NSGA-II with Iterated Greedy for a Bi-objective Three-stage Assembly Flowshop Scheduling Problem

Saulo Cunha Campos
Departamento de Informática
Universidade Federal de Viçosa (UFV)
Campus Universitário, Viçosa-MG
+55 (31) 3899-2394, Brazil, 36.570-000
saulo.campos@ufv.br

José Elias Claudio Arroyo
Departamento de Informática
Universidade Federal de Viçosa (UFV)
Campus Universitário, Viçosa-MG
+55 (31) 3899-2394, Brazil, 36.570-000
jarroyo@dpi.ufv.br

ABSTRACTT

In this paper we address a three-stage assembly flowshop scheduling problem where there are *m* machines at the first stage, a transportation machine at the second stage and an assembly machine at the third stage. At the first stage, different parts of a product are manufactured independently on parallel production lines. At the second stage, the manufactured parts are collected and transferred to the next stage. At the third stage, the parts are assembled into final products. The objective is to schedule *n* jobs on the machines so that total flowtime and the total tardiness of the jobs are minimized simultaneously. This problem has many applications in industry and belongs to the class of NP-Hard combinatorial optimization problems. In order to obtain near Pareto optimal solutions, we propose an Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II) coupled with Iterated Greedy (IG) strategy. IG is a simple heuristic that has shown excellent results for different flowshop scheduling problems. A comparative study is presented between the results obtained using the standard NSGA-II, the enhanced NSGA-II with IG approach and a single-objective GRASP heuristic. Experimental results on both medium and large size of instances show the efficiency of the hybrid NSGA-II approach.

Keywords

Assembly Flowshop Scheduling, Multi-objective Optimization, Heuristics, Genetic Algorithms, Local Search.

1. INTRODUCTION

Jobs scheduling is a decision-making problem that occurs in manufacturing systems. This problem deals with the allocation of available resources to jobs over given time periods and the goal is to optimize one or more objectives (or criteria).

The scheduling problems have been thoroughly studied since the mid-50 [3]. Nowadays, scheduling problems are one of the most studied problems. It occurs mainly by two aspects: the first one concerns their practical importance, with various applications in several industries. The second aspect is about the difficulty for solving the majority problems of this class (these problems belong to the class NP-Hard).

The scheduling problem focused in this paper is the three-stage Assembly Flowshop Scheduling (3sAFS) problem. In this problem, *n* jobs (or products) are performed in three stages. At the first stage, the different parts of a job are manufactured independently on *m* parallel machines (each job has *m* parts), at the second stage the produced parts are collected and transferred from the production site to the assembly site, and at the last stage they are assembled into final products. At the second and third stages there is only a single machine. The production shop is composed of three stages that are disposed in series. Each job visits each stage in order, characterizing a permutation flowshop. This problem is NP-hard since its special case when $m = 1$ (which is a regular three-machine flowshop scheduling problem) is NP-hard [10].

In order to make the problem real, we consider sequence dependent setup times on machines of the first stage, transportation machine and assembly machine. Setup time is necessary to prepare the machine (for example, tooling, cleaning, positioning accessories, inspection of materials, among others) when a job *j* is processed immediately after another job *i*.

The 3sAFS problem addressed in this work is to find the processing sequence of jobs (schedule) in order to minimize simultaneously two objectives: the total flowtime and the total tardiness. The goal is to provide the decision maker with a set of efficient schedules (Pareto-optimal solutions) such that he may choose the most suitable schedule.

In the literature, other versions of the Assembly Flowshop Scheduling (AFS) problem have been studied by some authors. Some solution approaches have been proposed to solve the single objective two-stage AFS (2sAFS) problem. For the maximum job completion time (makespan) minimization some heuristics methods were developed by Potts et al. [23]. Al-Anzi and Allahverdi [1] consider the total completion time minimization and propose three metaheuristics, simulated annealing, tabu search, and a hybrid tabu search. Maximum lateness criterion is minimized by Allahverdi and Al-Anzi [2]. They consider sequence independent setup times and propose a self-adaptive differential evolution heuristic.

The single objective 3sAFS problem has been also addressed by some authors. For minimizing the makespan, Koulamas and Kyparisis [15] analyze the worst-case ratio bound for several constructive heuristics. Koulamas and Kyparisis [15] extended the 2sAFS problem to 3sAFS problem with the objective of minimizing the makespan. For the 3sAFS problem with sequence dependent setup times on first and third stages, a mathematical MIP model for minimizing the total completion time is proposed by Andrés and Hatami [4].

The 3sAFS problem with multi-objective optimization has been less studied. Some authors address bi-objectives problems but

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.

http://dx.doi.org/10.1145/2576768.2598324

they use weighted linear functions, that is, weights (preferences) are defined to each objective and them are combined in a linear function.

To minimize the weighted sum of the mean flowtime and maximum tardiness in a 3sAFS problem with sequence dependent set up times, a tabu search and a simulated annealing metaheuristic were developed by Hatami et al. [12]. For the same problem, a Greedy Randomized Adaptive Search Procedure (GRASP) was proposed by Campos et al. [7]. Maleki et al. [17] consider a 3sAFS problem with blocking and sequence dependent setup times with the objective to minimize the weighted mean completion time and makespan. These authors propose a metaheuristic based on simulated annealing.

Recently, Tajbakhsh et al. [26] apply multi-objective algorithms based on genetic algorithm and particle swarm optimization for a bi-objective 3sAFS problem with the objective to minimize the makespan and sum of the earliness and tardiness costs, simultaneously.

In this paper, multi-objective optimization is understood in its traditional form, which involves the generation of the Pareto-optimal or efficient solutions. To solve the bi-objective 3sAFS problem, we propose a hybrid algorithm based on NSGA-II (Elitist Non-dominated Sorting Genetic Algorithm) and Iterated Greedy (IG) algorithm.

