A Study concerning the Application of Genetic Algorithms for solving the Multi-Objective Hybrid Flowshop Scheduling Problem

Eduardo C. de Siqueira[1], **Rodney O. M. Diana**[2], **Marcone J. F. Souza**[3], **Sérgio R. de Souza**[2]

[1] Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro
Paracatu, MG – Brazil

[2]Centro Federal de Educação Tecnológica de Minas Gerais
Belo Horizonte, MG – Brazil

[3]Departamento de Computação, Campus Universitário
Universidade Federal de Ouro Preto
Ouro Preto, MG – Brazil

eduardosiqueira@iftm.edu.br, rodneyoliveira@dppg.cefetmg.br,
marcone@iceb.ufop.br, sergio@dppg.cefetmg.br

Abstract. *This article shows the implementations of two metaheuristics based on genetic algorithms for solving the Multi-Objective Hybrid Flowshop Scheduling Problem. The implemented metaheuristics are NSGA-II and SPEA2 and both are known as second generation methods of evolutionary multi-objetive algorithms. The uniform crossover was used for the two metaheuristics. In addition, four different mutation operators are used. The implemented algorithms differ mainly by the mechanisms of evaluation and selection. For evaluation, the hyper-volume and epsilon metrics are used. The found results show the superior behavior of NSGA-II over SPEA2.*

1. Introduction

According to [Pinedo 2008], job scheduling is a decision process that is widely used in many industries and services. This issue deals with the allocation of resources to jobs during periods of time and involves the optimization of one or more objective. The scheduling problems can be described by a triple $\alpha\,|\beta|\gamma$. The field α describes the machine environment and contains only one entry. The field β provides details about the features and processing restrictions and can contain no entry, a single entry or multiple entries. The field γ describes the objective to be optimized, and can contain a single entry or more than one entry (multi-objective).

Multi-objective problems are characterized by involving two or more conflicting objectives simultaneously. Thus, a single solution that optimizes all objectives at the same time can not be found, that is, for this type of problem the challenge is to find a set of efficient solutions, the so-called Pareto frontier. According to [Ticona 2003], usually optimization problems encountered in the real world follow this characteristic.

In this article, a multi-objective scheduling problem in Hybrid Flowshop (HFS) environments is addressed. This problem is described in Section 2. The instances we

have used are the ones proposed by [Urlings 2010]. In this multi-objective problem, the makespan minimization as well as the minimization of the weighted sum of tardiness are considered. This important production problem suffers from the question: an optimal solution for the makespan minimization single objective is very efficient from the point of view of production, however it can be bad for the customer who expects the service to be delivered without delay.

The first study on scheduling in flowshop machine environments emerged in the 1950s from [Johnson 1954] and, since then, has attracted great interest in the scientific community. There are many studies about the scheduling problems. However, a gap between theory and practice always existed. On the other hand, there is a recent tendency to develop solution approaches to real problems [Ruiz et al. 2008]. According to [Ruiz et al. 2008], little effort has been spent to develop complex models for solving job scheduling problems in which many realistic situations are considered together. In this sense, this paper considers very common features in real problems, as the times of release for machines; the presence of unrelated parallel machines at each stage; sequence-dependent setup times; eligibility machines; and latency times (lag) between operations.

As [Naderi et al. 2010], two important questions about HFS draw attention: the determination of the sequence at each stage and the distribution of jobs on the machines at each stage. These same authors presented an algorithm based on the Iterated Local Search (ILS) metaheuristic in order to minimize the makespan.

In [Urlings and Ruiz 2010] and [Zandieh et al. 2010] Genetic Algorithms are proposed for solving this problem. The treated objective is also the minimization of the makespan. Another related work is [Defersha and Chen 2011], where a computational solution based on Genetic Algorithms is implemented in sequential and parallel platforms. [Siqueira et al. 2013] proposed an algorithm based on Evolutionary Strategies. This work dealt with the HFS problem with many real characteristics, with the objective of minimizing the makespan.

