A Modified Harmony Search Algorithm for multi-objective flowshop scheduling problem with due dates

Marco Frosolini, Marcello Braglia, Francesco Aldo Zammori

A Modified Harmony Search Algorithm for multi-objective flowshop scheduling problem with due dates

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2009-IJPR-0435.R4
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	03-Sep-2010
Complete List of Authors:	Frosolini, Marco; Pisa University, Mechanical, Nuclear and Production Engineering Braglia, Marcello; Pisa University, Mechanical, Nuclear and Production Engineering Zammori, Francesco; Pisa University, Mechanical, Nuclear and Production Engineering
Keywords:	FLOW SHOP SCHEDULING, GENETIC ALGORITHMS
Keywords (user):	Multi-criteria methodologies, Harmony Search Optimization

A Modified Harmony Search Algorithm for multi-objective flowshop scheduling problem with due dates

M. FROSOLINI, M. BRAGLIA and F. A. ZAMMORI

Dipartimento di Ingegneria Meccanica, Nucleare e della Produzione

Facoltà di Ingegneria, Università di Pisa, Via Bonanno Pisano, 25/B, 56126 Pisa, Italy

BRAGLIA Marcello
Dipartimento di Ingegneria Meccanica, Nucleare e della Produzione
Facoltà di Ingegneria, Università degli Studi di Pisa
Via Bonanno Pisano 25/B, 56126 Pisa
Phone: +39 050 913029
Fax: +39 050 913040
E-mail: m.braglia@ing.unipi.it

ZAMMORI Francesco
Dipartimento di Ingegneria Meccanica, Nucleare e della Produzione
Facoltà di Ingegneria, Università degli Studi di Pisa
Via Bonanno Pisano 25/B, 56126 Pisa
Phone: +39 050 913039
Fax: +39 050 913040
E-mail: francesco.zammori@ing.unipi.it

Corresponding author

FROSOLINI Marco
Dipartimento di Ingegneria Meccanica, Nucleare e della Produzione
Facoltà di Ingegneria, Università degli Studi di Pisa
Via Bonanno Pisano 25/B, 56126 Pisa
Phone: +39 050 913039
Fax: +39 050 913040
E-mail: m.frosolini@ing.unipi.it

Abstract words: 160
Article words: 8700

Abstract

This paper presents a *Modified Harmony Search Optimization Algorithm* (MHSO), specifically designed to solve two and three-objectives permutation flowshop scheduling problems, with due dates. To assess its capability, five sets of scheduling problems have been used to compare the MHSO with a known and highly efficient Genetic Algorithm (GA) chosen as benchmark. Obtained results show that the new procedure is successful in exploring large regions of the solution space and in finding a significant number of Pareto non-dominated solutions. For those cases where the exhaustive evaluation of sequences can be applied the algorithm is able to find the whole non-dominated Pareto border, along with a considerable number of solutions that share the same optimal values for the considered optimization parameters.
To validate the algorithm, five sets of scheduling problems are investigated in depth in comparison with the GA. Results obtained by both methods (exhaustive solutions have been provided as well for small sized problems) are fully described and discussed.

Keywords: Permutation flowshop scheduling; Multi-criteria methodologies; Genetic Algorithms; Harmony Search Optimization.

1. Introduction

Flowshop scheduling has attracted many researchers over time since it was firstly proposed by Johnson in 1954 (see, for example, Dannenbring, 1977, Lageweg *et al.*, 1978, Potts, 1980, Osman and Potts, 1989, Pinedo, 1995, Nowicki and Smutnicki, 1996, Carlier and Rebai, 1996, Cheng *et al.*, 1997, Haouari and Ladhari, 2003, Srikanth and Barkha, 2004, Ladhari and Haouari, 2005, Tseng and Lin, 2009) and has been extensively investigated by researchers both with single (refer, for instance, to Campbell, 1970, Ignall and Schrage, 1965, Nawaz *et al.*, 1983) and multi-objective techniques (an extensive review is given in T'Kindt and Billaut, 2001). As it notoriously represents a computationally *NP-hard* problem and exhaustive evaluation applies only to undersized cases (*i.e.*, when the number of job is reasonably small), most of the proposed approaches have focused on the use of optimization or on the adoption of metaheuristic techniques for single objective flowshop scheduling problems (Pan *et al.*, 1997, Stevens *et al.*,1997, Cheng *et al.*, 2001, Grabowski and Wodecki, 2004, Rajendran and Ziegler, 2005, Tasgetiren *et al.*, 2007, Rajkumar *et al.*, 2009, Lin *et al.*, 2009). Literature on the subject is extensive and a comprehensive survey of makespan minimization from early works up to recent approaches of metaheuristics is provided by Hejazi *et al.* (2005), whereas a broad review of the evolution of the available methods over time is presented in Gupta and Stafford (2006). It is worth noting that the use of multiple criteria enables a more practical solution for the

decision makers (T'Kindt and Billaut, 2001). Indeed, single criterion optimizations do not consider the important trade-offs that intrinsically characterize the scheduling problem. Conversely, whether multiple objectives are optimized properly and conjointly, the solution narrows down to a limited subset of optimal/efficient feasible schedules among which the analyst may choose after appropriate considerations on the criteria themselves. Schaffer (1985) firstly introduced multi-criteria genetic algorithms (MCGA) and, since then, some modified MCGA procedures have been proposed to improve both the quality of the solutions and the speed of the search algorithms (Murata *et al.*, 1996, Ponnambalam *et al.*, 2004). Other methods have been applied as well. For example, Gangadharan *et al.* (1994) proposed a Simulated Annealing algorithm for two-criteria scheduling problems. Rajendran (1994) approached the problem of scheduling in flowshop and flowline-based manufacturing cell with the bicriteria of minimizing makespan and total flowtime of jobs. Cao *et al.* (2003) worked on production scheduling problems in manufacturing systems with parallel machine flowshops. The authors developed a mathematical programming model for combined part assignment and job scheduling. The objective was to minimize a weighted sum of production cost and the cost incurred from late product delivery Interesting efforts have been made to adapt Multi-Objective Immune Algorithms (MOIA) to scheduling problems (Tavakkoli-Moghaddam *et al.*, 2007). Khan *et al.* (2007) addressed the problem of minimizing the weighted sum of makespan and maximum tardiness in an *m*-machine flow shop environment using a metaheuristic called Greedy Randomized Adaptive Search Procedure (GRASP). Yandra *et al.* (2007) discussed the application of a genetic algorithm featuring heterogeneous population to solve multi-objective flowshop scheduling problems. Braglia *et al.* (2009) presented a new heuristic for solving the flowshop scheduling problem that aims to minimize makespan and maximum tardiness. In this case, the Technique For Order Preference By Similarity of Ideal Solution (TOPSIS) algorithm is integrated with the Nawaz-Enscore-Ham (NEH) heuristic to generate a set of potential scheduling solutions. All the above mentioned techniques give interesting results, being able to evaluate a significant number of optimal or near-optimal solutions with reasonable computing efforts.

Recently, the Harmony Search Optimization (HSO), an algorithm that mimics the music composition processes to explore the space of the feasible solutions, has been introduced by Geem *et al.* (2001). Although HSO has shown to be effective in most engineering optimization problems (Lee *et al.*, 2004, Geem *et al.*, 2005, Geem, 2006, Kim *et al.*, 2001,Kim *et al.*, 2006), it has not been applied yet to scheduling issues. Owing to this, the present work presents a modified Harmony Search Optimization Algorithm (MHSO) addressed to solve two and three-objective permutation flowshop scheduling problems with due dates. Compared with a known and highly efficient GA (Murata *et al.*, 1996), the new algorithm shows to be successful in exploring large regions of the solution space and in finding a large amount of feasible efficient solutions. For those cases where the exhaustive evaluation of sequences can be applied the algorithm is also able to find the whole non-dominated Pareto frontier in a large number of instances having the same initial data, along with a considerable number of solutions that share the same efficient values for the considered optimization parameters (in the following these will be called "equipollent" solutions).

The paper is organized as follows. Firstly, the multi-objective permutation flowshop scheduling problem is briefly presented and discussed, along with the parameters that have been considered for optimization. Following, the standard HSO and the proposed MSHO are described in detail. Finally five sets of scheduling problems are investigated in comparison with the GA proposed by Ishibuchi and Murata (1998), being the latter one of the most effective algorithms proposed in literature to cope with the subject. Results obtained with both methods (exhaustive solutions have been provided as well for small sized problems) are fully described and discussed.

2. The multi-objective permutation flowshop scheduling problem with due dates

Multi-objective scheduling enables a more practical solution for the decision makers as it allows to consider at one time and in a suitable manner a broader range of decision criteria. However, due to the fact that a solution that optimizes all the considered criteria often does not exist, a new definition of optimality must be introduced. Here, in particular, we refer to the Pareto optimality as defined by T'Kindt and Billaut (2001). In brief, multi-objective flowshop scheduling algorithms (MOFSA) aim to find (all or most of) the sequences that minimize (or maximize, if the case) the whole set of decision criteria or, in other words, the Pareto non-dominated frontier of the solution space. Mathematically speaking, if $f_j(s_i)$ is the fitness function for the single *j-th* criterion to be evaluated for a generic sequence s_i, the problem can be expressed as:

$$min\{f_1(s_i), f_2(s_i), \dots, f_p(s_i)\} \quad (1)$$

where p is the global number of decision criteria. The sequence s_i is said to dominate another solution s_j if the following applies:

$$\forall p: f_p(s_i) \leq f_p(s_j) \text{ and } \exists h: f_h(s_i) < f_h(s_j) \quad (2)$$

Figure 1 shows dominated and non-dominated solutions for a two-objective scheduling problem.

FIGURE 1 HERE

The standard permutation flowshop scheduling problem with due dates, in particular, can be formally enunciated as follows: a set of n jobs $\{J_1, J_2, \dots, J_n\}$ has to be processed in sequence on m machines $\{M_1, M_2, \dots, M_m\}$. Each job consists of m operations $(O_1, O_2, \dots, O_m)$ and the *j-th* operation of each job must be processed on the *j-th* machine in the sequence. Each operation of a job is characterized by a processing time on each machine, leading to the definition of a n x m Processing Time Matrix (PTM) and by a due date (defining the n-sized Due Dates Vector, DDV).

The optimization criteria that have been used in the present study, in close accordance with Ishibuchi and Murata (1998), are the *makespan* (MS) and the *maximum tardiness* (MT) in the case of two-objective scheduling, being the *total flowtime* (TFT) the last criterion in the case of three-objective problems.

3. Harmony Search Optimization

Recently, *Harmony Search Optimization* (HSO) has been proposed as an algorithm that mimics music composing (Geem *et al.*, 2001). Briefly, it tries to replicate the process that leads musicians to continuously adjust pitches and tunes to produce better harmonies and is based on the concept of an *Harmony Memory* (HM) that represents the acquired know-how of the music player. The creative process is made up of three different stages, during which the musician:

1) draws from the HM to make the most of the past experiences;
2) tries to modify the known melodies by adjusting pitches and changing notes in order to get new compositions;
3) improvises and creates new harmonies from scratch, following inspiration.

If the new melody sounds good (with respect to an aesthetic benchmark) it is inserted within the HM to increase and enhance the available repertoire. In mathematical terms this can be obtained by representing each harmony as a vector made up of exactly n elements, if n is the number of the variables that define the model. Each element within the vector corresponds to a note. The aesthetic benchmark is replaced by an appropriate fitness function (or more, in case of multi criteria optimization) and the problem can be easily brought back to the classical form reported in equation 1 (paragraph 2). Once the *Harmony Memory Size* (HMS) has been defined, the whole HM can be written as a matrix of HMS vectors:

$$\begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_n^1 & | & f(x^1) \\ x_1^2 & x_2^2 & \cdots & x_n^2 & | & f(x^2) \\ \vdots & \cdots & \cdots & \cdots & | & \vdots \\ x_1^{HMS} & x_2^{HMS} & \cdots & x_n^{HMS} & | & f(x^{HMS}) \end{bmatrix}$$

The value of the fitness function(s) for each harmony is (are) included within the HM.

The algorithm (Figure 2) starts by generating HMS random new harmonies. In general, each note (variable) should be chosen within its definition set (valid range) and considering the other notes that have already been inserted in order to avoid violated constraints. This issue can be easily overcome both by checking and adjusting the validity of the vector or by imposing penalties to the violating sequences. At each iteration a new harmony is generated in n successive steps (starting form position 1 to position n, where n is the number of variables) by means of three mechanisms: (*i*) random selection, (*ii*) memory considerations and (*iii*) pitch adjusting.

If the *Harmony Memory Considering Rate* (HMCR) is the rate representing the probability of choosing a note from the memory, the first method involves the random selection of a note from the valid range with a probability of 1 – HMCR. Therefore, a random number is extracted and, if its value is greater than HMCR and less than 1 a note is selected within the definition set:

$$x_i^* \leftarrow x \in X$$

where x_i represents the variable at the current step and X is the above mentioned definition set. On the contrary, if the random value is less than HMCR a vector is randomly selected from the HM and the note corresponding to the current step is introduced within the new harmony:

$$x_i^* \leftarrow x \in \{x_i^1, x_i^2, \dots, x_i^{HMS}\}$$

If the constraints are somehow violated it is necessary to check the validity or to introduce penalties. Finally, as the musician modifies some notes to adjust their pitches to the neighboring ones, with the aim of obtaining a smooth and sound composition, so the algorithm requires to alter some variables with a certain probability (this is also referred to as *Pitch Adjusting Rate*, PAR). In practice, a note is randomly selected and its value is changed within the allowed range.