NSGA-II proposed by [8] has been widely used for solving a variety of multi-objective optimization problems. IG algorithm, proposed by [25], is a powerful heuristic that has been applied to all sorts of scheduling problems obtaining high quality results. The main feature of the IG is its simplicity which is contrary to sophisticated algorithms that embed problem specific knowledge and that usually have many control parameters. Despite its simplicity, IG has shown state-of-the-art results under different flowshop variants and objectives [21].

In the literature, some authors show that the performance of the NSGA-II algorithm can be improved by using local search techniques [9] [11] [22]. To the best of our knowledge this is the first paper that combines NSGA-II with IG.

The rest of this paper is organized as follows. Section 2 presents the 3sAFS problem statement. In Section 3 we describe the proposed algorithm. The computational experiments and the statistical analyses of the obtained results are presented in Section 4. The last Section concludes the work.

2. PROBLEM STATEMENT

The 3sAFS problem investigated in this work consists in processing (or manufacturing) n jobs (products) on three stages production shop. Each job has m components (or parts) that are manufactured separately at the first stage (production line composed by m parallel machines). At the second stage, the produced parts of a job are collected and transported from production site to assembly machine. At the third stage, the parts are assembled into a final product. All jobs are available to be processed at time zero. All machines (including transportation and assembly machines) process only one job at a time without interruption or preemption. In addition, the sequence of jobs at all stages is the same (only permutation schedules are considered).

The processing time of a job j on machine k of the first stage is known and defined by $t_{[j,k]}$. This time corresponds to the manufacturing time of part k. The transportation time ($tt_{[j]}$), assembly time (at[j]) and the due date ($d_{[j]}$) of a job j also are known.

Between the processing of two consecutive jobs i and j, on a machine k of the first stage, a sequence dependent setup time $s1_{[k,i,j]}$ is considered. The setups times on the transportation and assembly machines are defined by $s2_{[i,j]}$ and $s3_{[i,j]}$, respectively, where i and j are two consecutive jobs.

The objective of the problem is to determine a permutation schedule s (sequence of the jobs) in order to minimize the total flowtime (F) and the total tardiness of the jobs (T).

The two objectives F and T are computed by the following equations:

$$f_1 = F = \sum_{j=1}^{n} C_{3[j]}$$

$$f_2 = T = \sum_{j=1}^{n} max\left\{0,\ C_{3[j]} - d_j\right\}$$

where, $C_{3[j]}$ is the completion time of job j at the third stage. $C_{3[j]}$ is calculated by the following equation:

$$C_{3[j]} = \max\left\{C_{2[j]},\ C_{3[j-1]}\right\} + s3_{[j-1,j]} + at_{[j]}\ ,$$

where, $C_{1[j]}$ and $C_{2[j]}$ are the completion times of job j at the first and second stage, respectively. j-1 is the immediately preceding job of job j in a sequence. $C_{1[j]}$ and $C_{2[j]}$ are calculated by the following equations:

$$C_{1[j]} = \max_{k=1,\ldots,m}\left\{\sum_{i=1}^{j} s1_{[k,i-1,i]} + t_{[j,k]}\right\}$$

$$C_{2[j]} = max\left\{C_{1[j]},\ C_{2[j-1]}\right\} + s2_{[j-1,j]} + tt_{[j]}\ .$$

The optimized objectives are very important in manufacture systems. The first objective F is related to minimizing work-in-process inventories, while the second objective T is related to job due dates (customer service). These objectives are often of conflicting nature, usually there is not a single solution s optimizing F and T at once. Optimality in multi-objective optimization problems is therefore understood in the sense of Pareto-optimality, and the resolution of multi-objective optimization problems lies in the identification of all solutions belonging to the Pareto or efficient set, containing all alternatives s which are not dominated by any other alternative s'. To properly compare two solutions in a bi-objective minimization problem, the following definitions are used:

- A solution s dominates s' if the point ($f_1(s)$, $f_2(s)$) dominates ($f_1(s')$, $f_2(s')$), that is, $f_i(s) \leq f_i(s')$ for i=1, 2, and $f_i(s) < f_i(s')$ for at least one i.
- A solution s is Pareto-optimal (or efficient) if there is no s' such that ($f_1(s')$, $f_2(s')$) dominates ($f_1(s)$, $f_2(s)$).

In Figure 1, we present two schedules (solutions) for an instance of the 3sAFS problem with n = 4 jobs. Dark blocks denote the required setup times. In this instance the first stage has m = 2 machines (M1 and M2). The transportation and assembly machines are Mt and Ma, respectively. The due date of jobs are d_1 = 20, d_2 = 15, d_3 = 43 and d_3 = 33. The processing times of jobs

and the setups times are showed in the Figure 1. For this instance, the set of Pareto optimal solutions is formed by the following four job schedules: s_1 = {2, 1, 4, 3}, s_2 = {4, 2, 1, 3}, s_3 = {3, 4, 2, 1} and s_4 = {4, 3, 2, 1}. The objective values of these schedules are ($F(s_1)$,$T(s_1)$) = (114, 6), ($F(s_2)$, $T(s_2)$) = (108, 16), ($F(s_3)$, $T(s_3)$) = (106, 28) and ($F(s_4)$, $T(s_4)$) = (104, 29), respectively. Note that the schedules s_1 and s_4 (showed in Figure 1) provide the minimum value of the total tardiness (T) and the total flowtime (F), respectively. Figure 2 shows the Pareto optimal front for the considered instance. Note that, the objectives F and T are conflicting, that is, there is not a single schedule that minimizes F and T at once.