Multi-objective algorithms applied for solving flowshop problems are evaluated in [Minella et al. 2008]. However, few studies in the literature have dealt with the hybrid flowshop problem. [Behnamian et al. 2009] carried out a three-phase metaheuristic for solving a Hybrid Flowshop problem with identical machines in each stages and considering the setup times. An interesting work is presented in [Dugardin et al. 2010], where a new algorithm, called L-NSGA, is introduced. In this algorithm, the Pareto Dominance is replaced by the Lorenz dominance. The authors compared the results with those obtained by complete enumeration and adaptations of NSGA-II and SPEA2 algorithms. The provided computational tests show that L-NSGA found better solutions than the other methods. In [Ciavotta et al. 2013] a new algorithm is presented, called Restarted Iterated Pareto Greedy (RIPG), applied to the Multi-objective Sequence Dependent Setup Times Permutation Flowshop Problem. Another works related to the same problem are [Li and Li 2015], where a multi-objective local search algorithm-based decomposition is proposed for treating it, and [Li and Ma 2016], where a multi-objective memetic search algorithm is introduced. An application of the NSGA-II metaheuristic with local search is proposed in [Cunha Campos and Claudio Arroyo 2014] in order to solve a flowshop problem with two objectives and three stages. The objectives dealt with in this work are the flowtime minimization and the total tardiness minimization. A multi-objective hybrid approach us-

ing NSGA-II and VNS metaheuristic is adopted in [Asefi et al. 2014] for makespan minimization and average tardiness minimization in a no-wait flexible flowshop scheduling problem. On the other hand, an algorithm based on Iterated Pareto Greedy metaheuristic is applied in [Ying et al. 2014] for solving the hybrid flowshop scheduling problem also treating two objectives, i.e., makespan minimization and total tardiness minimization.

This article has aimed to adapt and compare two genetic algorithms to solve the Multi-Objective Hybrid Flowshop Scheduling Problem. The first algorithm is an adaptation of the Non-dominated Sorting Genetic Algorithm version II (NSGA-II), proposed in [Deb et al. 2002] as a new version of NSGA, which was introduced in [Srinivas and Deb 1995]. The second algorithm is an adaptation of the Strength Pareto Evolutionary Algorithm version II (SPEA2), showed in [Zitzler et al. 2002] as an improved version of SPEA, proposed in [Zitzler and Thiele 1998]. The rest of the paper is organized as follows: the dealt problem and their main characteristics are defined in Section 2. Section 3 details the genetic multi-objective algorithms proposed to solve the problem. Section 4 presents the results obtained by applying the proposed algorithms to the addressed problem. Last section ends the article and some conclusions and directions for future researches are included.

2. Problem Formulation

Let $N = \{1, 2, 3, ..., n\}$ be a set of jobs that must be performed in a set of stages $M = \{1, 2, 3, \cdots, m\}$. For each stage i, there is a set of unrelated parallel machines. Some jobs can skip stages and this is an important property of this problem. The processing of job j on stage i is called task. A common situation in practice is addressed, in which some tasks can only be performed in certain specialized machines, which, in turn, can only perform a certain task group. The main characteristics of the problem are:

- F_j: set of stages visited by job j, $1 < F_j < m$;
- p_{ilj}: processing time of job j in machine l and stage i;
- E_{ij}: set of eligible machines for the job j at stage i;
- d_j: due date for job j;
- w_j: weight (importance) of the job j;

The following objectives are treated:

- Makespan minimization ($C_{\max}$);
- Minimization of the weighted sum of Tardiness ($\sum w_j T_j$).

The completion time C_j of a job j is the instant in which the last task of this job is completed. Thus, the makespan $C_{\max}$ is the completion time of the last system task, i.e., $C_{\max} = \max_j\{C_j\}$. The tardiness T_j of a job is a defined as $\max(C_j - d_j, 0)$. Each job j is associated with a weight w_j according to its importance.

In order to exemplify the problem, consider an instance with four jobs and two stages, with two machines at each stage. Table 1 shows which machines l are eligible for each job j in each stage i. From this table, we could verify, for example, that job 1 could be performed on machines 1 and 2 at stage 1 and on machine 4 at stage 2. We could also verify that jobs 1, 2 and 4 visit all stages, while job 3 skips stage 1.