If the new harmony is better than the worst one contained in the HM, with the respect to the fitness function (or referring to the Pareto optimality in the case of multiple objectives), the latter is overwritten to improve the knowledge base. The process is iterated up to the maximum number of iterations or until a particular terminating condition is satisfied.

FIGURE 2 HERE

The algorithm, modified and improved in several manners, has been successfully applied to some interesting engineering problems, such as structural design (Lee and Geem, 2004), water network design (Geem, 2006), traffic routing (Geem *et al.*, 2005) and fluid leakage detection (Kim *et al.*, 2001).

4. A New Harmony Search for multi-objective permutation flowshop scheduling

Since the HSO proved to be extremely performing while coping with some noteworthy engineering problems, it appears to be a good candidate to cope with the multi-objective standard permutation flowshop scheduling problem. The aim of the present work is that of showing its capabilities (both with respect to performances and quality of the solutions) and this is done by comparison with a well known and highly efficient GA. In particular, the mentioned GA is known in literature since it provides a large number of optimal solutions within short lapses of time.

To enhance the original HSO algorithm capabilities of exploring larger regions of the Pareto border while keeping a good computational speed some modifications have been introduced within the procedure. To begin with, the intrinsic structure of the algorithm remains almost unchanged: notes/variables represent jobs and a whole harmony is a sequence. Since each job can be introduced into a sequence only once, a strict check is carried out on all new harmonies and duplication of jobs within a sequence is not allowed. The MHSO continues to refer to the three classical mechanisms used to generate new harmonies (random selection, memory considerations and pitch adjusting). With respect to the quality of new generated harmonies, the multi-objective algorithm makes use of the known Pareto optimality definition with respect to the *makespan*, the *maximum tardiness* and, in the case of three-objective scheduling, the *total flowtime* criteria. Further, the algorithm introduces new concepts that help finding better solutions and moves a step forward in the direction of emulating the human behavior. A musician who improvises a new composition is influenced by early works (his/her own and those of other composers as well, that constitute the personal knowledge base) and usually tends to make the most of this heritage. The original HSO mimics this behavior by replicating a single note at time from a known harmony to the new one, but it lacks of the capability of introducing and preserving larger portions of a composition. It is worth noting that this aspect represents a notable advantage of genetic algorithms and, therefore, a similar process, named *Large Portion Recovery* (LPR) has been introduced within the HSO, along with the corresponding probability (*Large Portion Recovery Ratio*, LPRR). Practically speaking, this is obtained by performing, from time to time and in accordance to the LPRR, that is usually kept very small, a crossover and a mutation procedure on the best harmonies available within the HM. It is also known that most musicians are able to improvise new compositions for a period, after which they tend to re-use their early works, to take inspiration from the other composers, as creativity inevitably decreases over time, or simply to explore new and different musical genres. Sometimes this determines the flourishing of new inspiration and the overall improvement of their compositions. This aspect strictly resembles what occurs when the HSO, after a generally high number of iterations, becomes unable to explore new sections of the solution space and repeatedly visits the same sequences without further improvements. A solution, namely *Harmony Recovery* (HR), to this particular issue has been proposed and evaluated, based on a large scale LPR procedure. After a given number of iterations, that can be defined and controlled by means of an *Harmony Recovery Ratio* (HRR), the algorithm increases the recovery mechanism probability (acting on the LPRR) and the corresponding crossover activity. Doing so, the algorithms starts to take advantage of the most effective portions of the sequences that have been previously found. This step is preceded by a random regeneration of the HM (up to the 80% of the harmony memory can be updated, but generally it is suitable to update 50% or less of it), aimed to reduce the convergence of a large number of harmonies to a limited subset of the solutions space. To increase the capability of the algorithm of escaping from such unlikable situations, a random regeneration of the HM has also been forced whether the non-dominated memory (the so-called “elite”), that will be introduced and explained later, has not been updated within a given timeframe (usually a given percentage of the maximum number of iterations).

Consequently, the whole algorithm can be summarized as follows:

1) the HM is randomly generated and the fitness functions are evaluated for each criterion and for each vector (as in the original HSO);
2) the iterative procedure starts and lasts until the terminating conditions, to be defined for the specific problem under study, are met or the maximum number of iterations is reached. In the present work both the GA and the MHSO are stopped when the maximum number of iterations is reached. Since the aim of the study is to compare the two algorithms this allows to evaluate both speed and the relative quality of the solutions;
3) for each iteration the following steps apply:
    a. if the current iteration number is less than a given amount, corresponding to the value of the HRR multiplied for the *Maximum Number of Iterations* (MNI) the HSO process is performed as usual (experience shows that good values for HRR are comprised between 0.6 and 0.8):
        i. a random number is extracted. If it is less than the LPRR (this value is kept very small, such as 0.001) then the LPR procedure is started and two harmonies from the HM are subjected to a two-points crossover operator to generate a new harmony. Indeed, harmonies are mere sequences and can be treated as chromosomes in a generic GA. The classical two-point crossover is here used to generate two new harmonies from two efficient parents: briefly, two random positions are selected within the parents and the corresponding chromosomes sections are plainly exchanged. The new harmony may also be subjected, in accordance to the PAR, to a pitch adjusting procedure as in the ordinary HSO. It is worth noting that this has been reduced to a mutation operator (both insertion and exchange of the elements of a sequence are randomly used), due to the fact that each job cannot be present more than once in each sequence. Also, the PAR may be optionally set linearly increasing between a minimum and a maximum values;
        ii. if the number is greater than LPRR the standard HSO procedure is performed. The worst - or better, an inefficient - harmony (the one that will be replaced by a new better solution) is picked on the basis of a roulette wheel selection process (Goldberg, 1989) rather than on the simple evaluation of the fitness function as in the original HSO. The procedure is based on the fact that the probability of each harmony of being selected and replaced is evaluated on the basis of a weighted fitness function of all criteria (with the same weights proposed by Ishibuchi and Murata (1998), that showed to be very effective). This is obviously greater for inefficient solutions. While on one hand the risk of removing good harmonies is kept very low due to a proper elitism procedure, on the other the algorithm prevents the threat of early convergence. Furthermore, this allows a straight and easy evaluation of the selection probability. Also, a local search procedure is started for every new harmony: the MHSO looks for a limited number of neighbors (carrying out a simple mutation procedure each time, where both insertion and exchange of the elements of a sequence are randomly used)

and, if a better solution emerges (here the Pareto optimality is used to evaluate the quality of the harmony), the original new sequence is overwritten. The previously selected inefficient harmony is finally replaced;

b. on the other hand, if the current iteration number is greater than the value of the HRR multiplied for the Maximum Number of Iterations, the HR procedure is started:
   i. up to 80% of the current population can be randomly generated (and, necessarily, the old one overwritten), to simulate the fact that the musician begins to take inspiration from other composers or that he simply wants to explore new musical genres. This occurs when the above condition is met and allows the algorithm to move from a section of the Pareto border and to explore new portions of the solution space;
   ii. the LPRR is increased up to a maximum of 0.5 (experience shows that values between 0.01 and 0.1 give the best results), and the algorithm goes on as usual;

4) as briefly pinpointed before, at each iteration a proper elitism procedure is applied to the HM. The non-dominated solutions found at every step are kept in a separate memory location that is punctually updated (*i.e.*, solutions that become dominated when better ones are found are removed from the elite). In this case, to determine dominated and non-dominated harmonies, the algorithm refers to the Pareto optimality (as described by equations 1 and 2 of paragraph 2) rather than on the evaluation of the weighted fitness function of all criteria. To take advantage of this non-dominated memory, up to a maximum of 10% of the population size elite harmonies are inserted within the HM with the aim of improving the MHSO knowledge base, without forcing a premature convergence of the whole memory to a limited portion of the solution space (greater percentages, indeed, have shown to be particularly inefficient);
5) if the non-dominated elite has not been updated within 1/6 of the maximum number of iterations (elite update delay) then a random regeneration (up to 80%) of the HM is forced.

The complete MHSO procedure is hereafter summarized in the form of Pascal pseudo-code:

```
generate random Harmony Memory;
set LPRR to 0.0001
while terminating conditions are not met or maximum iterations are not reached do
  if (current iteration < HRR × MNI) or (LPRR = 0.5) then
    extract a random number → Rand
    if Rand < LPRR then
      perform Large Portion Recovery (LPR) procedure to get a new harmony
      extract a random number → Rand
      if Rand < PAR then
        perform Pitch Adjusting (mutation) procedure on the new harmony
      end if
    else
```

perform standard HSO procedure and get new harmony → NHarmony
select inefficient harmony (roulette wheel selection) → IHarmony
perform local search procedure on NHarmony (in case of improvement overwrite)
if NHarmony improves IHarmony replaces it
end if
else
regenerate up to a maximum of 80% of current population
set LPRR to a maximum of 0.5
end if
perform elitism procedure (based on Pareto optimality) and update non-dominated elite
insert harmonies from elite to current population (up to a maximum of 10% of population size)
if elite not updated within 1/6 of MNI then regenerate up to 80% of current population
end while

It is also schematically reported if Figure 3.

FIGURE 3 HERE

5. Comparative analysis

The MHSO presented in the paper has been compared with the Genetic Algorithm proposed by Ishibuchi and Murata (1998), that in the following will be indicated as Ishibuchi-Murata GA (IMGA). To this aim their formulation has been adopted with respect to:

- the number of jobs and machines;
- the generation of the initial population;
- the selection of the objectives;
- the setting of the operating parameters (that have been optimized as proposed by the authors themselves).

This choice is mainly due to the fact that the above mentioned algorithm is known to be particularly effective in the context of the specific issue under analysis and that it refers both to two and three criteria optimization problems. Moreover, it is also known that the performances of GAs are seldom exceeded or even achieved by other procedures. Hence, it certainly represents a valuable benchmark.

To begin with, the problems that have been examined in depth can be divided into two large sets, characterized by the number of optimization criteria (respectively 2 and 3) and each subdivided into five subsets with respect to the number of jobs (we assumed, respectively, n = 10, 20, 30, 50 and 100) and machines (we assumed, respectively, m = 5, 10, 15, 20 and 30). This choice is mainly due to the fact that the

IMGA has been tested by the authors (Ishibuchi and Murata, 1998) on similar problems (actually, they reported results for the two cases n = 10, m = 5 and n = 20, m = 10) and, therefore, this allows to perform a straightforward comparison. It is noteworthy that the IMGA has been tested with different sets of operating parameters (population size, crossover and mutation probabilities, elitism) with the aim of finding their best combinations before starting the comparative process. However, for smaller problems (n = 10, m = 5 and n = 20, m = 10) the values proposed by the above mentioned authors still showed to be the best ones. For bigger problems (n = 30, m = 15, n = 50, m = 20 and n = 100, m = 30) the best combinations were chosen after a significant number of tests.

The first set refers to a two-objectives optimization, where the adopted criteria are the *makespan* and the *maximum tardiness*, whereas the latter is aimed to a three-objectives optimization, being the *total flowtime* the last criterion. The initial population, the processing times and the due dates have been calculated as proposed in Ishibuchi and Murata (1998) and each set of these data has been used unaltered both when running the MHSO and the IMGA with the aim of allowing an easy comparison of the algorithms. In brief, the processing time of all jobs in a sequence have been specified as random numbers in the interval [1, 99] (or, in other words, they have been extracted from a discrete uniform distribution in the interval [1, 99]), whereas the due date of the single job has been evaluated at first on the basis of a randomly generated sequence. Indeed, given the completion time of the i-th job, namely CTM(i,m), the corresponding due date has been assumed equal to the following:

$$DD(i) = CTM(i,m) + random[-100,100]$$

Also, the fitness functions used for the linear scaling roulette wheel selection strategy and the strategy itself (as formulated by Goldberg, 1989) have been borrowed by the same work. The former is represented by the weighted sum of the adopted criteria, or, concisely:

$$f(s) = -\sum_{i=1}^{k} \omega_i f_i(s)$$

where k is the number of criteria, ω_i is the weight, $f_i(s)$ is the evaluation of the *i-th* criterion for the sequence s and, finally, f(s) is the overall fitness function. In the present work the same weights (constant multipliers) proposed by Ishibuchi and Murata (1998) have been adopted since the variance of the makespan is much smaller than that of the maximum tardiness (and differs from that of the total flowtime too), suggesting a normalization process aimed to handle the scheduling criteria equally. Furthermore, after several tests, they showed to be very effective and sensibly enhanced the whole algorithm. If Ψ designates the whole HM, it is certainly possible to evaluate its worst solution $f_{worst}(\Psi)$. In the case of a minimization problem this corresponds to the higher value of the fitness function. Hence, the selection probability can be defined as:

$$P(s) = \frac{f(s) - f_{worst}(\Psi)}{\sum_{s \in \Psi}\{f(s) - f_{worst}(\Psi)\}}$$

This value is used both when selecting two candidate sequences for the crossover required by the Large Recovery Procedure and when a single sequence is picked up to be overwritten by a better solution. In this last circumstance, owing to the fact that it is necessary to substitute the worst candidate with the higher probability, the selection is performed referring to the complementary of the above calculated chance ratio.