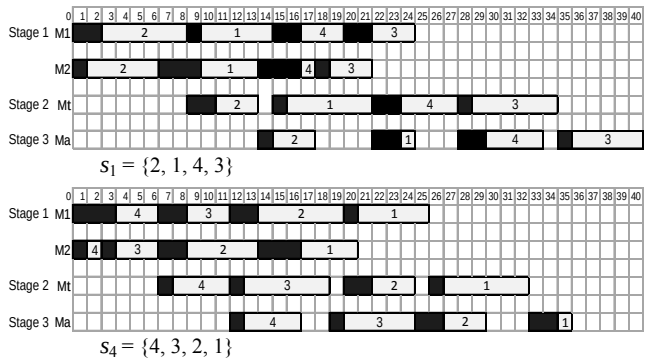

Figure 1: Two schedules for an instance of the 3sAFS problem.

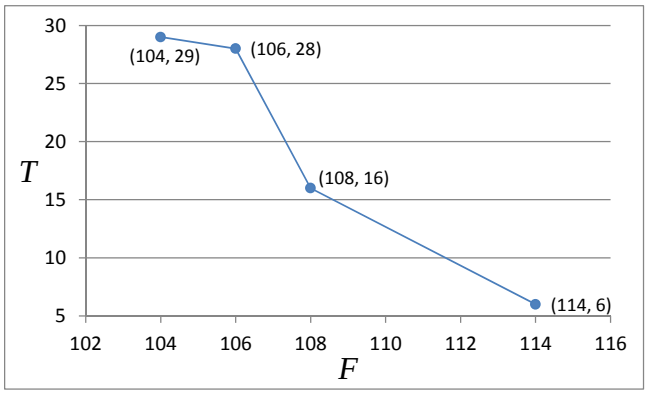

Figure 2: Pareto optimal front for an instance with *n*=4 jobs.

The goal of the bi-objective 3sAFS problem, addressed in this paper, is to determine the set of the Pareto-optimal permutation schedules in order to minimize F and T.

3. NSGA-II WITH ITERATED GREEDY

In this section we describe the proposed hybrid algorithm that combines the standard NSGA-II [8] with Iterated Greedy (IG) strategy. The proposed algorithm is named NSGA2-IG and is showed in Figure 3. The pseudocode description of NSGA2-IG is presented in Algorithm 1. The algorithm has three input parameters, N (size of a population), d (parameter used in the IG procedure) and *MaxCPUTime* (an amount of CPU time used as stopping criterion).

Initially (iteration t = 0), two solution populations P_t and Q_t, each of size N, are created. The solutions are generated randomly and by using a constructive heuristic.

The IG procedure begins with a non-dominated solution s selected randomly from P_t, executes a greedy local search and returns a population D' of non-dominated solutions. At the iteration t = 0, the IG procedure is not executed (D' = ∅).

The three populations P_t, Q_t and D' are combined to form a population R_t of size $2N + |D'|$, where $|P_t| = |Q_t| = N$. Next, a non-dominated sorting is used to classify the entire population $R_t = P_t \cup Q_t \cup D'$. Once, the non-dominated sorting is over, the population R_t becomes subdivided in k fronts (F_1, F_2,…, F_k). All elements of the same front have the same non-domination rank. The best N solutions of the best non-dominated fronts are chosen to be the next generation parent solutions P_{t+1}. The N members of the parent population P_{t+1} are chosen from the first l non-dominated fronts F_1, F_2,…, F_{l-1}, F_l ($l \leq k$), such that $|F_1 \cup F_2 \cup \ldots \cup F_{l-1}| < N$. If $|F_1 \cup F_2 \cup \ldots \cup F_l| = N$, the parent population is $P_{t+1} = F_1 \cup F_2 \cup \ldots \cup F_l$, and the solutions of $F_{l+1} \cup \ldots \cup F_k$ are rejected. If $|F_1 \cup F_2 \cup \ldots \cup F_l| > N$, the solutions of the last front F_l are sorted using the crowded distance comparison operator in descending order. The worst $|F_1 \cup F_2 \cup \ldots \cup F_l| - N$ solutions of front F_l are rejected. The solutions of $F_{l+1} \cup \ldots \cup F_k$ are also rejected. The new population P_{t+1} is now used for selection, crossover and mutation operators and to create a child population Q_{t+1} of size N. At next iteration (t=1), the IG procedure is executed. IG begins with a solution s selected randomly from F_1 ($F_1 \subset P_t$).

The above mentioned mechanism is continued until the stopping criterion is satisfied. The overall non-dominated solutions are kept as the final result.

```
Algorithm 1. NSGA2-IG(N, d, MaxCPUTime)
Inputs:
   N: Population size
   d: Parameter used in the IG strategy
   MaxCPUTime: Maximum CPU time that algorithm run
Output:
   D: Set of non-dominated solutions
Begin
  t ← 0; // iteration counter
  Pt ← ∅; //population of N solutions
  Qt ← ∅; //population of N child solutions
  D' ← ∅; //non-dominated solutions found by IG
  Rt ← ∅; //R=Pt∪Qt∪D', population to be classified
  CPUTime ← 0;
  Pt ← GenerateInitialPopulation(N);
  Qt ← GenerateInitialPopulation(N);
  while (CPUtime < MaxCPUTime) do
     if ( t ≥ 1) then
        s ← SolutionSelectedRandomly(F1);
        D' ← IG( s, d ); //IG method
     end-if
     Rt ← Pt ∪ Qt ∪ D';
     F1,F2...Fk ← Non_DominatedSorting( Rt );
     CrowdingDistanceSorting( Rt );
     Pt+1 ← Choose_The_Best_N_Solutions( Rt );
     Qt+1←SelectionCrossoverMutation(Pt+1,N);
     t ← t+1;
     updateCPUTime(CPUTime);
  end_while
  D ← F1;
  Return D;
End
```

In the next subsections, we describe in detail the population initialization phase, selection, crossover, mutation and the Iterated Greedy intensification.

