms)	Best	Average	Time(ms)	Best	Average
50_4_2	2787.92	5.51	6.71	17370.47	2.06	3.23	711717.63	1.86	3.16	719918.29	3.24	3.62
50_4_4	1453.50	0.59	1.88	24335.90	**-3,44**	**-2.20**	687512.55	**-4.12**	**-2.63**	735382.93	4.65	4.65
50_8_2	3326.67	1.76	2.23	36694.32	**-5.54**	**-4.65**	1404418.00	**-5.83**	**-4.83**	1421702.78	4.28	4.30
50_8_4	1828.58	1.13	1.52	51117.93	**-13.81**	**-12.62**	1217983.55	**-13.76**	**-12,87**	1338912.62	4.66	4.80
Average	-	2.25	3.08	-	**-5.18**	**-4.06**	-	**-5.46**	**-4.29**	-	4.21	4.09

tables are organized as follows: first column shows the family of instances; columns named "Best Literature" show the best values found for the family of instances, according to http://soa.iti.upv.es/files/InstancesMOHFS.7z; columns named "Best" contain values of gap between best values and found value of makespan; columns named "Average" contain gap associated to average value of makespan; columns "Time" show average value of execution time (in milliseconds). This table also presents the percentage of final solutions with improved solutions built. Note that negative values shows that computational achieved results are better than literature results.

For instances with 15 tasks, the proposed algorithm showed, in average, a gap in relation to the best values of 0.62% and a gap to the average values of 1.16%. For instances with 50 tasks, the proposed algorithm showed, again in average, a gap to the best values of -5.18% and, to the average values, a gap of -4.06%.

In order to compare the performance in each phase, we note that, for instances with 15 tasks, the generations phase and the local search phase with full representation achieved an improvement of 2.94% to the best solutions generated in construction phase. For instances with 50 tasks, the improvement is 4.21% to the best solutions generated in construction phase.

For instances of 15 tasks, 48 global optimum are known. The proposed algorithm was able to find 33 of them. In 48 test problems in which optimal solutions are not known (instances of 50 tasks), the proposed algorithm could improve the results of the literature in 34 of them.

VI. CONCLUSIONS AND FUTURE WORK

This article studied the Hybrid and Flexible Flowshop Scheduling Problem. The objective was minimize the makespan. The problem was solved with an algorithm based on Evolution Strategies. The initial solution algorithm was generated using: (i) an adaptation of NEH evaluation rule and (ii) IGS metaheuristic.

The main way to search the space of solutions of the algorithm developed is the mutation. It is performed based on two vectors, one associated to probability of applying each type of mutation and other associated with the intensity in which each mutation is applied. In both algorithms, after the mutation is made, a portion of the population of parents is refined by a local search procedure based on the Iterative Improvement Insertion method. Finally, the final population of individuals are refined by a local search method with full representation.

As future work we have proposed to test the generation of initial solutions with other assessment and develop other types of mutation, as well as test the performance of the algorithm in problems of larger size.

VII. ACKNOWLEDGMENTS

The authors would like to thank CEFET/MG, CAPES, CNPq and FAPEMIG for supporting the development of this research.

REFERENCES

[1] R. Ruiz, F. Sivrikaya, and T. Urlings, "Modeling realistic hybrid flexible flowshop scheduling problems," *Computers & Operations Research*, vol. 35, pp. 1151–1175, 2008.

[2] H. G. Beyer and H. P. Schwefel, "Evolution strategies - a comprehensive introduction," *Natural Computing*, vol. 1, pp. 3–52, 2002.

[3] L. Costa and P. Oliveira, "Estratégias evolutivas," in *Manual de Computação Evolutiva e Metaheurísticas*, A. Gaspar-Cunha, C. H. Antunes, and R. Takahashi, Eds. Imprensa da Universidade de Coimbra e Editora da UFMG, 2012, vol. 1, pp. 49–65.

[4] V. N. Coelho, M. F. J. Souza, I. M. Coelho, F. G. Guimaraes, and B. N. Coelho, "Estratégias evolutivas aplicadas a um problema de programação inteira mista," in *Anais do X Congresso Brasileiro de Inteligência Computacional*, Fortaleza, novembro de 2011, 8p, 2011.

[5] L. Costa and P. Oliveira, "Evolutionary algorithms approach to the solution of mixed integer non-linear programming problems," *Computers & Chemical Engineering*, vol. 25, no. 10, pp. 257–266, 2001.

[6] L. Burtseva, R. Romero, S. Ramirez, V. Yaurima, F. F. González-Navarro, and P. F. Perez, "Lot processing in hybrid flow shop scheduling problem," in *Production Scheduling*, R. Righi, Ed. InTech, 2012, pp. 65–96.

[7] I. Ribas, R. Leisten, and J. M. Framiñan, "Review and classification of hybrid flowshop scheduling problems from a production system and a solutions procedure perspective," *Computers & Operations Research*, vol. 37, pp. 1439–1454, 2010.

[8] M. S. Salvador, "A solution to a special class of flowshop scheduling problems," in *Symposium on the theory of scheduling and its applications*. Berlin:Springer, 1973, pp. 83–91.