5.1 The 10 jobs – 5 machines problem with 2 objectives

A first class of 100 problems has been generated for n = 10 and m = 5, as this is the first test performed by Ishibuchi and Murata. This class is also interesting in that it allows to perform the exhaustive search and, therefore, to refer to a significant benchmark. The GA parameters have been set as follows:

- *PopSize* = 20;
- crossover probability = 0.9;
- mutation probability = 0.3.

The number of non-dominated chromosomes re-inserted within the population at each step has been fixed to 3 and the local search depth to 2. Finally, the Maximum Number of Iterations (MNI) has been put equal to 10000, in order to evaluate *PopSize* x MNI = 200000 chromosomes.
As an example, referring to the processing times and the due dates reported in Table 1 and Table 2, the IMGA required about 2.55 seconds on a Core 2 Duo T7200 processor (2.0 GHz) with 2 GB RAM.

TABLE 1 HERE

TABLE 2 HERE

On average, running 100 times the same problem, the algorithm found 12 non-dominated heuristic Pareto points for a total of 62 sequences, as depicted by small orange triangles in Figure 4. Instead, the exhaustive solution (represented in Figure 4 by diamonds) required about 91.45 seconds to weigh up 10! sequences, found 11 border points for a total of 111 sequences and showed that 4 out of 12 of the border points found by the IMGA were dominated by the optimal border.

FIGURE 4 HERE

The same problem was in turn submitted to a MHSO characterized by the following parameters:

- HMS = 12;
- HMCR = 0.95;
- PAR = linearly increasing between 0.05 and 0.35;
- LPRR = 0.0001;
- HRR = 0.7.

Referring to the maximum number of iterations and considering that the MHSO generates a single sequence at each step, whereas the IMGA generates *PopSize* chromosomes at each iteration, it was decided to set the value equal to *PopSize* × MNI. Actually, due to the fact that the MHSO performs the Harmony Recovery Procedure during the late iterations and that this involves a crossover procedure on the whole HM, it generally would examine a greater number of solutions than IMGA. Therefore, a constraint has been set to force the number of evaluated sequences to coincide with that of the IMGA. The algorithm required about 4.91 seconds to evaluate 200000 sequences and found the whole Pareto border (11 points) corresponding to the exhaustive solution (Figure 5), for a total of 96 sequences (86.5% of the complete solution set).

FIGURE 5 HERE

On average, the IMGA required about 2.5 seconds to evaluate 200000 chromosomes, obtaining 7.9 Pareto points and a total of 51.7 non-dominated sequences. Actually, a simple comparison with the solutions found by the MHSO showed that IMGA was able to find an average of 4.8 non-dominated Pareto points and a total of 42.5 non-dominated sequences, the rest being taken over by the MHSO frontier. Indeed, the MSHO required about 5.13 seconds to discover 7.1 Pareto border points and 69.8 non-dominated sequences. In this case, the solutions always coincided with those found by the exhaustive (except two cases, where the MHSO missed one point) and were never dominated by those found by the IMGA.
To better illustrate these results, a subset constituted by the first 10 trials has been reported in detail in Table 3.

TABLE 3 HERE

Briefly, the PB column reports the Pareto border points (*i.e.*, the non dominated solutions) found both by MHSO and IMGA, whereas the EPP columns shows the "equipollent" non-dominated points (two different sequences can be said "equipollent" if they share the same Pareto Point, or, in other words, if they give the same solutions for all the scheduling criteria). The third column reports the total number of non-dominated sequences (*i.e.*, the sum of PB and EPP) and Time indicates the seconds that the algorithms took to get the solutions. Further, Act PB and Act EP refer to the compared non dominated borders found by both algorithms. Concisely, they show how many solutions are not dominated by those found by the other algorithm. Finally, the last column indicates whether the sequences coincide with those found by the

exhaustive procedure. A second trial (the same sets of data) has been carried out modifying the MNI for the IMGA (this also modifies the maximum number of iterations for the MHSO). To improve the IMGA performances, it was decided to let run the algorithm for exactly 100000 iterations. For instance, with respect to the first problem (extensively discussed above), the algorithm required on average (over 100 runs) 25.91 seconds to evaluate 2000000 chromosomes and discovered all the 11 Pareto points and a total of 81 non-dominated solutions on 111 (72.97%). On the other hand, the MHSO needed about 66.72 seconds to elaborate 2000000 harmonies and to find all the 11 Pareto points and a total of 106 non-dominated sequences on 111 (95.49%). It emerged, however, that the MHSO spent most of the time in verifying the "equipollent" non-dominated memory. Therefore, a test has been performed introducing a constraint on the number of iterations: whereas the MNI for the IMGA has been left unaltered, the MHSO has been forced to stop after 1/3 of the original value. Results confirmed that the algorithm was still able to find the whole Pareto border in most cases, along with a greater number of "equipollent" solutions than the IMGA in less time. Indeed, on average, it found more than the 98% of the "equipollent" solutions evaluated in the previous test in 17.75 seconds. A further test has been performed increasing the IMGA population size to 30 chromosomes, but results did not differ significantly from those described above.

5.2 The 20 jobs – 10 machines problem with 2 objectives

The following step entailed the analysis of a two-objective scheduling with n = 20 and m = 10. The trial was carried out based on 20 runs. Following the indications given by Ishibuchi and Murata the IMGA settings were left unchanged with respect to the previous test, with the exception of the populations size, fixed to 30, and the MNI that was set to 50000 iterations in order to evaluate 1500000 chromosomes. Referring to the MHSO, only the memory size was changed to 24 harmonies.

As the exhaustive algorithm cannot be applied to this case, in order to compare the quality of the solutions two supplementary tools were used: (*i*)a graphical representation of the non-dominated Pareto frontier size evolving over time (*Non-dominated growing rate*, NDGR) and (*ii*) the Esbensen quality estimation (Esbensen, 1996). The former allows to get a direct and intuitive overview of the ability of the algorithm to find new solutions and the speed with which the non-dominated frontier improves (Figure 6), while the latter evaluates the average quality of the solutions. In addition, it has been introduced since it has been successfully used by Ishibuchi and Murata to assess the quality of the IMGA solutions and due to the fact that it represents a valuable tool to evaluate the quality also in a three-dimensional objective space (*i.e.*, when dealing with 3 objective problems).

FIGURE 6 HERE

Briefly, following Esbensen the quality of a set of non-dominated solution can be estimated by randomly generating a large number (k) of weights and - using these values - by calculating the objective function over

the entire non-dominated population. Greater values (smaller in *modulus*) indicate better solution sets. In brief, it is possible to write:

$$Q(S) = \frac{1}{k}\sum_{i=1}^{k} max\left\{-\sum_{j=1}^{m} \omega_j^i f_j(s)\right\}$$

where:

- Q(S) is the quality of the whole set of non-dominated solutions;
- *k* is the large number of weights sets (in the following *k* will be assumed equal to 10000);
- *m* is th number of criteria;
- ω_j^i indicates the weight for the *j-th* criterion within the *i-th* set. Weights are randomly generated and are normalized to 1;
- $f_j(s)$ is the evaluation of the *j-th* criterion for the sequence *s*.

The IMGA behaved well as usual, being able to find good non-dominated borders. However, the MHSO always outperformed it. On average, over 20 tests, the IMGA found 16.2 Pareto points and a total of 36.7 solutions in 42.83 seconds. On the other hand, the MHSO found 12.65 Pareto points and a total of 188.6 solutions in 65.78 seconds. However, despite of the little advantage with respect to the computing time, the true non-dominated Pareto points discovered by the IMGA were only 2.66 (with a total of non-dominated solutions equal to 1.73), being the remaining dominated by those found by the MHSO. To confirm this tendency, the Esbensen index resulted equal to -718.98 for the IMGA and to -532.45 for the MHSO. Only a few common Pareto points were found during these tests.
To better illustrate these results, a subset constituted by the first 10 trials has been reported in detail in Table 4.

TABLE 4 HERE

The graphical representation of the NDGR shows, on average, that the MHSO finds many non-dominated solutions very early, at the beginning of the run, and, later on, it improves the Pareto border continuously. At a later stage, the algorithm, thanks to the *Harmony Recovery* procedure, begins to explore new border portions finding other good solutions. On the contrary, the IMGA tends to maintain the non-dominated border unchanged after a certain amount of iterations and, often, is unable to improve it anymore. This behavior is clearly visible in Figure 7.

FIGURE 7 HERE

5.3 The 10 jobs – 5 machines and 20 jobs – 10 machines problems with 3 objectives

The second set of tests was aimed to solve the scheduling problem with respect to three objectives. After some runs necessary to set the parameters correctly, both the IMGA and the MHSO were modified only with respect to the population size (respectively, 30 chromosomes and 18 harmonies) as changes in the other parameters proved to be ineffective. To show the results of the 10 jobs – 5 machines problem, a subset constituted by the first 10 runs has been reported in detail in Table 5.

TABLE 5 HERE

The outcomes prove that the MHSO again outperformed the IMGA, being able to find a larger amount of true non-dominated Pareto points. The influence the *Harmony Recovery* procedure was even greater than in the previous cases. It emerges that the MHSO often finds a border that coincides with the one found by the exhaustive procedure, whereas the IMGA rarely identifies more than 50% of it.

The following step involved the analysis of a three-objective scheduling with $n = 20$ and $m = 10$. The test was carried out based on 20 runs. The IMGA settings were left unchanged with respect to the previous test, with the exception of the populations size, fixed to 30, and the MNI that was set to 50000 iterations in order to evaluate 1500000 chromosomes. Referring to the MHSO, only the memory size was increased to 36 harmonies. A subset constituted by the first 10 runs has been reported in detail in Table 6. Corresponding examples of NDGRs are shown in Figure 8.

TABLE 6 HERE

FIGURE 8 HERE

Again, it emerged that the MHSO spent most of the time in verifying the "equipollent" non-dominated memory and, therefore, it took more time to complete the job. A test has been performed introducing a constraint on the number of iterations: whereas the MNI for the IMGA has been left unchanged, the MHSO has been forced to stop after half of the original value. Results confirmed that the algorithm was still able to find the whole Pareto border in most cases, along with a greater number of "equipollent" solutions than the IMGA in less time. The outcomes have been reported in Table 7 and the NDGR corresponding to the first run is shown in Figure 9.

TABLE 7 HERE

FIGURE 9 HERE

Figure 10 shows the Pareto borders (in two dimensions, with respect to the *makespan* and the *tardiness* criteria) for the MHSO and the IMGA found at the end of the first run, that required similar computation

times (45.66 and 43.73 seconds respectively). After comparing the two sets of data the whole IMGA border results to be dominated, confirming that even when evaluating half sequences with respect to the IMGA, the MHSO is able to appraise a very good solution.

FIGURE 10 HERE

This is also supported by the average values of the Esbensen index. Indeed, for the IMGA it remains almost unchanged (as obvious, since no modifications were introduced within the algorithm), while for the MHSO it grows a bit (in *modulus*). This means a worse solution with respect to the unconstrained case, but the MHSO still performs better than the IMGA.

5.4 The 30 jobs – 15 machines problem with 2 and 3 objectives

To further verify the algorithm capabilities additional tests were carried out on larger problems. Owing to the fact that Ishibuchi and Murata (1998) do not report data for these sets of problems some efforts were devoted to test the IMGA and to fix the parameters that give the best results before starting the comparative analysis. In particular, it emerged that the most sensible parameter is the population size, whereas the crossover and the mutation probabilities are effective in a limited range around the values proposed by the authors (these were, therefore, maintained almost unchanged).

In the case of $n = 30$ and $m = 15$ the best performance for the IMGA was reached with a population size of 40 chromosomes and 50000 iterations. The crossover and mutation probabilities have been taken equal to 0.9 and 0.35 respectively. The number of non-dominated chromosomes re-inserted within the population at each step has been fixed to 3 and the local search depth to 2 as for smaller problems. Comparison was carried out with a MHSO having a population of 36 harmonies. Also, the MHSO was forced to stop after evaluating 40% of the solutions evaluated by the IMGA, with the aim of making it more performing in terms of computational time (as already stated, after a number of iterations the MHSO spends most of the time in verifying the "equipollent" non-dominated memory and for large problems this penalizes it in terms of time). The test consisted of 100 runs and results showed that the IMGA required on average 23.69 seconds to find 23 non dominated Pareto points and 41 equipollent solutions (the corresponding Esbensen quality estimation was equal to -1468.60). On the other hand, the MHSO found on average 28 Pareto points and 63 equipollent solutions in 18.03 seconds (with an Esbensen estimation of -1432.06). Comparing the two Pareto borders it emerged, however, that the MHSO again outperformed the IMGA: on average the former found 9 Pareto border points and 22 equipollent solutions that the IMGA did not discover or dominate. On the other hand, the latter found 4 Pareto points and 5 equipollent solutions that were not present within the solutions obtained by the MHSO. The IMGA gave better results (a better Pareto border and the corresponding Esbensen estimation) only 7 times on 100 runs. Results are summarized in Table 8, where ND is the column of the non-dominated solutions, Eqp that of the equipollent non-dominated points for the two algorithms. On

the other hand, the Exclusive ND and Eqp represent the non dominated solution that each algorithm did not discover or dominate with respect to the other.