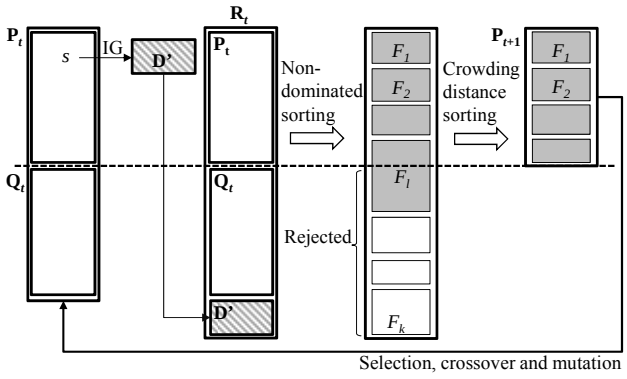

Figure 3. NSGA2-IG basic iteration.

3.1 Initialization

A solution (sequence of n jobs) is represented by an array with n elements. The NSGA2-IG algorithm begins with a population with $2N$ solutions. Part of this population is generated by the multi-objective partial enumeration heuristic (MOPE) proposed by Arroyo and Armentano [5]. The other part of the population was generated randomly.

MOPE heuristic is inspired on the NEH heuristics proposed by Nawaz [20]. MOPE uses the Pareto dominance concept to maintain not just one incomplete partial solution at each iteration (as in NEH), but a whole set of non-dominated partial solutions generated during the job insertion process. To minimize the combination of objectives total flowtime and total tardiness, the MOPE heuristic uses the following dispatching rules:

SPT (shortest processing time): the jobs are arranged in increasing order of the total processing time on the three stages.

TLB (tardiness lower bound): the jobs are arranged in decreasing order of a tardiness lower bound of job i [5].

The MOPE heuristic generates N_h (<$2N$) non-dominated solutions. Then, $2N$ - N_h solutions are generated randomly.

3.2 Selection, Crossover and Mutation

To create a child population Q_t, the following operators are executed: selection, recombination and mutation.

From two parent solutions, two child solutions are created. Each parent solution is selected from population P_t by using the binary tournament technique. This technique consists in choosing randomly two solutions of the population and the best one is selected. The solutions belonging to smaller fonts are always better. When comparing two solutions belonging to the same front, the one with the bigger crowding distance wins.

The crossover operator creates offspring solutions by coalescing two parents solutions. The aim is to generate better children, i.e. to generate better solutions after the crossover. Many different general and specific crossover operators have been proposed for the flowshop scheduling problems in the literature. In this paper, we use two effective operators: Two-Point Crossover (TPX) [13] and Similar Block 2-Point Order Crossover (SB2OX) [24].

The mutation operator used here is the interchange operator. This operator randomly choses two points along the child chromosome and swaps with each other and genes (jobs) in these two points are reversed. In each child solution, two swaps are performed with a mutation probability.

3.3 Iterated Greedy Intensification

To accelerate the convergence speed of the NSGA-II algorithm, we employ an intensification procedure based on the Iterated Greedy (IG) heuristic. IG is a simple method which has been applied to different sorts of single-objective scheduling problems. The first application of IG for flowshop scheduling problems was given by Ruiz and Stützle [25]. Minella, et al. [19] propose an adaptation of IG for multi-objective flowshop scheduling problems. Arroyo et al. [6] used the same intensification procedure in a multi-objective VNS heuristic to solve a single machine scheduling problem.

In this paper, the intensification procedure based on IG begins from a solution s selected randomly from the best set of non-dominated solutions (front F_1 obtained by the non-dominated sorting strategy). The pseudocode description of the used IG intensification is showed in Algorithm 2. The procedure is composed of two stages: destruction and construction. In the destruction stage, d jobs (selected randomly) are removed from s and a partial solution s_p (of size n-d) is obtained. The removed jobs are stored in J (where $J_{[i]}$, i=1,...,d, are the removed jobs). The construction stage has d steps. In step i=1, (n-d+1) partial solutions are constructed by inserting job $J_{[1]}$ in all possible positions of s_p. From the (n-d+1) partial solutions, the set D_i of non-dominated partial solutions are selected. In the next step, new solutions (of size n-d+2) are obtained by inserting job $J_{[2]}$ in each partial non-dominated solutions. From the solutions constructed in each step, the non-dominated solutions are always selected. In step d, a set D' of complete non-dominated solutions is determined. Figure 4 illustrates the idea of the greedy intensification procedure that starts from a solution s with n = 4 jobs and considering d =2.

```
Algorithm 2. IG (s, d)
Inputs:
   s: solution chosen randomly from F1
   d: number of jobs to be removed from s
Output:
   D': A set of non-dominated solutions obtained from s
begin
   J ← ∅
   //Destruction stage
   for (i ← 1 to d)
      J[i] ←remove a randomly selected job from s;
   sp ← s; //partial solution with n-d jobs
   //Construction stage:
   D' ← {sp}
   //Construction of non-dominated solution:
   for ( i ← 1 to d)
      Di ← ∅;
      for ( each partial solution s' of D')
           Ds' ← non-dominated solutions obtained
               after inserting job J[i] in all
               possible positions of s';
           Di ← non-dominated solution Di ∪ Ds';
      end-for
      D' ← Di;
   end-for
   Return D';
end
```

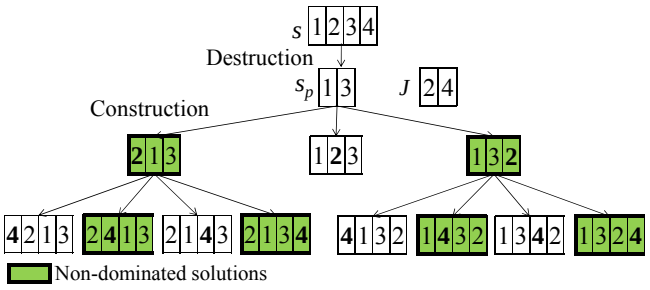

Figure 4. Iterated Greedy Intensification.