Table 2 shows the processing time (p_{ilj}) of each job j at each machine l and at each stage i. In this table, we conclude that the processing time of job 1 at machine 2 and stage 1 is 8 units. The processing time of a job in a non-eligible machine is null.

Table 1. Eligibility.

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

Table 2. Processing Time.

	i	1		2	
	l	1	2	3	4
j	1	10	8	-	21
	2	13	15	45	-
	3	-	-	15	22
	4	-	31	17	12

Table 3 shows the due date (d_j) and the weight (w_j) of each job j. From this table, the due date of job 4 is $d_4 = 51$ and the weight is $w_4 = 5$.

Table 3. Due Date and weight.

		d_j	w_j
j	1	35	4
	2	60	3
	3	48	1
	4	51	5

Figure 1 illustrates a possible sequence for this example. Note that the makespan for this sequence is 65 units and the weighted sum of tardiness is equal to 75 units. In this figure, the vertical axis shows the machines in operation and the horizontal axis shows the production time horizon.

3. Methodology

In this section two genetic multi-objective algorithms are presented. The first algorithm is an adaptation of the Non-dominated Sorting Genetic Algorithm version II (NSGA-II), proposed in [Deb et al. 2002]. The second algorithm is an adaptation of the Strength Pareto Evolutionary Algorithm version II (SPEA2), showed in [Zitzler et al. 2002]. This section is organized as follows: Subsection 3.1 shows as a solution is represented. Subsection 3.2 shows how the mutation operators proposed are defined. Subsection 3.3 explains how to generate the initial population for the two algorithms, while Subsection 3.4 presents the uniform crossover operator used in this work. Subsection 3.5 details the adapted NSGA-II algorithm and Subsection 3.6 details the adaptation of SPEA2 for solving the problem.

3.1. Solution Representation

An individual (i.e., a solution) ind of the problem is represented by a list vector, where each position of this vector is a machine and the list shows the sequence of jobs to be performed in this machine, in the order in which they appear.

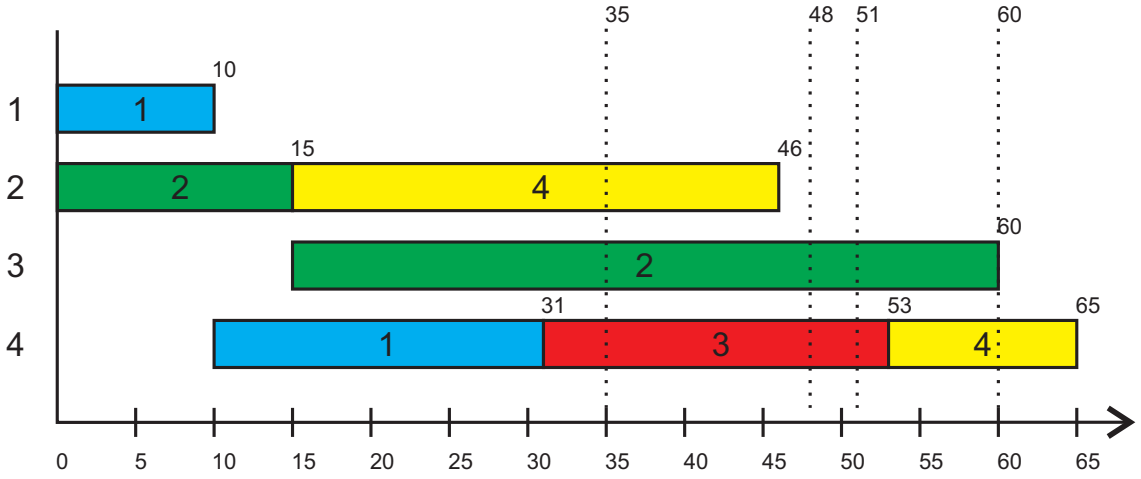

Figure 1. GANTT Diagram. $C_{max} = 65$. $\sum w_j T_j = 75$

Figure 2 shows a representation of a solution for the problem. The task sequence of machine 1 in the first stage is 3, 1, 7, 10 and 4, while in machine 2 is 6, 5, 2 and 9. In the second stage, the task sequences are 1, 3, 2 and 4, for machine 3; and 7, 5, 10, 8 and 9, for machine 4.