TABLE 8 HERE

The same test was repeated introducing the third criterion and modifying the population size parameter both for the IMGA and the MHSO (respectively, 50 chromosomes and 40 harmonies). The other parameters were left unchanged (after some tests these proved to be the most efficient), with the exception of the MHSO that was forced to stop after evaluating 50% of the solutions evaluated by the IMGA. The MHSO performed even better than in the previous case, being able to find on average 86 non dominated points and 42 equipollent solutions, whereas the IMGA found 92 non dominated points and 56 equipollent solutions. However, comparing the Pareto borders it emerged clearly that on average the former found 51 Pareto border points and 2 equipollent solutions that the IMGA did not discover, while the latter found 9 Pareto points and 5 equipollent solutions that were not present within the solutions obtained by the MHSO. The average Esbensen quality estimations were, respectively, -16905.71 and -17501.43.

5.5 The 50 jobs – 20 machines problem with 2 and 3 objectives

In the case of n = 50 and m = 20 with 2 objectives the best performance was obtained with a population size of 60 chromosomes and 50000 iterations for the IMGA (again, the crossover and mutation probabilities have been taken equal to 0.9 and 0.35 respectively. The number of non-dominated chromosomes re-inserted within the population at each step has been fixed to 3 and the local search depth to 2) and 40 harmonies for the MHSO. Again, the MHSO was forced to stop after evaluating 50% of the solutions evaluated by the IMGA, with the aim of making it less time consuming. The test consisted of 100 runs and results showed that the IMGA required on average 73.12 seconds to find 27 non dominated Pareto points and 21 equipollent solutions (the corresponding Esbensen quality estimation was equal to -2373.04). On the other hand, the MHSO found on average 24 Pareto points and 33 equipollent solutions in 65.71 seconds (with an Esbensen estimation of -2300.85). Comparing the two Pareto borders it emerged again that the MHSO worked better than the IMGA: on average the former found 11 Pareto border points and 31 equipollent solutions that the IMGA did not discover. On the other hand, the latter found 7 Pareto points and 11 equipollent solutions that were not present within the solutions obtained by the MHSO. The IMGA gave better results (a better Pareto border and the corresponding Esbensen estimation) only 5 times on 100 runs.

The runs were repeated for the three objective problem. This time, all parameters were left unchanged with respect to the previous test, as these proved to be the most efficient in a significant number of tests. The MHSO performed satisfactorily, being able to find on average 31 non dominated points and 67 equipollent solutions, whereas the IMGA found 115 non dominated points and 77 equipollent solutions. Comparing the Pareto borders it emerged that on average the MHSO found 31 Pareto border points and 2 equipollent

solutions that the IMGA did not discover or dominate, while the latter found 13 Pareto points and 0 equipollent solutions that were not present within the solutions obtained by the MHSO. The average Esbensen quality estimations were, respectively, -43090.63 and -46445.67.

5.6 The 100 jobs – 30 machines problem with 2 and 3 objectives

Finally, the algorithm was applied to the case of n = 100 and m = 30 with 2 and 3 criteria. Referring to the IMGA the best performance was obtained with a population size of 60 chromosomes and 40000 iterations. Again, the crossover and mutation probabilities have been taken equal to 0.9 and 0.35 respectively. The number of non-dominated chromosomes re-inserted within the population at each step has been fixed to 3 and the local search depth to 2. The same problem was in turn submitted to a MHSO characterized by the following parameters:

- HMS = 40;
- HMCR = 0.65;
- PAR = linearly increasing between 0.05 and 0.15;
- LPRR = 0.0001;
- HRR = 0.85.

The MHSO was forced to stop after evaluating 60% of the solutions evaluated by the IMGA to reduce the computing time. With respect to the two-criteria problem, the test consisted of 100 runs and results showed that the IMGA required on average 98.56 seconds to find 36 non dominated Pareto points and 11 equipollent solutions (the corresponding Esbensen quality estimation was equal to -5123.74). On the other hand, the MHSO found on average 25 Pareto points and 9 equipollent solutions in 109.93 seconds (with an Esbensen estimation of -5068,77). Comparing the two Pareto borders it emerged again that the MHSO worked better than the IMGA: on average the former found 13 Pareto border points and 7 equipollent solutions that the IMGA did not discover. On the other hand, the latter found 5 Pareto points and 10 equipollent solutions that were not present within the solutions obtained by the MHSO. All other points found by the IMGA were dominated by the MHSO border. The IMGA gave better results (a better Pareto border and the corresponding Esbensen estimation) 9 times on 100 runs.

The runs were subsequently repeated for the three objective problem. All parameters were left unchanged with respect to the previous test, as these proved to be the most efficient in a significant number of runs. The MHSO found on average 168 non dominated points and 5 equipollent solutions, whereas the IMGA found 143 non dominated points and 2 equipollent solutions. However, comparing the Pareto borders it emerged that on average the MHSO found 76 Pareto border points and 1 equipollent solutions that the IMGA did not discover or dominate, while the latter found 39 Pareto points and 0 equipollent solutions that were not present within the solutions obtained by the MHSO. The average Esbensen quality estimations were, respectively, -162157.12 and -169322.37. For such large problems, during the tests the IMGA showed to be

somewhat faster than the MHSO. Indeed, the IMGA required on average about 102 seconds to run, whereas the MHSO, evaluating 60% of the solutions calculated by the IMGA, needed about 123 seconds to complete. Results of this test are summarized in Table 9.

5.7 Kruskal-Wallis hypothesis test for homogeneity

To complete the validation of the algorithm, a Kruskal-Wallis hypothesis test for homogeneity (Law and Kelton, 1991) was performed to show that the non dominated Pareto border found by the MHSO outperforms the IMGA. The test was applied to the whole set of problems and showed that the MHSO obtained efficient solutions. Fox example, in the case of the 50 jobs, 20 machines and two criteria problem based on 50 runs (*i.e.*, 50 different problems) for each algorithm, the statistic (T) was evaluated for two independent samples (k = 2) and at a level α = 0.99. Results clearly showed that the null hypothesis can be rejected, being T = 28.77 and $\chi^2_{2-1,1-0.99}$ = 6.635 (where $\chi^2_{k-1,1-\alpha}$ is the upper 1 – α critical value for a Chi-square distribution with k – 1 degrees of freedom). When the test was performed with respect to the 100 jobs, 30 machines problem there was evidence that the null hypothesis can be rejected again, being T = 21.15. The difference between T and $\chi^2_{k-1,1-\alpha}$ is even greater for smaller problems. Consequently, the Pareto border found by the MHSO is actually different from the one found by the IMGA and, considering the Esbensen quality estimations, it is possible to affirm that the modified Harmony Search behaved better that the Genetic Algorithm.

Conclusions

The paper has presented a modified Harmony Search Optimization (MHSO) algorithm applied to multi-criteria permutation flowshop scheduling problems with due dates. Also, the results of a comparative analysis with respect to the Genetic Algorithm (GA) proposed by Ishibuchi and Murata (1998) are presented, with the aim of showing its remarkable performances. This choice is due to the fact that the mentioned GA is known to be particularly effective in the solution of the above mentioned problem. It is also known that in similar situations the performances of GAs are seldom exceeded or even achieved by other procedures representing, therefore, a valuable benchmark.

The problems that have been examined in depth can be divided into two large sets, characterized by the number of optimization criteria (respectively 2 and 3) and each subdivided into two subsets with respect to the number of jobs (respectively, n = 10, 20, 30, 50 and 100) and machines (we assumed, respectively, m = 5, 10, 15, 20 and 30). The first set refers to a two-objectives optimization, where the adopted criteria are the *makespan* and the *maximum tardiness*, whereas the latter is aimed to a three-objectives optimization, being the *total flowtime* the third criterion.

The outcomes show that the MHSO outperforms the IMGA, being able to find a larger amount of true non-dominated Pareto points. When applied to small sized problems (n = 10), the exhaustive evaluation has shown that the MHSO often finds the whole border. As the exhaustive algorithm cannot be applied to larger problems, in order to compare the quality of the solutions two supplementary tools have been introduced: (*i*)a graphical representation of the non-dominated Pareto frontier size evolving over time and (*ii*) the Esbensen quality estimation (Esbensen, 1996). To complete the validation of the algorithm a Kruskal-Wallis hypothesis test for homogeneity has been performed to show that the non dominated Pareto borders found by the MHSO actually outperform those found by the IMGA.

Due to the fact that the MHSO takes more time than the GA when evaluating the same number of sequences, constraints have been introduced to the number of iterations of the MHSO to verify its performances even in unfavorable situations. Results clearly show the validity of the proposed algorithm in that it still outperforms the GA in several tests.

With respect to potential future works, the obtained results suggest to enhance the neighborhood search procedure to speed up the algorithm and to adapt it to other problems such as, for instance, layout optimization and network analysis (in particular Hub-and-Spoke problems). Also, it would be useful to develop the algorithm integrating it with other metaheuristic methods (GAs, Simulated Annealing, *etc.*) in an attempt to further improve its overall performance.

References

Braglia, M., Grassi, A., (2009), A new heuristic for the flowshop scheduling problem to minimize makespan and maximum tardiness, *International Journal of Production Research*, 47(1), 273-288.

Campbell, H.G., Dudek R.A., Smith M.L., 1970, A heuristic algorithm for the n-job m-machine sequencing problem, *Management Science*, 16B, 630-637.

Cao, D., Chen, M., 2003, Parallel flowshop scheduling using Tabu search, *International Journal of Production Research*, 41(13), 3059-3073.

Carlier, J., Rebai, I., 1996, Two branch-and-bound algorithms for the permutation flow shop problem, *European Journal of Operational Research*, 90, 238-251.

Cheng, J., Kise, H., Matsumoto, H., 1997, A branch-and-bound algorithm with fuzzy inference for a permutation flow shop scheduling problem, *European Journal of Operational Research*, 96, 578-590.

Cheng, J., Steiner, G., Stephenson, P., 2001, A computational study with a new algorithm for the three-machine permutation flow-shop problem with release time, *European Journal of Operational Research*, 130, 559-575.

Dannenbring, D.G., 1977, An evaluation of flow shop sequencing heuristics, *Management Science*, 23, 1174-1182.

Esbensen, H., 1996, *Defining solution set quality*, Elect. Res. Lab., College Eng., University of California, Berkeley, Memorandum, UCB/ERL M96/1.

Grabowski, J., Wodecki, M., 2004, A very fast Tabu Search algorithm for the permutation flow shop problem with makespan criterion, *Computers and Operations Research*, 31, 1891-1909.

Gangadharan, R., Rajendran, C., 1994, A simulated annealing heuristic for scheduling in a flowshop with bicriteria, *Proc. Of the 16 th International Conference on Computers and Industrial Engineering*, 345-348.

Geem, Z.W., Kim, J.H., Loganathan, G.V., 2001, A new heuristic optimization algorithm: Harmony Search, *Simulation*, 76(2), 60-68.

Geem, Z.W., Lee, K.S., Park, Y., 2005, Application of harmony Search to vehicle routing, *American Journal of Applied Sciences*, 2(12), 1552-1557.

Geem, Z.W., 2006, Optimal cost design of water distribution networks using Harmony Search, *Engineering Optimization*, 38(3), 259-280.

Goldberg, D.E., 1989, *Genetic Algorithms in Search, Optimization and Machine Learning*, Reading, MA, Addison-Wesley.

Gupta, J.N.D., Stafford, Jr.E.F., 2006, Flowshop scheduling research after five-decades, *European Journal of Operational Research*, 169, 699-711.

Haouari, M., Ladhari, T., 2003, A branch-and-bound-based local search method for the flow shop problem, *Journal of the Operational Research Society*, 54, 1076-1084.

Hejazi, S.R., Saghafian, S., (2005), Flowshop-scheduling problems with makespan criterion: a review, *International Journal of Production Research*, 43(14), 2895-2929.

Ignall, E., Schrage, L.E., 1965, Application of branch and bound technique to some flow-shop problems, *Operations Research*, 13(3), 400-412.

Ishibuchi, H., Murata, T., 1998, A multi-objective genetic local search algorithm and its application to flowshop scheduling, *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and reviews*, 28(3), 392-403.

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

Khan, B.S.H., Prabhaharan, G., Asokan, P., 2007, A grasp algorithm for m-machine flowshop scheduling problem with bicriteria of makespan and maximum tardiness, *International Journal of Computer Mathematics*, 84(12), 1731-1741.

Kim, J.H., Geem, Z.W., Kim, E.S., 2001, Parameter estimation of the nonlinear Muskin-gum model using Harmony Search, *Journal of the American Water Resources Association*, 37(5), 1131-1138.

Kim, S.H., Yoo, W.S., Oh, K.J., Hwang, I.S., Oh, J.E., 2006, Transient analysis and leakage detection algorithm using GA and HS algorithm for a pipeline system, *Journal of Mechanical Science and Technology*, 20(3), 426-434.

Ladhari, T., Haouari, M., 2005, A computational study of the permutation flow shop problem based on a tight lower bound, *Computers and Operations Research*, 32(7), 1831-1847.

Lageweg, B.J., Lenstra, J.K., Rinnooy Kan A.H.G., 1978, A general bounding scheme for the permutation flow-shop problem, *Operations Research*, 26, 53-67.

Law, A.M. and Kelton, W.D., *Simulation Modeling and Analysis*, McGraw-Hill, New York, NY, 1991.

Lee, K.S., Geem, Z.W., 2004, A new structural optimization method based on the Harmony Search algorithm, *Computers and Structures*, 82(9 – 10), 781-798.