4. COMPUTATIONAL EXPERIMENTS

In this work, we analyze the efficiency of the IG intensification procedure used in the NSGA-II algorithm proposed by Deb et al. [8]. To solve the bi-objective three-stage assembly flowshop scheduling problem addressed in this work, we compare the hybrid NSGA2-IG algorithm with the standard NSGA-II and with a single-objective GRASP algorithm proposed by [7]. All algorithms were coded in C++ and executed on an Intel(R) Core Quad Q9400 2.67GHz and 8 GB of RAM.

The single-objective GRASP algorithm is based on the minimization of different weighted utility functions ($f_w = w_1 \times F + w_2 \times T$, where $w_1 + w_2 = 1$). To determinate a set of non-dominated solutions, the GRASP algorithm was executed with different weight vectors (w_1, w_2). We used the following eleven weights: (w_1, w_2) = (0, 1), (0.1, 0.9), (0.2, 0.8), (0.3, 0.7),…,(0.9, 0.1) and (1, 0). In each execution, a single solution was obtained. We selected the non-dominated solutions from the eleven obtained solutions.

In each execution of the algorithms NSGA2-IG, NSGA-II and GRASP, we use the same stopping criterion which is based on an amount of CPU time. This time was fixed in $110 \times n \times m$ milliseconds.

4.1 Benchmark Instances for the Problem

The instances of the problem were generated randomly according to Liefooghe et al. [16]. We considered test problems with number of jobs $n \in \{30, 50, 80, 100, 200\}$ and number of machines (at first stage) $m \in \{5, 10, 20\}$. The processing times are generated randomly, according to a uniform distribution: t_{ij}, tt_j, $ta_j \in U[0, 99]$. The setup times are uniformly distributed in the range [0–49]. The due dates for every jobs were generated randomly, with uniform distribution, over the interval $[3 \times p, (n+2) \times p]$, where p is the average value of previously generated processing times. We use the factor 3 because the 3sAFS problem can be considered as a flowshop with three machines. Thus, a due date d_i roughly lies between the average completion date of the first scheduled job and the average completion date of the last scheduled job [16]. For each configuration $n \times m$, 10 instances were generated. Therefore, a total of 150 instances were tested.

4.2 Performance Measures

In this work, to assess the quality of the non-dominated solutions attained by the three algorithms GRASP, NSGA-II and NSGA2-IG, two multi-objective performance measures are used: distance metric and hypervolume indicator.

We denoted by D_1, D_2 and D_3 the sets of non-dominated solutions (approximated Pareto fronts) obtained by the algorithms GRASP, NSGA-II and NSGA2-IG, respectively. We measure the quality of each set D_i (i =1,2,3) with relation to the reference set (*Ref*) constituted by gathering all non-dominated solutions obtained by the three algorithms, that is, *Ref* is the set of approximated non-dominated solutions obtained from ($D_1 \cup D_2 \cup D_3$). The *Ref* set is the best known Pareto-optimal front.

The used performance measures are defined as follow:

Average distance: measures the proximity between the solutions $s' \in D_i$ and the solutions $s \in Ref$. It also measures the solutions spreading on set D_i [14]. This metric is defined as follows:

$$d_{av}(D_i) = 100 \times \frac{1}{|Ref|} \sum_{s \in Ref} \min_{s' \in D_i} d(s, s')$$

$$\text{where, } d(s, s') = \max\left\{ \frac{f_1(s) - f_1(s')}{\Delta_1}, \frac{f_2(s) - f_2(s')}{\Delta_2} \right\},$$

and Δ_i is the difference between the biggest and smallest value of the objective f_i, considering the solutions of set *Ref*. Smaller values of $d_{av}(D_i)$ correspond to higher quality of the solutions in D_i.

Hypervolume indicator: measures the covered or dominated area by set D_i. In this paper, this metric is defined as follows:

$$H^*(D_i) = 100 \times \frac{H(Ref) - H(D_i)}{H(Ref)}$$

where, $H(X)$ is the hypervolume (area) of the solution space dominated by the solutions of the set X, i.e. the portion of the objective space that is dominated by X. Note that, we considered the relative percentage deviation of $H(D_i)$ with relation to $H(Ref)$. Smaller values of $H^*(D_i)$ correspond to higher quality of the solutions in D_i. The hypervolume indicator was introduced by Zitzler and Thiele [27].

4.3 Parameters of the Algorithms

The parameters of the algorithms NSGA-II and NGSA2-IG were experimentally fine-tuned. The algorithm GRASP was implemented following the original paper [7].

The best results of the standard NGSA-II were achieved using the following parameters: Population size N = 100. Crossover operator: TPX and SB2OX (to generate a child solution, a crossover operator is chosen randomly). Crossover probability: 100%. Mutation probability 30%.