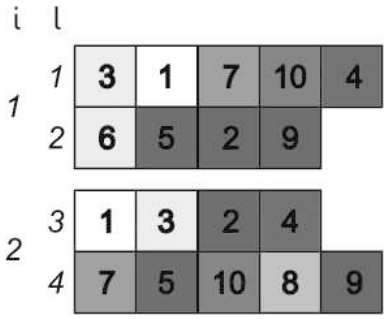

Figure 2. Representation of an individual.

3.2. Types of Mutation

Four types of mutation are used as search operators:

- Exchange in the sequence: this operation consists of performing, in a given machine, the position exchange between two jobs of the sequence;
- Reallocation in the sequence: this operation consists of choosing a job in a given machine and reallocate it to a new position in the sequence;
- Swap of machine: this operation chooses two jobs of a given stage and swap the machines that perform them;
- Reallocation of a machine: this operation consists of reallocating a job of a given stage to a new machine.

3.3. Initial Solutions

The initial solutions are generated by a random method that operates as follows. First, for each stage i, a job j is randomly selected, without repetition, among the jobs that pass through that stage, i.e., $i \in F_j$. Following, a machine l is randomly selected among the

eligible machines for the selected job, such that $l \in E_{ij}$. Thus, the job j is allocated to the next position of the sequence of machine l. These steps are repeated until all tasks are allocated.

3.4. Uniform Crossover

At the uniform crossover, two parents (*parent*1 and *parent*2) are given to produce two offsprings (*offspring*1 and *offspring*2). Its operation is explained below and illustrated in Figure 3. From a binary vector randomly generated, in the positions where the vector element is equal to 1, the *offspring*1 inherits the genes from *parent*1 and the *offspring*2 inherits the genes from *parent*2. The remaining job, that were not considered, are allocated in the order in which they appear in the *parent*2 for *offspring*1 and in the *parent*1 for *offspring*2.

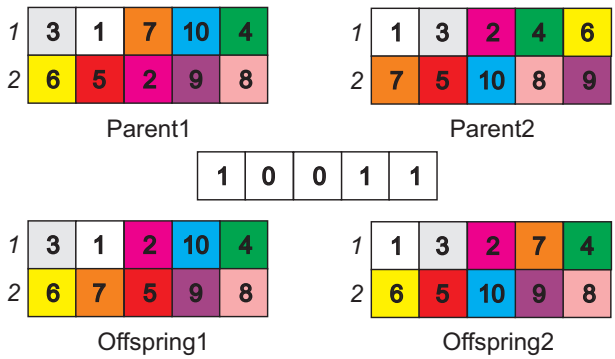

Figure 3. Uniform Crossover.

3.5. The adapted NSGA-II Algorithm

The adaptation of the NSGA-II proposed for solving the problem under analysis is showed in Algorithm 1.

Firstly, the proposed algorithm (lines 1 to 3 of Algorithm 1) generates an initial population with M individuals, built by the method described in Subsection 3.3.

After this first phase, the algorithm goes to an iterative phase (lines 4 to 13), which consists in applying iteratively on the population of individuals the procedures of crossover, mutation and selection until a stopping criterion is attained.

The crossover process is applied to the population in line 7 of Algorithm 1. In this procedure, M offsprings are generated through the choice of two parents for each offspring (linha 6). Every parent is chosen by binary tournament method. After the selection of parents, the crossover operator is applied to generate a offspring with a probability *probCross*. The parent selection and the crossover application are repeated until M offsprings are generated.