Lin, S.W:, Gupta, J.N.D., Ying, K.C., Lee, Z.J., 2009, Using simulated annealing to schedule a flowshop manufacturing cell with sequence-dependent family setup times, *International Journal of Production Research*, 47(12), 3205-3217.

Murata, T., Ishibuchi, I., Tanaka, H., 1996, Multi-objective Genetic Algorithm and its applications to flowshop scheduling, *Computers and Industrial Engineering*, Vol. 30, No. 4, 957-968.

Nawaz, M., Enscore, Jr.E.E., Ham, I., 1983, A heuristic algorithm for the m-machine, n-job flow shop sequencing problem, *Omega*, 11, 91-95.

Nowicki, E., Smutnicki, C., 1996, A fast tabu search algorithm for the flow shop problem, *European Journal of Operational Research*, 91, 160-175.

Osman, I.H., Potts, C.N., 1989, Simulated annealing for permutation flow-shop scheduling, *Omega*, 17, 551-557.

Pan, J.C.H, Fan, E.T, 1997, Two-machine flowshop scheduling to minimize total tardiness, *International Journal of Systems Science*, 28(4), 405-414.

Pinedo, M., *Scheduling: theory, algorithms, and systems*, Englewood Cliffs, NJ, Prentice-Hall, 1995.

Ponnanbalam, S.G., Jagannathan, H., Kataria, M., Gadicherla, A., 2004, A TSP-GA multi objective algorithm for the flowshop scheduling, *International Journal of Advanced Manufacturing Technology*, 23, 909-915.

Potts, C.N., 1980, An adaptive branching rule for the permutation flow-shop problem, *European Journal of Operational Research*, 5, 19-25.

Rajendran, C., 1994, A heuristic for scheduling in flowshop and flowline-based manufacturing cell with multi-criteria, International Journal of Production Research, 32(11), 2541-2558.

Rajendran, C., Ziegler, H., 2005, Two ant-colony algorithms for minimizing total flowtime in permutation flowshops, *Computers and Industrial Engineering*, 48, 789-797.

Rajkumar, R., Shahabudeen, P., 2009, An improved genetic algorithm for the flowshop scheduling problem, *International Journal of Production Research*, 47(1), 233-249.

Schaffer, J.D., 1985, Multiple objective optimization with vector evaluated genetic algorithms, *Proc. Of the 1 th ICGA*, 93-100.

Srikanth, K.I., Barkha, S., 2004, Improved genetic algorithm for the permutation flowshop scheduling problem, *Computers and Operations Research*, 31(4), 593-606.

Stevens, J.W., Gemill, D.D., (1997), Scheduling a two-machine flowshop with travel times to minimize maximum lateness, *International Journal of Production Research*, 35(1), 1-15.

Tasgetiren, M.F., Liang, Y.C., Sevkli, M., Gencylmaz, G., 2007, A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem, *European Journal of Operational Research*, 177, 1930-1947.

Tavakkoli-Moghaddam, R., Rahimi-Vahed, A., Hossein Mirzaei, A., 2007, Solving a multi-objective no-wait flowshop problem by a hybrid multi-objective immune algorithm, *Multiprocessor Scheduling: Theory and Applications*, Itech Education and Publishing, Vienna, Austria.

T'Kindt, V., Billaut, J.C., 2001, Multicriteria scheduling problems: a survey, *Operations Research*, 35, 143-163.

Tseng, L.Y., Lin, Y.T., 2009, A hybrid genetic local search algorithm for the permutation flowshop scheduling problem, *European Journal of Operational Research*, 198(1), 84-92.

Yandra, Tamura, H., 2007, A new multiobjective genetic algorithm with heterogeneous population for solving flowshop scheduling problems, *International Journal of Computer Integrated Manufacturing*, 20(5), 465-477.

Table captions

	Machine 1	*Machine 2*	*Machine 3*	*Machine 4*	*Machine 5*
Job 1	85	31	3	84	18
Job 2	98	8	94	69	7
Job 3	26	75	30	24	43
Job 4	18	10	9	25	11
Job 5	44	13	96	45	7
Job 6	42	10	50	39	87
Job 7	96	42	81	13	28
Job 8	15	84	33	27	4
Job 9	96	39	14	97	93
Job 10	49	72	83	81	87

Table 1 – Processing times for a 10 jobs – 5 machines problem

Due 1	92
Due 2	258
Due 3	321
Due 4	432
Due 5	599
Due 6	666
Due 7	772
Due 8	852
Due 9	928
Due 10	928

Table 2 – Due dates for a 10 jobs – 5 machines problem

MHSO							
PB	EPP	Tot	Time	Act PB	Act EP	%	
11	83	94	3,91	11	83	84,68%	Y
10	11	21	4,34	10	11	100,00%	Y
6	15	21	5,12	6	15	100,00%	Y
6	233	239	4,98	6	233	32,61%	Y
5	76	81	7,34	5	76	60,45%	Y
8	11	19	3,23	8	11	95,00%	N
8	2	10	4,12	8	2	100,00%	Y
4	88	92	5,25	4	88	46,94%	Y
7	35	42	3,62	7	35	68,85%	Y
7	68	75	5,22	7	68	57,69%	Y
7,20	62,20	69,40	4,71	7,20	62,20	0,75	

Exhaustive	
PB	Tot
11	111
10	21
6	21
6	733
5	134
9	20
8	10
4	196
7	61
7	130

IMGA							
PB	EPP	Tot	Time	Act PB	Act EP	%	
9	25	34	2,51	8	25	29,73%	N
13	5	18	2,46	6	5	52,38%	N
6	9	15	2,36	5	9	66,67%	N
8	145	153	2,56	4	145	20,33%	N
5	61	66	2,41	5	61	49,25%	Y
7	5	12	2,47	2	3	25,00%	N
8	1	9	2,52	5	1	60,00%	N
3	70	73	2,52	3	70	37,24%	N
8	21	29	2,45	5	21	42,62%	N
7	54	61	2,53	6	54	46,15%	N
7,40	39,60	47,00	2,48	4,90	39,40	0,43	

Table 3 – Outcomes of 10 runs for a 10 jobs – 5 machines – 2 objectives problem

MHSO							
PB	EPP	Tot	Time	Act PB	Act EP	Esbensen	Common
14	15	29	67,08	10	21	-549,70	2
15	75	90	65,80	15	75	-521,37	0
20	291	311	66,59	20	291	-742,65	0
14	8	22	70,81	13	18	-668,94	1
6	38	44	65,48	6	38	-309,03	0
6	976	982	65,68	6	976	-405,84	0
12	280	292	65,09	12	280	-554,89	0
9	43	52	65,97	9	43	-555,69	0
11	14	25	66,02	11	14	-494,78	0
20	40	60	66,73	20	40	-477,93	0
12,70	178,00	190,70	66,53	12,20	179,60	-528,08	

IMGA						
PB	EPP	Tot	Time	Act PB	Act EP	Esbensen
14	15	29	43,68	2	0	-673,75
14	12	26	42,58	2	0	-751,94
20	23	43	42,62	2	0	-839,73
9	10	19	42,25	6	7	-756,75
11	5	16	53,61	0	0	-655,87
12	61	73	42,01	0	0	-656,41
16	26	42	42,83	3	6	-711,91
17	30	47	42,92	4	0	-736,60
7	13	20	43,16	2	6	-664,06
21	7	28	43,12	3	0	-784,04
14,10	20,20	34,30	43,88	2,40	1,90	-723,11

Table 4 – Outcomes of 10 runs for a 20 jobs – 10 machines – 2 objectives problem

MHSO							Common	Exhaustive	
PB	EPP	Tot	Time	Act PB	Act EP	%		PB	Tot
119	3	122	5,50	119	3	96,83%	21	123	126
85	1	86	4,44	85	1	87,76%	33	96	98
34	1	35	4,19	34	1	83,33%	12	40	42
41	0	41	5,66	41	0	97,62%	16	42	42
52	0	52	5,03	52	0	100,00%	21	52	52
47	0	47	4,41	47	0	100,00%	6	47	47
27	0	27	5,01	27	0	93,10%	14	29	29
9	0	9	7,11	9	0	100,00%	4	9	9
49	0	49	4,34	49	0	94,23%	8	51	52
33	0	33	4,39	33	0	100,00%	17	33	33
49,60	0,50	50,10	5,01	49,60	0,50	0,95			

IMGA						
PB	EPP	Tot	Time	Act PB	Act EP	%
87	0	87	2,83	21	0	16,67%
54	1	55	2,81	34	1	35,71%
27	0	27	2,59	16	0	38,10%
30	0	30	2,58	16	0	38,10%
36	0	36	2,69	21	0	40,38%
27	0	27	2,62	7	0	14,89%
23	2	25	2,78	14	0	48,28%
9	0	9	2,52	4	0	44,44%
43	0	43	2,78	11	0	21,15%
30	4	34	2,64	17	0	51,52%
36,60	0,70	37,30	2,68	16,10	0,10	0,35

Table 5 – Outcomes of 10 runs for a 10 jobs – 5 machines – 3 objectives problem

MHSO							
PB	EPP	Tot	Time	Act PB	Act EP	Esbensen	Common
96	7	103	89,27	96	7	-1384,46	0
63	0	63	89,83	63	0	-792,25	0
111	0	111	88,33	111	0	-988,98	0
190	0	190	90,39	190	0	-982,77	0
77	1	78	88,86	77	1	-972,37	0
91	1	92	88,19	91	1	-733,55	1
184	4	188	90,80	184	4	-1078,09	0
157	6	163	88,20	157	6	-1149,36	0
97	0	97	88,69	97	0	-685,71	0
82	1	83	89,78	82	1	-1038,82	0
114,80	2,00	116,80	89,23	114,80	2,00	-980,64	

IMGA						
PB	EPP	Tot	Time	Act PB	Act EP	Esbensen
61	3	64	45,3	3	3	-3848,02
31	0	31	43,83	5	0	-3683,66
71	0	71	44,59	8	0	-3728,62
101	0	101	45,92	4	0	-3657,03
52	0	52	44,16	0	0	-3934,26
42	0	42	43,94	2	0	-3580,59
51	0	51	44,41	2	0	-4041,30
67	0	67	45,19	0	0	-3844,47
46	0	46	45,2	1	0	-3839,29
51	0	51	44,41	3	0	-3641,25
57,30	0,30	57,60	44,70	2,80	0,30	-3779,85

Table 6 – Outcomes of 10 runs for a 20 jobs – 10 machines – 3 objectives problem

MHSO							
PB	EPP	Tot	Time	Act PB	Act EP	Esbensen	Common
108	1	109	45,66	108	1	-739,73	0
82	0	82	44,86	82	0	-969,34	0
117	3	120	44,50	117	3	-1446,53	0
95	21	116	45,81	95	21	-1392,46	0
66	0	66	45,01	66	0	-856,68	0
66	2	68	45,05	66	2	-931,56	0
122	0	122	44,98	122	0	-927,24	0
61	0	61	46,09	61	0	-1203,56	0
54	0	54	45,11	54	0	-685,18	0
122	1	123	46,02	122	1	-1000,75	0
89,30	2,80	92,10	45,31	89,30	2,80	-1015,30	

IMGA						
PB	EPP	Tot	Time	Act PB	Act EP	Esbensen
45	0	45	43,73	0	0	-3730,77
71	0	71	44,94	25	0	-3911,00
72	1	73	46,78	4	0	-3745,36
80	3	83	46,16	20	2	-3759,68
43	0	43	47,28	2	0	-3666,02
36	0	36	44,77	4	0	-3693,97
51	0	51	45,17	0	0	-3623,60
38	0	38	44,48	4	0	-3877,27
37	0	37	44,59	0	0	-3564,01
92	0	92	45,75	0	0	-4132,84
56,50	0,40	56,90	45,37	5,90	0,20	-3770,45

Table 7 – Outcomes of 10 runs for a 20 jobs – 10 machines – 3 objectives problem with constraints on the MHSO iterations

	ND	Eqp	Esbensen	Exclusive ND	Exclusive Eqp
MHSO	28	63	-1458,95	9	22
IMGA	23	41	-1482,00	4	5

Table 8 – Average outcomes of 100 runs for a 30 jobs – 15 machines – 2 objectives problem

	ND	Eqp	Esbensen	Exclusive ND	Exclusive Eqp
MHSO	168	5	-162157,12	76	1
IMGA	143	2	-169322,37	39	0

Table 9 – Average outcomes of 100 runs for a 100 jobs – 30 machines – 3 objectives problem