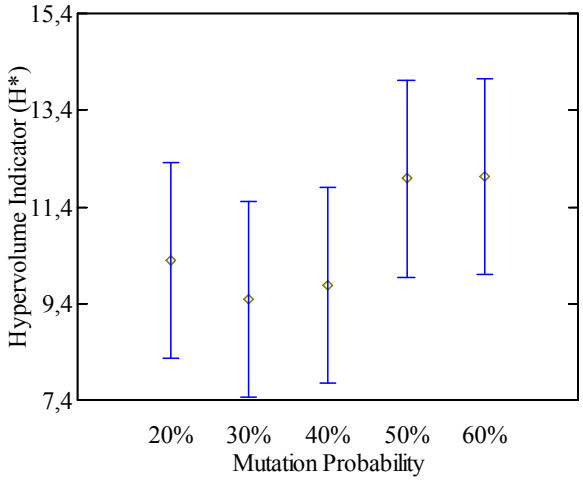

Figure 5. Means plot and Tukeys HSD intervals with 95% confidence level - Calibration of the mutation probability of the NSGA-II.

Figure 5 shows the performance of the NSGA-II algorithm (regarding the hypervolume indicator) for different mutation probabilities (20%, 30%, 40% and 50%). We performed the non-parametric Kruskal-Wallis Test [18] to validate the obtained results. The statistical analysis indicates that the obtained results are not significantly different for the five mutation probability values. However, we can see that the lowest average hypervolume was obtained for mutation probability 30%.

The NSGA2-IG uses the same parameters of NSGA-II. For the parameter d used the Iterated Greedy intensification, we tested different values. The best results were achieved using d =4 (for instances with $n \leq 100$) and $d = 2$ (for instances with $n = 200$).

4.4 Obtained Results

The non-dominated solutions obtained by the proposed hybrid algorithm NSGA2-IG are compared with the non-dominated solutions obtained by the other tested algorithms (GRASP and NSGA-II). To solve each test problem, the algorithms were run 10 times for all the 150 instances of the 3sAFS problem. The sets D_1, D_2 and D_3 contain the non-dominated solutions found among all the runs of the algorithms GRASP, NSGA-II and NSGA2-IG, respectively. For each instance, the reference set *Ref* is non-dominated solutions obtained from ($D_1 \cup D_2 \cup D_3$).

We note that, for all the 150 instances tested, the algorithms NSGA2-IG and GRASP determine 94.66% and 5.49% of the reference solutions, respectively. The NSGA-II algorithm does not determine any reference solution.

The distance and hypervolume measures are calculated for each set D_i. In Table 1, results attained by the algorithms in relation to the average distance metric are presented. In this Table, the results (average values) are grouped according to the number of jobs and number of machines ($n \times m$) of the instances.

We can see that, for all groups of instances the NSGA2-IG algorithm is the one that produces lower average distances, that is, closer to zero. Then the proposed algorithm is notoriously better than the standard NSGA-II and GRASP. We can see also that the algorithm GRASP performs better than NSGA-II.

The distance metric measures the proximity between the solutions of a set D_i and the solutions of set *Ref*. Therefore, the higher the percentage of solutions of D_i in the *Ref* set, the lower tends to be the values of the distance metrics. The values of the distance metrics tend to be smaller, but those values also depend of the distance between D_i solutions and solutions belonging to *Ref* set obtained by other algorithms.

Table 2 presents the comparison of the algorithms regarding the hypervolume indicator. In this Table we can see again that NSGA2-IG algorithm presents lower average values of the relative percentage deviation of the hypervolume (H^*) for all group of instances.

4.5 Statistical Analysis

In order to validate the obtained results, we apply again a statistical analysis using the average distance (d_{av}) and hypervolume (H^*) measures as response variables.

First, to verify if the observed differences between the obtained results are statistically significant, we performed an analysis of variance (ANOVA) [18]. The three main assumptions of ANOVA were checked: normality, homoscedasticity and independence. Since the normality test was not satisfied, we performed the non-parametric Kruskal-Wallis Test. This test compares between the medians of the three algorithms to determine if there is a significant difference. The Kruskal-Wallis results (not shown in detail due to reasons of space) indicate that there is statistically significant difference between the obtained results at a 95% confidence level. It was observed that the computed P-value is less than 0.05 which shows that exist significant difference between the algorithms. The Kruskal Wallis Test does not specify which algorithms are different. So, we use a non-parametric Multiple Comparisons test to compare each pair of means with a 95% confidence level.

Table 1. Average distance (d_{av}) results.

n	*m*	GRASP	NSGA-II	NSGA2_IG
30	**5**	7.21	23.52	1.05
30	**10**	6.92	26.29	0,67
30	**20**	8.52	25.53	0.61
50	**5**	13.48	27.65	0.50
50	**10**	13.11	24.82	0.26
50	**20**	16.98	31.16	0.21
80	**5**	14.41	24.64	0.55
80	**10**	20.36	32.21	0.27
80	**20**	19.51	36.34	0.09
100	**5**	16.93	25.81	0.25
100	**10**	14.80	29.30	0.19
100	**20**	22.82	40.17	0.22
200	**5**	10.77	19.45	0.31
200	**10**	13.59	22.18	0.16
200	**20**	17.92	25.26	0.06
	Average	**14.49**	**27.62**	**0.36**

Table 2. Hypervolume indicator (H^*) results.

n	*m*	GRASP	NSGA-II	NSGA2_IG
30	**5**	13.75	37.68	3.66
30	**10**	15.92	36.15	3.74
30	**20**	16.63	34.48	3.81
50	**5**	23.25	43.07	3.82
50	**10**	18.28	39.27	3.25
50	**20**	22.46	36.88	3.28
80	**5**	19.34	39.68	2.89
80	**10**	28.00	43.45	2.70
80	**20**	23.12	40.88	2.59
100	**5**	22.90	41.37	2.77
100	**10**	19.71	40.10	2.06
100	**20**	23.55	41.34	2.03
200	**5**	15.29	38.62	2.03
200	**10**	19.60	34.96	2.17
200	**20**	23.24	29.03	2.18
	Average	**20.34**	**38.46**	**2.87**

The Tables 3 and 4 show the result of the Multiple Comparisons test regarding the average distance and hypervolume indicator metrics, respectively. The first column of these Tables shows the pairs of algorithms being compared. The "Difference" column displays the sample mean of the first algorithm minus that of the second. The "+/- Limits" column shows an uncertainty interval for

the difference. Any pair of algorithms for which the absolute value of the difference exceeds the limit is statistically significant at the selected confidence level 95% and is indicated by an asterisk (*) in the "Significant" column. In Tables 3 and 4 we can see that there are significant differences between all the pairs of algorithms. The same statistical analysis can be displayed in Figures 6 and 7.