The binary tournament is used to select parents for crossover procedure applied in the algorithm NSGA-II. Initially, two individuals of the population are selected. Then, these individuals are compared each other. If both are part of the same front of dominance, they are evaluated according to the crowding distance, being selected as parent the individual with greater distance. If individuals are from different fronts, the individual

Algorithm 1 Adapted NSGA-II

Input: M, $probCross$, $probMut$, $stoppingCriterion$

1: **for** $w \leftarrow 1$ to M **do**
2: $Pop_w \leftarrow$ generateRandomSolution();
3: **end for**
4: **repeat**
5: **for** $w \leftarrow 1$ to M **do**
6: $[parent1, parent2] \leftarrow$ selectParents(Pop)
7: *Offspring*$_w$ $\leftarrow$ applyCrossover(*parent*1, *parent*2, *probCross*);
8: **end for**
9: **for** $w \leftarrow 1$ to M **do**
10: *Offspring*$_w \leftarrow$ applyMutation(*Offspring*$_w, probMut$);
11: **end for**
12: $Pop \leftarrow Selection(Pop,$ *Offspring*$)$
13: **until** Stopping criterion be attained

Output: Pop;

belongs to the front more dominant, that is, the one that has greater fitness, is chosen as parent. This procedure is performed twice for the selection of two parents.

After the crossover, the generated offsprings are subjected to mutation procedure. The mutation of individuals (line 10 of Algorithm 1) operates as follows. For each offspring, a random real number between 0 and 1 is generated, and checked if this number satisfies the probability condition $probMut$. The choice of the stage, the machine and the job involved in the procedure is random.

In line 12 the method of selection of individuals surviving to the next generation is applied. Initially, a expanded population with $2M$ individuals is created as a result of the union of the population of parents and offsprings. All individuals of the expanded population are evaluated based on sorting by non-dominant fronts. In the follow, the individuals from the lower fronts will form the new population until a maximum of M individuals. This new population is initiated with individuals of the best non-dominated front and continues with the solutions of the second, following with the third and so on. As the population size is fixed, not all fronts will be present in the file. Thus, when the last front is considered to form the population, there may exist a number of individuals in this front which exceeds the size M. If this happens, it is necessary to remove individuals from the last front selected. This is done by eliminating individuals with less crowding distance.

The crowding distance allows to quantify the space around an individual. For this, the perimeter of the hypercube formed by the neighbor solutions to the individual that are located in the same dominance front must be calculated.

3.6. The adapted SPEA2 Algorithm

Algorithm 5 shows the pseudo-code of the adapted SPEA2 algorithm implemented in this work. Regarding to the initial solution for this method, individuals are generated by the same construction method used to generate the initial solution to the adapted NSGA-II algorithm. This method was presented in Subsection 3.3. In lines 1 to 3 the construction

Algorithm 2 Adapted SPEA2

Input: M, X, *probCross*, *probMut*, *stoppingCriterion*

1: **for** $w \leftarrow 1$ to M **do**
2: $Pop_w \leftarrow$ generateRandomSolution();
3: **end for**
4: $File \leftarrow \emptyset$
5: **repeat**
6: $Pop \leftarrow Pop \cup$ *File*
7: evaluateFitness(Pop)
8: $File \leftarrow$ updateFile(Pop)
9: **for** $w \leftarrow 1$ to M **do**
10: $[pai1, pai2] \leftarrow$ selectParents(*File*)
11: *Offspring*$_w \leftarrow$ applyCrossover(*parent*1, *parent*2, *probCross*);
12: **end for**
13: **for** $w \leftarrow 1$ to M **do**
14: $Pop_w \leftarrow$ applyMutation(*Offspring*$_w$, *probMut*);
15: **end for**
16: **until** Stopping criterion be attained

Output: *File*;

method is repeated to create M individuals to be included in the initial population.

The algorithm has a repeating loop between the lines 5 to 16, where, in each iteration, which we call generation, the search space is explored by means of crossover, mutation and selection operators.

Before crossover, mutation and selection procedures, the population is attached to the file, forming a new population. The file starts empty and has a maximum size of X. A fitness value is assigned to each individual of the new population, after being evaluated (line 7). To calculate the fitness value, a force is given to the individual ind. This force is determined by the amount of individuals in the population that ind dominates, i. e., more individuals are dominated by ind, the greater is its force. The fitness of an individual is calculated by summing the forces of all their dominators, added to a density measurement.