Figure captions

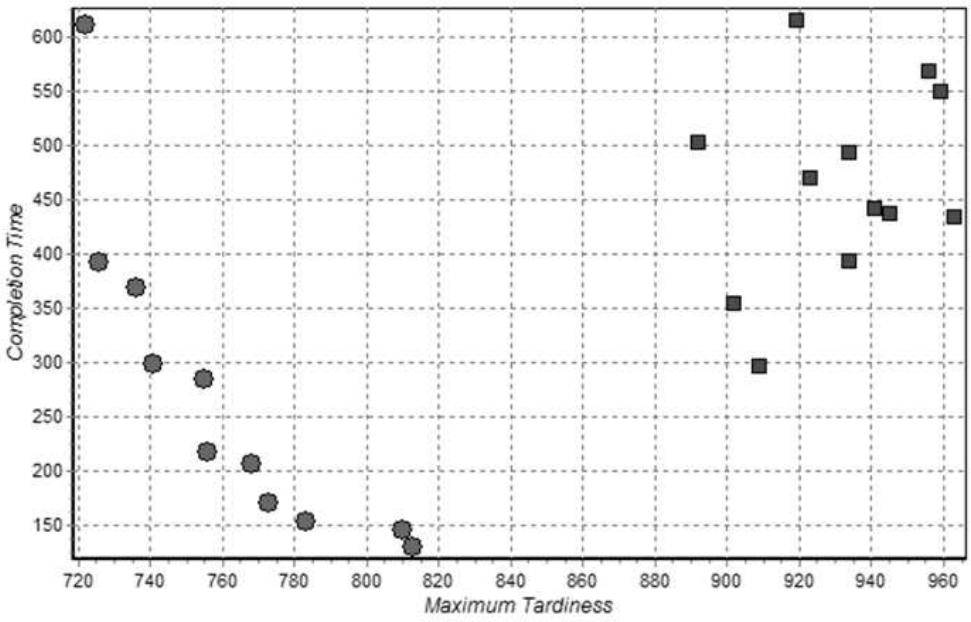
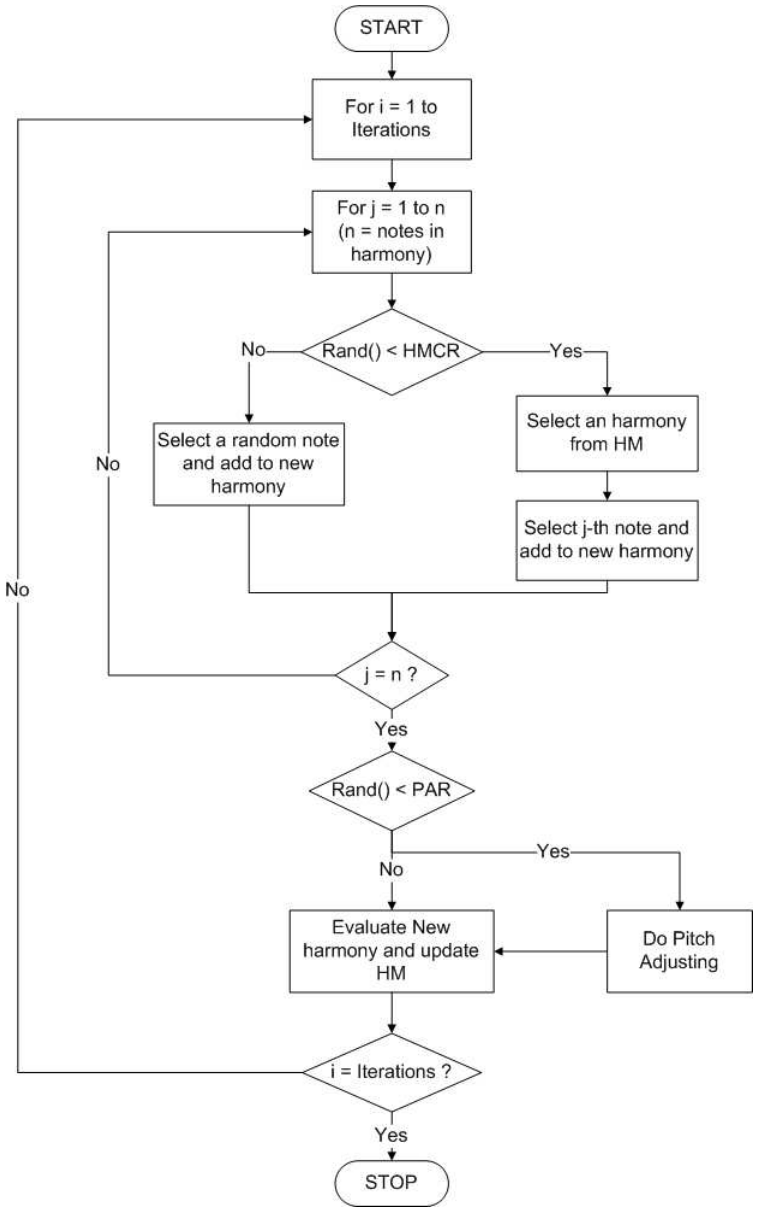
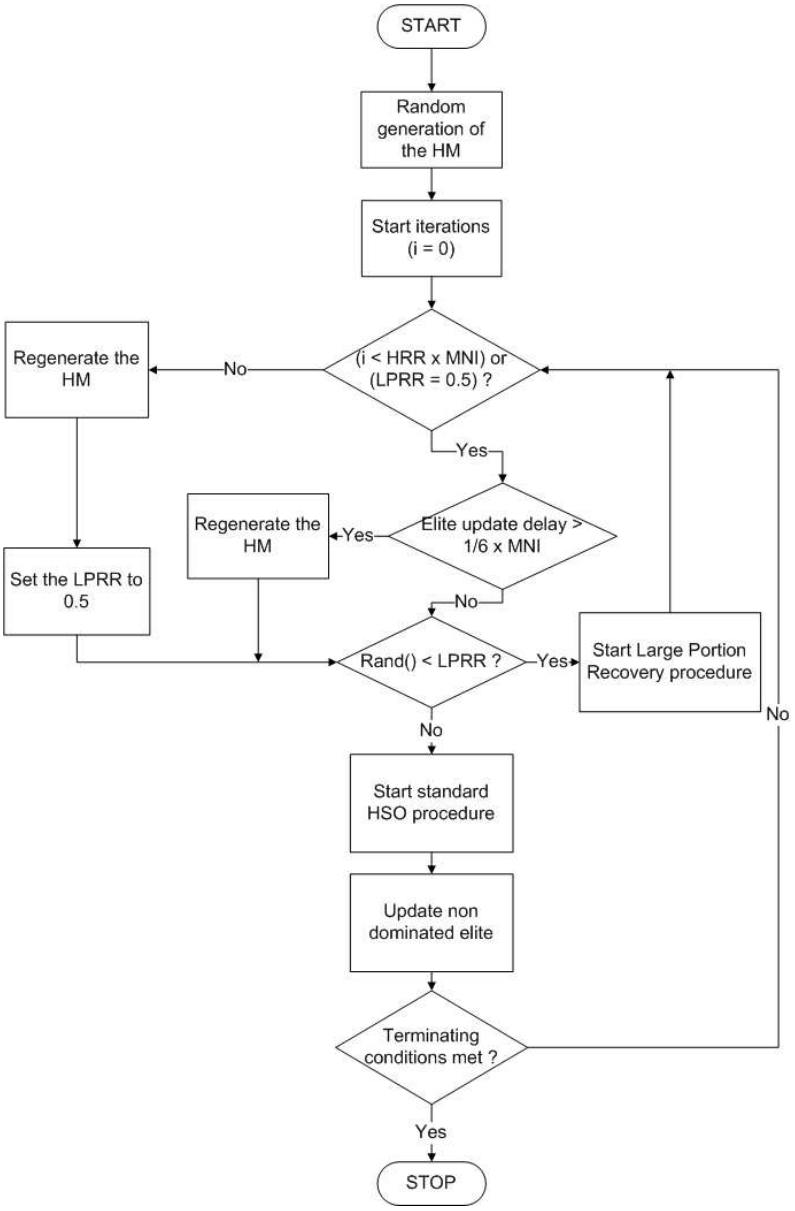


Figure 1 – Dominated (square points) and non-dominated (rounded points) solutions
204x130mm (88 x 88 DPI)



164x263mm (96 x 96 DPI)



181x276mm (96 x 96 DPI)

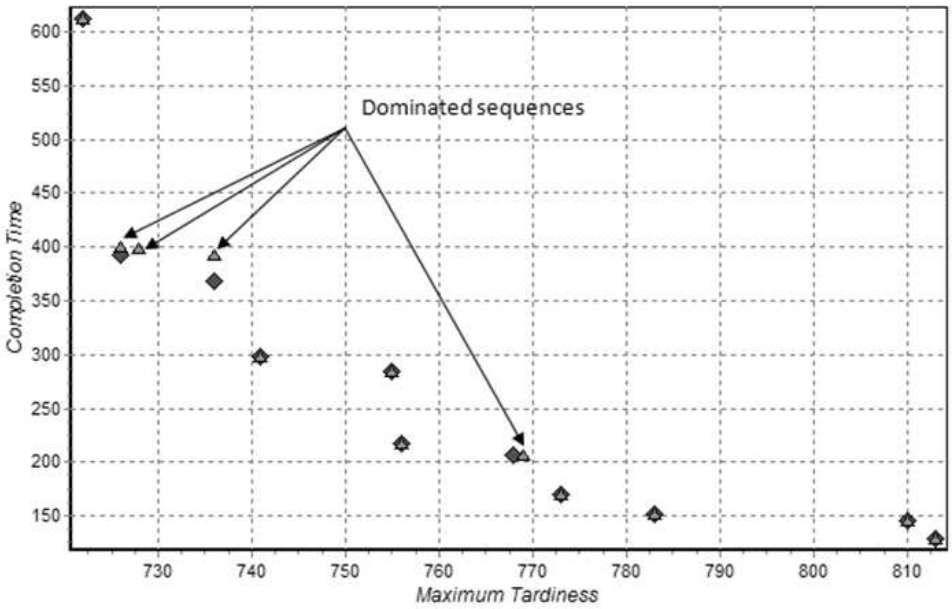


Figure 4 – IMGA and Exhaustive Pareto non-dominated frontier for a two-objective problem
210x135mm (88 x 88 DPI)

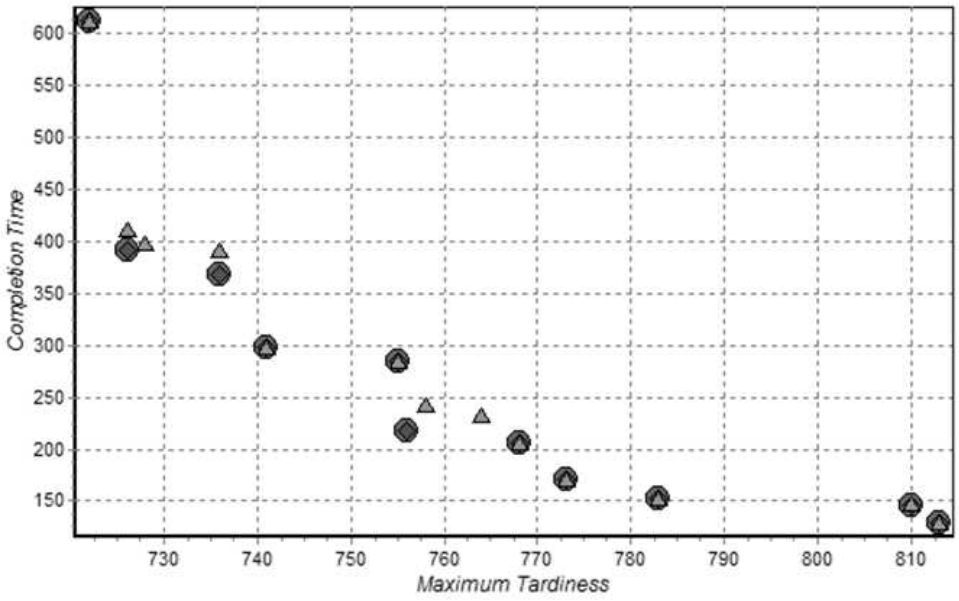


Figure 5 – IMGA (triangles), MHSO (circles) and exhaustive Pareto (diamonds) non-dominated frontier for a two-objective problem
199x123mm (88 x 88 DPI)

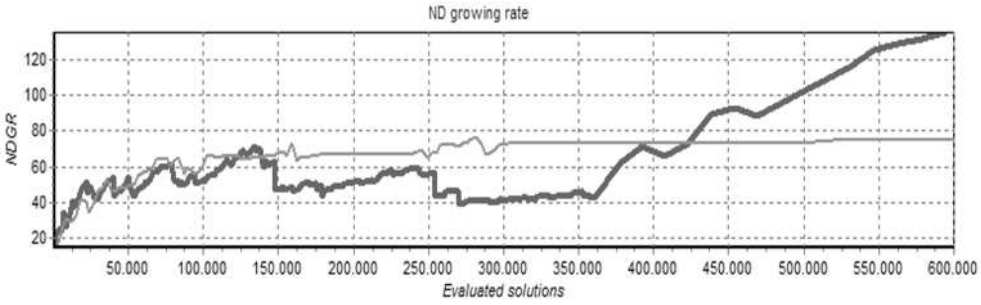


Figure 6 – NDGR for the MHSO (thick line) and for the IMGA
224x73mm (88 x 88 DPI)

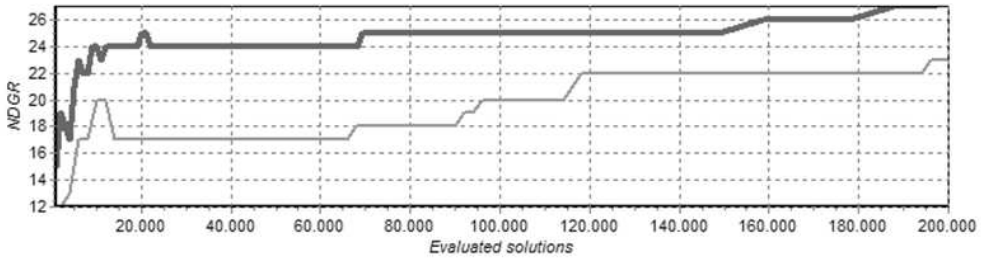


Figure 7 – NDGR for the MHSO (thick line) and for the IMGA in a 20 jobs – 10 machines – 2 objectives problem
227x64mm (88 x 88 DPI)