[9] M. R. Garey and D. S. Johnson, *Computers and intractability: a guide to the theory of NP-completness*. San Francisco: Freeman, 1979.

[10] V. Oduguwa, A. Tiwari, and R. Roy, "Evolutionary computing in manufacturing industry: an overview of recent applications," *Applied Soft Computing*, vol. 5, pp. 281–299, 2005.

[11] S. Parthasarathy and C. Rajendran, "An experimental evaluation of heuristics for scheduling in a real-life flowshop with sequence-dependent setup times of jobs," *International Journal of Production Economics*, vol. 49, pp. 255–263, 1997.

[12] R. Ruiz and C. Maroto, "A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility," *European Journal of Operational Research*, vol. 169, pp. 781–800, 2006.

[13] B. Naderi and R. Ruiz, "The distributed permutation flowshop scheduling problem," *Computers & Operations Research*, vol. 37, pp. 754–768, 2010.

[14] R. Ruiz and C. Andrés-Romano, "Scheduling unrelated parallel machines with resource-assignable sequence-dependent setup times," *International Journal of Advanced Manufacturing Technology*, vol. 57, pp. 777–794, 2011.

[15] R. Ruiz and J. A. Vázquez-Rodríguez, "The hybrid flow shop scheduling problem," *European Journal of Operational Research*, vol. 205, pp. 1–18, 2010.

[16] T. Nishi, Y. Hiranaka, and M. Inuiguchi, "Lagrangian relaxation with cut generation for hybrid flowshop scheduling problems to minimize the total weighted tardiness," *Computers & Operations Research*, vol. 37, pp. 189–198, 2010.

[17] A. Azzi, M. Faccio, A. Persona, and F. Sgarbossa, "Lot splitting scheduling procedure for makespan reduction and machine capacity increase in a hybrid flow shop with batch production," *International Journal of Advanced Manufacturing Technology*, vol. 59, pp. 775–786, 2011.

[18] M. Akhshabi, M. Akhshabi, and J. Khalatbari, "Proposing a clonal selection algorithm for hybrid flow shop," *African Journal of Business Management*, vol. 6, pp. 5744–5749, 2012.

[19] M. Boudhar and N. Meziani, "Two-stage hybrid flow shop with recirculation," *International Transactions in Operational Research*, vol. 17, pp. 239–255, 2010.

[20] S. Carpov, J. Carlier, D. Nace, and R. Sirdey, "Two-stage hybrid flowshop with precedence constraints and parallel machines at second stage," *Computers & Operations Research*, vol. 39, pp. 736–745, 2012.

[21] A. Ziaeifar, R. Tavakkoli-Moghaddam, and K. Pichka, "Solving a new mathematical model for a hybrid flow shop scheduling problem with a processor assignment by a genetic algorithm," *International Journal of Advanced Manufacturing Technology*, vol. 56, pp. 1–11, 2011.

[22] V. Yaurima, L. Burtseva, and A. Tchernykh, "Hybrid flowshop with unrelated machines, sequence-dependent setup time, availability constraints and limited buffers," *Computers & Industrial Engineering*, vol. 56, pp. 1452–1463, 2009.

[23] B. Tadayon and N. Salmasi, "A flexible flowshop scheduling problem with machine eligibility constraint and two criteria objective function," *Engineering and Technology*, vol. 62, pp. 103–108, 2012.

[24] ——, "A two-criteria objective function flexible flowshop scheduling problem with machine eligibility constraint," *International Journal of Advanced Manufacturing Technology*, vol. 60, pp. 1–15, 2012.

[25] B. Naderi, R. Ruiz, and M. Zandieh, "Algorithms for a realistic variant of flowshop scheduling," *Computers & OperationsResearch*, vol. 37, pp. 236–246, 2010.

[26] F. M. Defersha and M. Chen, "Mathematical model and parallel genetic algorithm for hybrid flexible flowshop lot streaming problem," *International Journal of Advanced Manufacturing Technology*, vol. 57, pp. 1–17, 2011.

[27] T. Urlings and R. Ruiz, "Genetic algorithms with different representation schemes for complex hybrid flexible flow line problems," *International Journal of Metaheuristics*, vol. 1, pp. 30–54, 2010.

[28] M. Zandieh, E. Mozaffari, and M. Gholami, "A robust genetic algorithm for scheduling realistic hybrid flexible flow line problems," *Journal of Intelligent Manufacturing*, vol. 21, pp. 731–743, 2010.

[29] R. Ruiz and T. Stützle, "An iterated greedy algorithm for the flowshop problem with sequence dependent setup times," in *The 6th Metaheuristics International Conference*, 2005, pp. 817–826.

[30] T. Urlings, R. Ruiz, and T. Stützle, "Shifting representation search for hybrid flexible flowline problems," *European Journal of Operational Research*, vol. 207, pp. 1086–1095, 2010.

[31] M. Nawaz, E. E. Enscore Jr, and I. Ham, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem," *The International Journal of Management Science*, vol. 11, pp. 91–95, 1983.