Table 3. Multiple Comparisons test - Average Distance.

Contrast	*Significant*	Difference	+/- Limits
GRASP ↔ NSGA-II	*	-13.1333	3.07722
GRASP ↔ NSGA2_IG	*	14.1287	3.07722
NSGA-II ↔ NSGA2_IG	*	27.262	3.07722

Table 4. Multiple Comparisons test - Hypervolume Indicator.

Contrast	*Significant*	Difference	+/- Limits
GRASP ↔ NSGA-II	*	-18.1287	1.46216
GRASP ↔ NSGA2_IG	*	18.6296	1.46216
NSGA-II ↔ NSGA2_IG	*	36.7583	1.46216

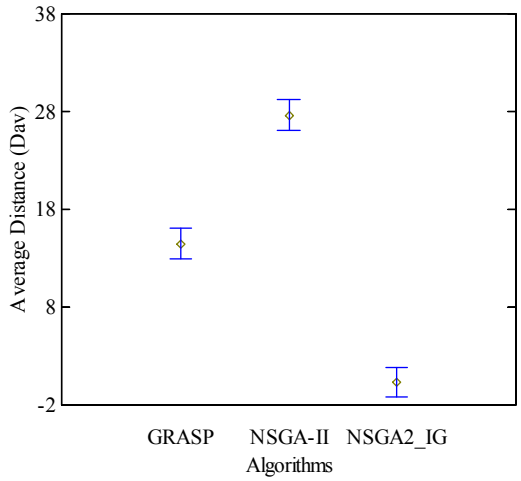

Figure 6. Means plot and Tukeys HSD intervals with 95% confidence level - Average Distance metric (d_{av}).

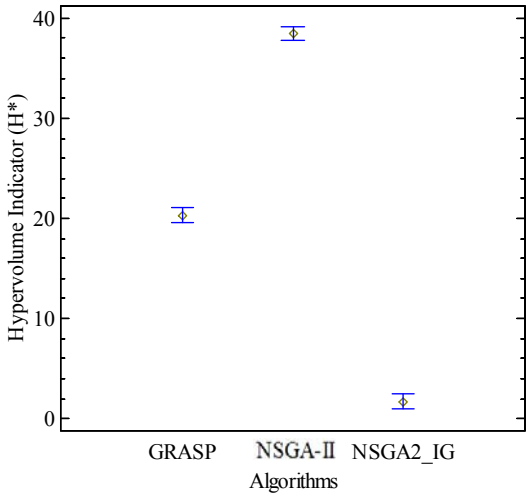

Figure 7. Means plot and Tukeys HSD intervals with 95% confidence level - Hypervolume Indicator (*H).**

Figures 6 and 7 show the means plot and Tukey's Honestly Signifiant Difference (HSD) intervals at 95% confidence level from the Multiple Comparisons test for two performance measures, respectively. Since the confidence interval of NSGA2-IG algorithm does not overlap any of the other intervals, the mean of NSGA2-IG is significantly different than that of the other two algorithms. So we can state that, on average, NSGA2-IG is better than NSGA-II and GRASP. We can see also that, on average, GRASP is better than NSGA-II.

The statistical analysis shows that the use of IG improves significantly the results of the basic NSGA-II algorithm.

5. CONCLUSIONS

This paper has proposed a hybrid algorithm for a bi-objective three-stage Assembly Flowshop Scheduling problem with sequence-dependent setup times. In order to find an approximation of the Pareto optimal set, the hybrid algorithm (NSGA2-IG) combines the basic NSGA-II [8] scheme with an intensification strategy based on Iterated Greedy.

We have performed a comparative study between the proposed algorithm and two other algorithms considering medium and large instances of the problem. According to the obtained results and the statistical analyses, the proposed NSGA2-IG algorithm performed better than the two algorithms, NSGA-II and GRASP. The results of NSGA-II were improved significantly by using IG intensification. After the computational experiments we can conclude that the proposed hybrid algorithm shows an excellent performance overcoming the original NSGA-II.

6. ACKNOWLEDGMENTS

This work was supported by Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG), Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES).

7. REFERENCES

[1] Al-Anzi, F. S., and Allahverdi, A. 2006. A hybrid tabu search heuristic for the two-stage assembly scheduling problem. *International Journal of Operations Research*, 3, 2 (2006), 109-119.

[2] Al-Anzi, F. S., and Allahverdi, A. 2007. A self-adaptive differential evolution heuristic for two-stage assembly scheduling problem to minimize maximum lateness with setup times. *European Journal of Operational Research*, 182, 1 (oct. 2007) 80-94.

[3] Allahverdi, A., Ng, C. T., Cheng, T. E., and Kovalyov, M. Y. 2008. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187, 3 (2008), 985-1032.

[4] Andrés, C., and Hatami, S. 2011. The three stage assembly permutation flowshop scheduling problem. In *V international conference on industrial engineering and industrial management.* (Colombia, Cartagena, September 7-9, 2011). 867-875.