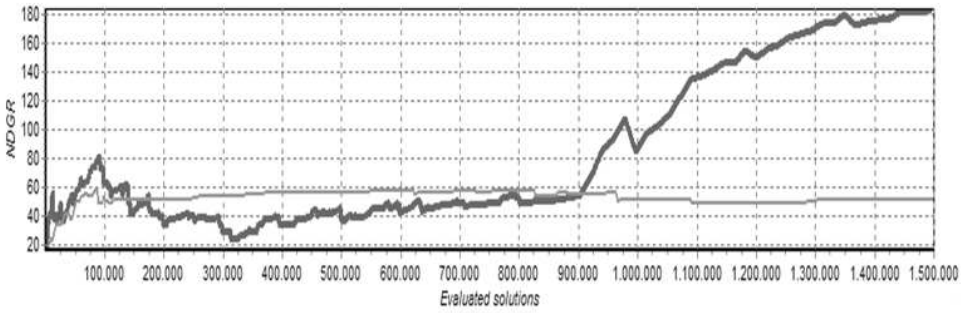
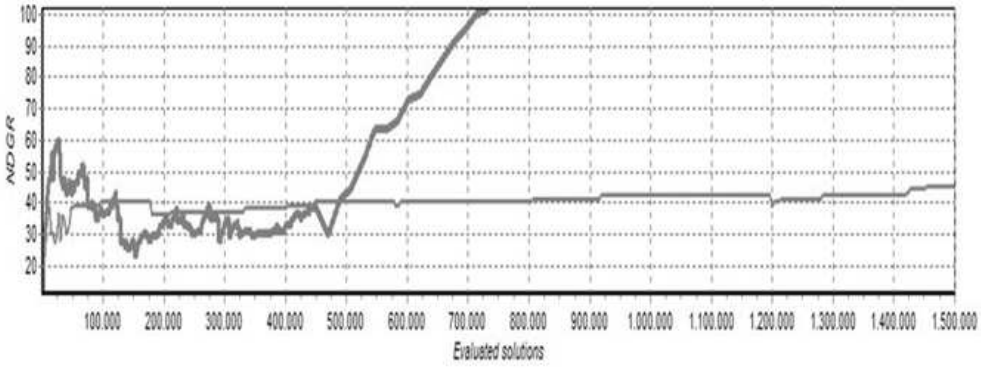


Figure 8 – Example of NDGR for the MHSO (thick line) and for the IMGA in a 20 jobs – 10 machines – 3 objectives problem
230x76mm (88 x 88 DPI)



195x74mm (88 x 88 DPI)

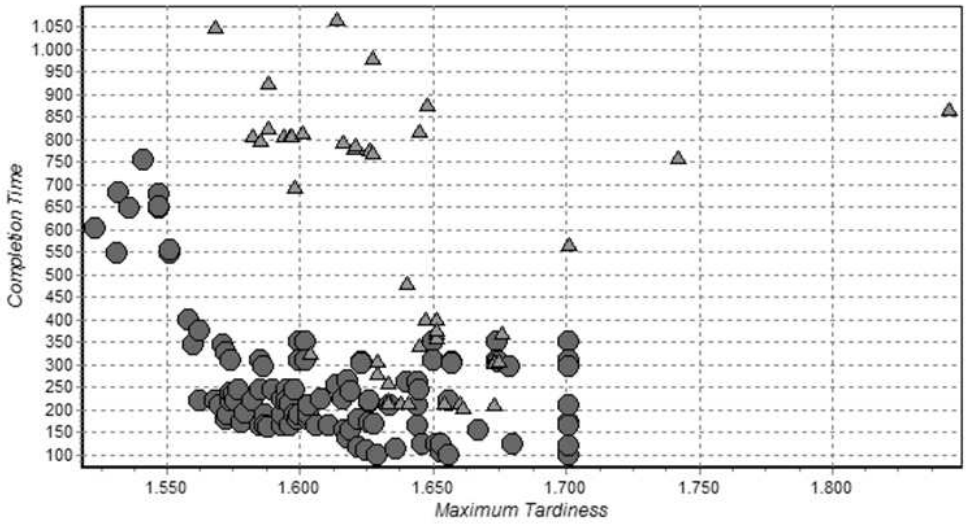


Figure 10 – MHSO (circles) and IMGA (triangles) Pareto borders
221x121mm (88 x 88 DPI)

A Modified Harmony Search Algorithm for multi-objective flowshop scheduling problem with due dates

M. FROSOLINI, M. BRAGLIA and F. A. ZAMMORI

Dipartimento di Ingegneria Meccanica, Nucleare e della Produzione

Facoltà di Ingegneria, Università di Pisa, Via Bonanno Pisano, 25/B, 56126 Pisa, Italy

BRAGLIA Marcello
Dipartimento di Ingegneria Meccanica, Nucleare e della Produzione
Facoltà di Ingegneria, Università degli Studi di Pisa
Via Bonanno Pisano 25/B, 56126 Pisa
Phone: +39 050 913029
Fax: +39 050 913040
E-mail: m.braglia@ing.unipi.it

ZAMMORI Francesco
Dipartimento di Ingegneria Meccanica, Nucleare e della Produzione
Facoltà di Ingegneria, Università degli Studi di Pisa
Via Bonanno Pisano 25/B, 56126 Pisa
Phone: +39 050 913039
Fax: +39 050 913040
E-mail: francesco.zammori@ing.unipi.it

Corresponding author

FROSOLINI Marco
Dipartimento di Ingegneria Meccanica, Nucleare e della Produzione
Facoltà di Ingegneria, Università degli Studi di Pisa
Via Bonanno Pisano 25/B, 56126 Pisa
Phone: +39 050 913039
Fax: +39 050 913040
E-mail: m.frosolini@ing.unipi.it

Abstract words: 160
Article words: 8700

Abstract

This paper presents a *Modified Harmony Search Optimization Algorithm* (MHSO), specifically designed to solve two and three-objectives permutation flowshop scheduling problems, with due dates. To assess its capability, five sets of scheduling problems have been used to compare the MHSO with a known and highly efficient Genetic Algorithm (GA) chosen as benchmark. Obtained results show that the new procedure is successful in exploring large regions of the solution space and in finding a significant number of Pareto non-dominated solutions. For those cases where the exhaustive evaluation of sequences can be applied the algorithm is able to find the whole non-dominated Pareto border, along with a considerable number of solutions that share the same optimal values for the considered optimization parameters.
To validate the algorithm, five sets of scheduling problems are investigated in depth in comparison with the GA. Results obtained by both methods (exhaustive solutions have been provided as well for small sized problems) are fully described and discussed.

Keywords: Permutation flowshop scheduling; Multi-criteria methodologies; Genetic Algorithms; Harmony Search Optimization.

1. Introduction

Flowshop scheduling has attracted many researchers over time since it was firstly proposed by Johnson in 1954 (see, for example, Dannenbring, 1977, Lageweg *et al.*, 1978, Potts, 1980, Osman and Potts, 1989, Pinedo, 1995, Nowicki and Smutnicki, 1996, Carlier and Rebai, 1996, Cheng *et al.*, 1997, Haouari and Ladhari, 2003, Srikanth and Barkha, 2004, Ladhari and Haouari, 2005, Tseng and Lin, 2009) and has been extensively investigated by researchers both with single (refer, for instance, to Campbell, 1970, Ignall and Schrage, 1965, Nawaz *et al.*, 1983) and multi-objective techniques (an extensive review is given in T'Kindt and Billaut, 2001). As it notoriously represents a computationally *NP-hard* problem and exhaustive evaluation applies only to undersized cases (*i.e.*, when the number of job is reasonably small), most of the proposed approaches have focused on the use of optimization or on the adoption of metaheuristic techniques for single objective flowshop scheduling problems (Pan *et al.*, 1997, Stevens *et al.*,1997, Cheng *et al.*, 2001, Grabowski and Wodecki, 2004, Rajendran and Ziegler, 2005, Tasgetiren *et al.*, 2007, Rajkumar *et al.*, 2009, Lin *et al.*, 2009). Literature on the subject is extensive and a comprehensive survey of makespan minimization from early works up to recent approaches of metaheuristics is provided by Hejazi *et al.* (2005), whereas a broad review of the evolution of the available methods over time is presented in Gupta and

Stafford (2006). It is worth noting that the use of multiple criteria enables a more practical solution for the decision makers (T'Kindt and Billaut, 2001). Indeed, single criterion optimizations do not consider the important trade-offs that intrinsically characterize the scheduling problem. Conversely, whether multiple objectives are optimized properly and conjointly, the solution narrows down to a limited subset of optimal/efficient feasible schedules among which the analyst may choose after appropriate considerations on the criteria themselves. Schaffer (1985) firstly introduced multi-criteria genetic algorithms (MCGA) and, since then, some modified MCGA procedures have been proposed to improve both the quality of the solutions and the speed of the search algorithms (Murata *et al.*, 1996, Ponnambalam *et al.*, 2004). Other methods have been applied as well. For example, Gangadharan *et al.* (1994) proposed a Simulated Annealing algorithm for two-criteria scheduling problems. Rajendran (1994) approached the problem of scheduling in flowshop and flowline-based manufacturing cell with the bicriteria of minimizing makespan and total flowtime of jobs. Cao *et al.* (2003) worked on production scheduling problems in manufacturing systems with parallel machine flowshops. The authors developed a mathematical programming model for combined part assignment and job scheduling. The objective was to minimize a weighted sum of production cost and the cost incurred from late product delivery Interesting efforts have been made to adapt Multi-Objective Immune Algorithms (MOIA) to scheduling problems (Tavakkoli-Moghaddam *et al.*, 2007). Khan *et al.* (2007) addressed the problem of minimizing the weighted sum of makespan and maximum tardiness in an *m*-machine flow shop environment using a metaheuristic called Greedy Randomized Adaptive Search Procedure (GRASP). Yandra *et al.* (2007) discussed the application of a genetic algorithm featuring heterogeneous population to solve multi-objective flowshop scheduling problems. Braglia *et al.* (2009) presented a new heuristic for solving the flowshop scheduling problem that aims to minimize makespan and maximum tardiness. In this case, the Technique For Order Preference By Similarity of Ideal Solution (TOPSIS) algorithm is integrated with the Nawaz-Enscore-Ham (NEH) heuristic to generate a set of potential scheduling solutions. All the above mentioned techniques give interesting results, being able to evaluate a significant number of optimal or near-optimal solutions with reasonable computing efforts.

Recently, the Harmony Search Optimization (HSO), an algorithm that mimics the music composition processes to explore the space of the feasible solutions, has been introduced by Geem *et al.* (2001). Although HSO has shown to be effective in most engineering optimization problems (Lee *et al.*, 2004, Geem *et al.*, 2005, Geem, 2006, Kim *et al.*, 2001,Kim *et al.*, 2006), it has not been applied yet to scheduling issues. Owing to this, the present work presents a modified Harmony Search Optimization Algorithm (MHSO) addressed to solve two and three-objective permutation flowshop scheduling problems with due dates. Compared with a known and highly efficient GA (Murata *et al.*, 1996), the new algorithm shows to be successful in exploring large regions of the solution space and in finding a large amount of feasible efficient solutions. For those cases where the exhaustive evaluation of sequences can be applied the algorithm is also able to find the whole non-dominated Pareto frontier in a large number of instances having the same initial data, along with a considerable number of solutions that share the same efficient values for the considered optimization parameters (in the following these will be called "equipollent" solutions).

The paper is organized as follows. Firstly, the multi-objective permutation flowshop scheduling problem is briefly presented and discussed, along with the parameters that have been considered for optimization. Following, the standard HSO and the proposed MSHO are described in detail. Finally five sets of scheduling problems are investigated in comparison with the GA proposed by Ishibuchi and Murata (1998), being the latter one of the most effective algorithms proposed in literature to cope with the subject. Results obtained with both methods (exhaustive solutions have been provided as well for small sized problems) are fully described and discussed.

2. The multi-objective permutation flowshop scheduling problem with due dates

Multi-objective scheduling enables a more practical solution for the decision makers as it allows to consider at one time and in a suitable manner a broader range of decision criteria. However, due to the fact that a solution that optimizes all the considered criteria often does not exist, a new definition of optimality must be introduced. Here, in particular, we refer to the Pareto optimality as defined by T'Kindt and Billaut (2001). In brief, multi-objective flowshop scheduling algorithms (MOFSA) aim to find (all or most of) the sequences that minimize (or maximize, if the case) the whole set of decision criteria or, in other words, the Pareto non-dominated frontier of the solution space. Mathematically speaking, if $f_j(s_i)$ is the fitness function for the single *j-th* criterion to be evaluated for a generic sequence s_i, the problem can be expressed as:

$$min\{f_1(s_i), f_2(s_i), \ldots, f_p(s_i)\} \quad (1)$$

where p is the global number of decision criteria. The sequence s_i is said to dominate another solution s_j if the following applies:

$$\forall p: f_p(s_i) \leq f_p(s_j) \text{ and } \exists h: f_h(s_i) < f_h(s_j) \quad (2)$$

Figure 1 shows dominated and non-dominated solutions for a two-objective scheduling problem.