In this work, we have used, as density measure, the expression:

$$d = \frac{1}{\sigma_k + 2}$$

where $k = \sqrt{M + X}$ and σ_k is the distance, in the objectives space, from the individual to its kth nearest neighbor.

After the individuals being evaluated the best X are placed on file (line 8). Thus, the algorithm proceeds to crossover procedure, where M offsprings are generated by choosing two parents for each offspring (line 10). Every parent is chosen by binary tournament method. After selecting the parents, the crossover procedure is applied to generate a offspring with a probability $probCross$. The selection of parents and the application of the crossover procedure are repeated until M offsprings are generated.

The binary tournament applied here is different from that applied to the NSGA-

Table 4. Algorithm parameters

Algorithm	M	X	*probCross*	*probMut*
NSGA-II	500	-	80%	10%
SPEA2	500	50	80%	10%

II. Initially, two individuals in the population are selected. Then, these two ones are compared to each other, and the one that has lower fitness is chosen as parent. This procedure is performed twice for the selection of two parents.

After the crossover procedure, the generated offsprings are subjected to mutation procedure. Mutation of individuals (line 14 of Algorithm 2) operates in the same manner as in NSGA-II, however, in the SPEA2, the population is completely replaced by offsprings.

4. Computational Results and Evaluations

The two versions of the proposed algorithms were implemented in C++, using the IDE Netbeans 6. The tests were performed on an Intel Core i7 computer, 2.00GHz with 16GB of RAM under Linux Ubuntu 64-bit operating system.

The set of instances used in the experiments were introduced by [Urlings 2010] and is available in [SOA 2016]. It is composed by 432 instances. These instances are subdivided by the number of jobs (n), machines (m) and stages per machine (m_i), according to the following settings: $n = \{5, 7, 9, 11, 13, 15\}$, $m = \{2, 3\}$ e $m_i = \{3\}$. Each possible combination of these settings consists of 32 instances.

Table 4 shows the values of the parameters used by the NSGA-II and SPEA2 algorithms, developed in this work. The values of these parameters were determined empirically. The runtime was stopping criteria used in the experiments, defined by the following equation: $25 \times n \times m_i \times m$ milliseconds, where n is the number of jobs and m_i is the number of stages per machine.

Every instance was evaluated 10 times for each algorithm. The results were compiled, normalized and compared using hypervolume and *epsilon* metrics. The hypervolume metric $H(Q_{alg}, R_0)$ [Zitzler and Thiele 1998] measures the volume between the Pareto front Q_{alg} achieved by the algorithm and a point of reference R_0. An area with higher volume indicates both a greater scattering of the solutions as increased convergence of the same. Thus, a high value of hypervolume is desired. The point of reference R_0 was set for each instance as the highest value of makespan and tardiness contained in the set formed by union of all solutions found after all the experiments performed in this work. Already the $epsilon$, $I_e(Q_{alg}, Q_{ref})$ indicator [Zitzler et al. 2003] determines a minimum e factor, if multiplied by each point of the set Q_{ref}, makes weakly dominated by Q_{alg} the set of approximations resultant. Since Q_{ref} is the Pareto optimal frontier, hence, the lower the value of e, the higher the algorithm convergence. As the Pareto optimal frontier of each instance is not known in this work, Q_{ref} is defined as non-dominated points of the set formed by union of all executions performed for each tested instance. Both metrics were calculated using the EMOA package of statistical computing software R.