[5] Arroyo, J. E. C., and Armentano, V. A. 2004. A partial enumeration heuristic for multi-objective flowshop scheduling problems. *Journal of the Operational Research Society*, 55, 9 (sep. 2004), 1000-1007.

[6] Arroyo, J. E. C., Ottoni, R. S. and Oliveira, A. P. 2011. Multi-objective Variable Neighborhood Search Algorithms

for a Single Machine Scheduling Problem with Distinct due Windows. *Electronic Notes in Theoretical Computer Science*, 281 (Dec. 2011), 5-19.

[7] Campos, S. C., Arroyo, J. E. C., and Gonçalves, L. B. 2013. Uma heuristica grasp-vnd para o problema de sequenciamento de tarefas num ambiente assembly flowshop com três estágios e tempos de setup dependentes da sequência. In *Proceedings of the XLV Brazilian Symposium of Operational Research.* (Natal-RN, Brazil, Setember 16 - 19, 2013). 2147-2158.

[8] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. A. M. T. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation*, IEEE Transactions on, 6, 2 (Apr. 2002), 182-197.

[9] Deb, K., Steuer, R. E., Tewari, R., and Tewari, R. 2011. Bi-objective portfolio optimization using a customized hybrid NSGA-II procedure. In *Proceedings of the 6th international conference on Evolutionary Multi-criterion Optimization, EMO 2011.* (Ouro Preto-MG, Brazil, April 5-8, 2011). Springer Berlin Heidelberg, 358-373.

[10] Garey M.R., Johnson D.S., Sethi R. 1976. The complexity of flowshop and job shop scheduling. Mathematics and Operations Research, 1, 2 (May. 1976), 117–29.

[11] Guo, Y., Chen, Z. R., Ruan, Y. L., and Zhang, J. 2012. Application of NSGA-II with local search to multi-dock cross-docking sheduling problem. In *2012 IEEE International Conference on Systems, Man, and Cybernetics, SMC.* (Seoul, Korea, October 14-17, 2012). IEEE, 779-784.

[12] Hatami, S., Ebrahimnejad, S., Tavakkoli-Moghaddam, R., and Maboudian, Y. 2010. Two meta-heuristics for three-stage assembly flowshop scheduling with sequence-dependent setup times. *The International Journal of Advanced Manufacturing Technology*, 50, 9-12 (oct. 2010), 1153-1164.

[13] Ishibuchi, H., Yoshida, T., and Murata, T. 2003. Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation*, 7, 2, (Apr. 2003), 204-223.

[14] Knowles, J., and Corne, D. 2002. On metrics for comparing nondominated sets. In *Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02.* (Honolulu, HI, May 12-17, 2002). IEEE, 1, 711-716.

[15] Koulamas, C., and J Kyparisis, G. 2001. The three-stage assembly flowshop scheduling problem. *Computers & Operations Research*, 28, 7 (jun. 2001), 689-704.

[16] Liefooghe, A., Basseur, M., Humeau, J., Jourdan, L., and Talbi, E. G. 2012. On optimizing a bi-objective flowshop scheduling problem in an uncertain environment. *Computers & Mathematics with Applications*, 64, 12 (dec. 2012), 3747-3762.

[17] Maleki-Darounkolaei, A., Modiri, M., Tavakkoli-Moghaddam, R., and Seyyedi, I. 2012. A three-stage assembly flow shop scheduling problem with blocking and sequence-dependent set up times. *Journal of Industrial Engineering International*, 8, 8 (oct. 2012), 1-7.

[18] Montgomery, D. C., and Montgomery, D. C. 1997. *Design and analysis of experiments* (Vol. 7). New York: Wiley.

[19] Minella, G., Ruiz, R., and Ciavotta, M. 2011. Restarted Iterated Pareto Greedy algorithm for multi-objective flowshop scheduling problems. *Computers & Operations Research*, 38, 11 (Nov. 2011), 1521-1533.

[20] Nawaz, M., Enscore, E. E., and Ham, I. 1983. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, 11, 1 (1983), 91-95.

[21] Pan, Q. K., and Ruiz, R. 2014. An effective iterated greedy algorithm for the mixed no-idle permutation flowshop scheduling problem. *Omega*, 44, (Apr. 2014), 41-50.

[22] Pires, D. F., Antunes, C. H., and Martins, A. G. 2012. NSGA-II with local search for a multi-objective reactive power compensation problem. *International Journal of Electrical Power & Energy Systems,* 43, 1 (dec. 2012), 313-324.

[23] Potts, C. N., Sevast'Janov, S. V., Strusevich, V. A., Van Wassenhove, L. N., and Zwaneveld, C. M. 1995. The two-stage assembly scheduling problem: complexity and approximation. *Operations Research*, 43, 2 (Apr. 1995), 346-355.

[24] Ruiz, R., Maroto, C., and Alcaraz, J. 2005. Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. *European Journal of Operational Research*, 165, 1 (Aug. 2005) 34-54.

[25] Ruiz, R., and Stützle, T. 2008. An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives*. European Journal of Operational Research*, 187, 3 (jun. 2008), 1143-1159.

[26] Tajbakhsh, Z., Fattahi, P., and Behnamian, J. 2013. Multi-objective assembly permutation flow shop scheduling problem: a mathematical model and a meta-heuristic algorithm. *Journal of the Operational Research Society*. (set. 2013).

[27] Zitzler, E., and Thiele, L. 1998. Multiobjective optimization using evolutionary algorithms – a comparative case study. In *Proceedings of the 5th International Conference Amsterdam on Parallel problem solving from nature (PPSN V).* (Amsterdam, The Netherlands, September 27–30, 1998). Springer Berlin Heidelberg, 1498, 292-301.