Table 5 shows the averages ($\pm$ standard deviation) of the hypervolume and epsilon metrics grouped by combinations of the number of jobs n, number of machines m and

Table 5. Results for hypervolume and $epsilon$ metrics

Group	HYPERVOLUME		EPSILON	
	NSGA-II	SPEA2	NSGA-II	SPEA2
5x3x3	33.35 ± 3.01	30.46 ± 2.96	0.44 ± 0.29	0.48 ± 0.03
5x3x3	99.94 ± 0.08	98.68 ± 1.55	0.91 ± 0.65	1.51 ± 1.29
7x2x3	99.90 ± 0.22	98.94 ± 0.64	0.89 ± 0.77	1.49 ± 0.81
7x3x3	99.96 ± 0.05	95.36 ± 2.87	0.27 ± 0.46	3.15 ± 1.85
9x2x3	99.89 ± 0.30	97.19 ± 1.51	0.50 ± 2.96	0.13 ± 0.65
9x3x3	99.90 ± 0.11	96.92 ± 1.48	0.52 ± 0.66	2.44 ± 1.40
11x2x3	99.97 ± 0.03	93.99 ± 1.93	0.06 ± 0.24	7.32 ± 4.10
11x3x3	99.81 ± 0.54	90.32 ± 3.03	0.63 ± 1.03	6.14 ± 2.27
13x2x3	99.87 ± 0.10	90.60 ± 3.26	0.76 ± 0.99	6.27 ± 2.74
13x3x3	99.87 ± 0.11	85.56 ± 3.91	0.74 ± 1.19	8.64 ± 2.70
15x2x3	99.89 ± 0.09	92.45 ± 3.31	0.78 ± 1.05	4.93 ± 2.34
15x3x3	99.88 ± 0.13	91.62 ± 3.80	0.87 ± 1.14	6.16 ± 2.86

stages by machine m_i. The first column represents the groups of instance, the second and third columns show the value of hypervolume metric for the NSGA-II and SPEA2 algorithms respectively. The fourth and fifth columns show the values obtained for the indicator $epsilon$. Note that, for the set of instances of smaller dimension (5 and 7 jobs), both algorithms have reached very similar values in both hypervolume metric as in the $epsilon$ indicator. As the dimensionality of the instances grows (11, 13, 15 jobs), clearly verifies a gap in the performance of the algorithms. In these instances, the NSGA-II has better convergence than SPEA2, fact indicated by the low value of $epsilon$ indicator and the high value of hypervolume metric. The latter metric also indicates that there is a better spread of solutions by the NSGA-II.

5. Conclusions and Future Works

This article studied the multi-objective hybrid flowshop problem. The considered objectives were to minimize the makespan and the weighted sum of tardiness. The evolutionary algorithms NSGA-II and SPEA2 were used for solving it.

In both algorithms, the initial population is composed of individuals randomly constructed. These algorithms explore the solution space through crossover, mutation and selection operators. Four different types of mutations were implemented and each of them is applied with a certain probability. The mutation is performed in both algorithms considering all four of these types of mutation. The uniform crossover was implemented for both algorithms. The algorithms differ primarily by the selection mechanisms. In NSGA-II, the dominance front and the crowding distance are used to evaluate and select individuals, while in SPEA2 there is a fitness calculation based on the amount of dominated solutions and density of the solution. Furthermore, SPEA2 based algorithm maintains a file with the best individuals and the selection operator acts only on these individuals.

Two metrics were used to compare the algorithms: hypervolume and $epsilon$. For the set of test problems with smaller dimensions, the algorithms showed similar perfor-

mance. For the set of test problems with larger dimensions, however, NSGA-II showed better convergence and better scattering of the solutions.

For future work, it is proposed to implement other algorithms for generation of initial solutions, the development of other types of mutation and crossover operators as well as the insertion of local search techniques for refinement of the solutions.

Acknowledgements

The authors would like to thank FAPEMIG, CNPq, CAPES, CEFET-MG, UFOP and IFTM for supporting the development of the present study.

References

Asefi, H., Jolai, F., Rabiee, M., and Araghi, M. T. (2014). A hybrid NSGA-II and vns for solving a bi-objective no-wait flexible flowshop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 75(5-8):1017–1033.

Behnamian, J., Ghomi, S. M. T. F., and Zandieh, M. (2009). A multiphase covering pareto-optimal front method to multi-objective scheduling in a realistic hybrid flowshop using a hybrid metaheuristic. *Expert Systems with Applications*, 36(8):11057–11069.

Ciavotta, M., Minella, G., and Ruiz, R. (2013). Multi-objective sequence dependent setup times permutation flowshop: A new algorithm and a comprehensive study. *European Journal of Operational Research*, 227(2):301–313.

Cunha Campos, S. and Claudio Arroyo, J. E. (2014). NSGA-II with iterated greedy for a bi-objective three-stage assembly flowshop scheduling problem. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 429–436. ACM.

Deb, K., Pratap, A., Agarwal, S., , and Meyarivan, T. (2002). A fast elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6:182–197.

Defersha, F. M. and Chen, M. (2011). Mathematical model and parallel genetic algorithm for hybrid flexible flowshop lot streaming problem. *International Journal of Advanced Manufacturing Technology*, 57:1–17.

Dugardin, F., Yalaoui, F., and Amadeo, L. (2010). New multi-objective method to solve reentrant hybrid flow shop scheduling problem. *European Journal of Operational Research*, 203(1):20–31.

Johnson, S. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, 1:61–68.

Li, X. and Li, M. (2015). Multiobjective local search algorithm-based decomposition for multiobjective permutation flow shop scheduling problem. *IEEE Transactions on Engineering Management*, 62(4):544–557.

Li, X. and Ma, S. (2016). Multi-objective memetic search algorithm for multi-objective permutation flow shop scheduling problem. *IEEE Access*, 4:2154–2165.

Minella, G., Ruiz, R., and Ciavotta, M. (2008). A review and evaluation of multiobjective algorithms for the flowshop scheduling problem. *INFORMS Journal on Computing*, 20(3):451–471.

Naderi, B., Ruiz, R., and Zandieh, M. (2010). Algorithms for a realistic variant of flowshop scheduling. *Computers & OperationsResearch*, 37:236–246.

Pinedo, M. L. (2008). *Scheduling: Theory, Algorithms, and Systems*. Springer, 3 edition.

Ruiz, R., Sivrikaya, F., and Urlings, T. (2008). Modeling realistic hybrid flexible flowshop scheduling problems. *Computers & Operations Research*, 35:1151–1175.

Siqueira, E. C., Souza, M. J. F., de Souza, S. R., de França Junior, M. F., and Marcelino, C. G. (2013). An algorithm based on evolution strategies for makespan minimization in hybrid flexible flowshop scheduling problems. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 989–996, Cancún, Junho de 2013.

SOA (2016). Sistemas de optimización aplicada, accessed on May 21 2016, from http://soa.iti.es/problem-instances.

Srinivas, N. and Deb, K. (1995). Multi-objective function optimization using non-dominated sorting genetic algorithms. *Evolutionary Computation*, 2(3):221–248.

Ticona, W. G. C. (2003). *Algoritmos evolutivos multi-objetivo para a reconstrução de árvores filogenéticas*. PhD thesis, ICMC, USP, São Carlos.

Urlings, T. (2010). *Heuristics and metaheuristics for heavily constrained hybrid flowshop problems*. PhD thesis, Universidad Politécnica de Valencia.

Urlings, T. and Ruiz, R. (2010). Genetic algorithms with different representation schemes for complex hybrid flexible flow line problems. *International Journal of Metaheuristics*, 1:30–54.

Ying, K.-C., Lin, S.-W., and Wan, S.-Y. (2014). Bi-objective reentrant hybrid flowshop scheduling: an iterated pareto greedy algorithm. *International Journal of Production Research*, 52(19):5735–5747.

Zandieh, M., Mozaffari, E., and Gholami, M. (2010). A robust genetic algorithm for scheduling realistic hybrid flexible flow line problems. *Journal of Intelligent Manufacturing*, 21:731–743.

Zitzler, E., Laumanns, M., and Thiele, L. (2002). SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In Giannakoglou, K. C. et al., editors, *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*, pages 95–100. International Center for Numerical Methods in Engineering (CIMNE).

Zitzler, E. and Thiele, L. (1998). An Evolutionary Approach for Multiobjective Optimization: The Strength Pareto Approach. TIK Report 43, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, SW.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and da Fonseca, V. G. (2003). Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132.