FIGURE 1 HERE

The standard permutation flowshop scheduling problem with due dates, in particular, can be formally enunciated as follows: a set of n jobs $\{J_1, J_2, \ldots, J_n\}$ has to be processed in sequence on m machines $\{M_1, M_2, \ldots, M_m\}$. Each job consists of m operations $(O_1, O_2, \ldots, O_m)$ and the *j-th* operation of each job must be processed on the *j-th* machine in the sequence. Each operation of a job is characterized by a processing time on each machine, leading to the definition of a n x m Processing Time Matrix (PTM) and by a due date (defining the n-sized Due Dates Vector, DDV).

The optimization criteria that have been used in the present study, in close accordance with Ishibuchi and Murata (1998), are the *makespan* (MS) and the *maximum tardiness* (MT) in the case of two-objective scheduling, being the *total flowtime* (TFT) the last criterion in the case of three-objective problems.

3. Harmony Search Optimization

Recently, *Harmony Search Optimization* (HSO) has been proposed as an algorithm that mimics music composing (Geem *et al.*, 2001). Briefly, it tries to replicate the process that leads musicians to continuously adjust pitches and tunes to produce better harmonies and is based on the concept of an *Harmony Memory* (HM) that represents the acquired know-how of the music player. The creative process is made up of three different stages, during which the musician:

1) draws from the HM to make the most of the past experiences;
2) tries to modify the known melodies by adjusting pitches and changing notes in order to get new compositions;
3) improvises and creates new harmonies from scratch, following inspiration.

If the new melody sounds good (with respect to an aesthetic benchmark) it is inserted within the HM to increase and enhance the available repertoire. In mathematical terms this can be obtained by representing each harmony as a vector made up of exactly n elements, if n is the number of the variables that define the model. Each element within the vector corresponds to a note. The aesthetic benchmark is replaced by an appropriate fitness function (or more, in case of multi criteria optimization) and the problem can be easily brought back to the classical form reported in equation 1 (paragraph 2). Once the *Harmony Memory Size* (HMS) has been defined, the whole HM can be written as a matrix of HMS vectors:

$$\left[\begin{array}{cccc|c} x_1^1 & x_2^1 & \cdots & x_n^1 & f(x^1) \\ x_1^2 & x_2^2 & \cdots & x_n^2 & f(x^2) \\ \vdots & \cdots & \cdots & \cdots & \vdots \\ x_1^{HMS} & x_2^{HMS} & \cdots & x_n^{HMS} & f(x^{HMS}) \end{array}\right]$$

The value of the fitness function(s) for each harmony is (are) included within the HM.

The algorithm (Figure 2) starts by generating HMS random new harmonies. In general, each note (variable) should be chosen within its definition set (valid range) and considering the other notes that have already been inserted in order to avoid violated constraints. This issue can be easily overcome both by checking and adjusting the validity of the vector or by imposing penalties to the violating sequences. At each iteration a new harmony is generated in n successive steps (starting form position 1 to position n, where n is the number of variables) by means of three mechanisms: (*i*) random selection, (*ii*) memory considerations and (*iii*) pitch adjusting.

If the *Harmony Memory Considering Rate* (HMCR) is the rate representing the probability of choosing a note from the memory, the first method involves the random selection of a note from the valid range with a probability of 1 – HMCR. Therefore, a random number is extracted and, if its value is greater than HMCR and less than 1 a note is selected within the definition set:

$$x_i^* \leftarrow x \in X$$

where x_i represents the variable at the current step and X is the above mentioned definition set. On the contrary, if the random value is less than HMCR a vector is randomly selected from the HM and the note corresponding to the current step is introduced within the new harmony:

$$x_i^* \leftarrow x \in \left\{x_i^1, x_i^2, \ldots, x_i^{HMS}\right\}$$

If the constraints are somehow violated it is necessary to check the validity or to introduce penalties. Finally, as the musician modifies some notes to adjust their pitches to the neighboring ones, with the aim of obtaining a smooth and sound composition, so the algorithm requires to alter some variables with a certain probability (this is also referred to as *Pitch Adjusting Rate*, PAR). In practice, a note is randomly selected and its value is changed within the allowed range.

If the new harmony is better than the worst one contained in the HM, with the respect to the fitness function (or referring to the Pareto optimality in the case of multiple objectives), the latter is overwritten to improve the knowledge base. The process is iterated up to the maximum number of iterations or until a particular terminating condition is satisfied.

FIGURE 2 HERE

The algorithm, modified and improved in several manners, has been successfully applied to some interesting engineering problems, such as structural design (Lee and Geem, 2004), water network design (Geem, 2006), traffic routing (Geem *et al.*, 2005) and fluid leakage detection (Kim *et al.*, 2001).

4. A New Harmony Search for multi-objective permutation flowshop scheduling

Since the HSO proved to be extremely performing while coping with some noteworthy engineering problems, it appears to be a good candidate to cope with the multi-objective standard permutation flowshop scheduling problem. The aim of the present work is that of showing its capabilities (both with respect to performances and quality of the solutions) and this is done by comparison with a well known and highly

efficient GA. In particular, the mentioned GA is known in literature since it provides a large number of optimal solutions within short lapses of time.

To enhance the original HSO algorithm capabilities of exploring larger regions of the Pareto border while keeping a good computational speed some modifications have been introduced within the procedure. To begin with, the intrinsic structure of the algorithm remains almost unchanged: notes/variables represent jobs and a whole harmony is a sequence. Since each job can be introduced into a sequence only once, a strict check is carried out on all new harmonies and duplication of jobs within a sequence is not allowed. The MHSO continues to refer to the three classical mechanisms used to generate new harmonies (random selection, memory considerations and pitch adjusting). With respect to the quality of new generated harmonies, the multi-objective algorithm makes use of the known Pareto optimality definition with respect to the *makespan*, the *maximum tardiness* and, in the case of three-objective scheduling, the *total flowtime* criteria. Further, the algorithm introduces new concepts that help finding better solutions and moves a step forward in the direction of emulating the human behavior. A musician who improvises a new composition is influenced by early works (his/her own and those of other composers as well, that constitute the personal knowledge base) and usually tends to make the most of this heritage. The original HSO mimics this behavior by replicating a single note at time from a known harmony to the new one, but it lacks of the capability of introducing and preserving larger portions of a composition. It is worth noting that this aspect represents a notable advantage of genetic algorithms and, therefore, a similar process, named *Large Portion Recovery* (LPR) has been introduced within the HSO, along with the corresponding probability (*Large Portion Recovery Ratio*, LPRR). Practically speaking, this is obtained by performing, from time to time and in accordance to the LPRR, that is usually kept very small, a crossover and a mutation procedure on the best harmonies available within the HM. It is also known that most musicians are able to improvise new compositions for a period, after which they tend to re-use their early works, to take inspiration from the other composers, as creativity inevitably decreases over time, or simply to explore new and different musical genres. Sometimes this determines the flourishing of new inspiration and the overall improvement of their compositions. This aspect strictly resembles what occurs when the HSO, after a generally high number of iterations, becomes unable to explore new sections of the solution space and repeatedly visits the same sequences without further improvements. A solution, namely *Harmony Recovery* (HR), to this particular issue has been proposed and evaluated, based on a large scale LPR procedure. After a given number of iterations, that can be defined and controlled by means of an *Harmony Recovery Ratio* (HRR), the algorithm increases the recovery mechanism probability (acting on the LPRR) and the corresponding crossover activity. Doing so, the algorithms starts to take advantage of the most effective portions of the sequences that have been previously found. This step is preceded by a random regeneration of the HM (up to the 80% of the harmony memory can be updated, but generally it is suitable to update 50% or less of it), aimed to reduce the convergence of a large number of harmonies to a limited subset of the solutions space. To increase the capability of the algorithm of escaping from such unlikable situations, a random regeneration of the HM has also been forced whether the non-dominated memory (the so-called “elite”), that will be introduced and

explained later, has not been updated within a given timeframe (usually a given percentage of the maximum number of iterations).

Consequently, the whole algorithm can be summarized as follows:

1) the HM is randomly generated and the fitness functions are evaluated for each criterion and for each vector (as in the original HSO);
2) the iterative procedure starts and lasts until the terminating conditions, to be defined for the specific problem under study, are met or the maximum number of iterations is reached. In the present work both the GA and the MHSO are stopped when the maximum number of iterations is reached. Since the aim of the study is to compare the two algorithms this allows to evaluate both speed and the relative quality of the solutions;
3) for each iteration the following steps apply:
    a. if the current iteration number is less than a given amount, corresponding to the value of the HRR multiplied for the *Maximum Number of Iterations* (MNI) the HSO process is performed as usual (experience shows that good values for HRR are comprised between 0.6 and 0.8):
        i. a random number is extracted. If it is less than the LPRR (this value is kept very small, such as 0.001) then the LPR procedure is started and two harmonies from the HM are subjected to a two-points crossover operator to generate a new harmony. Indeed, harmonies are mere sequences and can be treated as chromosomes in a generic GA. The classical two-point crossover is here used to generate two new harmonies from two efficient parents: briefly, two random positions are selected within the parents and the corresponding chromosomes sections are plainly exchanged. The new harmony may also be subjected, in accordance to the PAR, to a pitch adjusting procedure as in the ordinary HSO. It is worth noting that this has been reduced to a mutation operator (both insertion and exchange of the elements of a sequence are randomly used), due to the fact that each job cannot be present more than once in each sequence. Also, the PAR may be optionally set linearly increasing between a minimum and a maximum values;
        ii. if the number is greater than LPRR the standard HSO procedure is performed. The worst - or better, an inefficient - harmony (the one that will be replaced by a new better solution) is picked on the basis of a roulette wheel selection process (Goldberg, 1989) rather than on the simple evaluation of the fitness function as in the original HSO. The procedure is based on the fact that the probability of each harmony of being selected and replaced is evaluated on the basis of a weighted fitness function of all criteria (with the same weights proposed by Ishibuchi and Murata (1998), that showed to be very effective). This is obviously greater for inefficient solutions. While on one hand the risk of removing good harmonies is kept very low due to a proper elitism procedure, on the other the algorithm prevents the threat of early convergence. Furthermore, this allows a straight and easy evaluation of the selection probability. Also, a local search procedure is started for every new harmony: the MHSO

looks for a limited number of neighbors (carrying out a simple mutation procedure each time, where both insertion and exchange of the elements of a sequence are randomly used) and, if a better solution emerges (here the Pareto optimality is used to evaluate the quality of the harmony), the original new sequence is overwritten. The previously selected inefficient harmony is finally replaced;

b. on the other hand, if the current iteration number is greater than the value of the HRR multiplied for the Maximum Number of Iterations, the HR procedure is started:
   i. up to 80% of the current population can be randomly generated (and, necessarily, the old one overwritten), to simulate the fact that the musician begins to take inspiration from other composers or that he simply wants to explore new musical genres. This occurs when the above condition is met and allows the algorithm to move from a section of the Pareto border and to explore new portions of the solution space;
   ii. the LPRR is increased up to a maximum of 0.5 (experience shows that values between 0.01 and 0.1 give the best results), and the algorithm goes on as usual;

4) as briefly pinpointed before, at each iteration a proper elitism procedure is applied to the HM. The non-dominated solutions found at every step are kept in a separate memory location that is punctually updated (*i.e.*, solutions that become dominated when better ones are found are removed from the elite). In this case, to determine dominated and non-dominated harmonies, the algorithm refers to the Pareto optimality (as described by equations 1 and 2 of paragraph 2) rather than on the evaluation of the weighted fitness function of all criteria. To take advantage of this non-dominated memory, up to a maximum of 10% of the population size elite harmonies are inserted within the HM with the aim of improving the MHSO knowledge base, without forcing a premature convergence of the whole memory to a limited portion of the solution space (greater percentages, indeed, have shown to be particularly inefficient);
5) if the non-dominated elite has not been updated within 1/6 of the maximum number of iterations (elite update delay) then a random regeneration (up to 80%) of the HM is forced.

The complete MHSO procedure is hereafter summarized in the form of Pascal pseudo-code:

```
generate random Harmony Memory;
set LPRR to 0.0001
while terminating conditions are not met or maximum iterations are not reached do
  if (current iteration < HRR × MNI) or (LPRR = 0.5) then
    extract a random number → Rand
    if Rand < LPRR then
      perform Large Portion Recovery (LPR) procedure to get a new harmony
      extract a random number → Rand
      if Rand < PAR then
        perform Pitch Adjusting (mutation) procedure on the new harmony
```

end if

else

perform standard HSO procedure and get new harmony → NHarmony

select inefficient harmony (roulette wheel selection) → IHarmony

perform local search procedure on NHarmony (in case of improvement overwrite)

if NHarmony improves IHarmony replaces it

end if

else

regenerate up to a maximum of 80% of current population

set LPRR to a maximum of 0.5

end if

perform elitism procedure (based on Pareto optimality) and update non-dominated elite

insert harmonies from elite to current population (up to a maximum of 10% of population size)

if elite not updated within 1/6 of MNI then regenerate up to 80% of current population

end while

It is also schematically reported if Figure 3.

FIGURE 3 HERE

5. Comparative analysis

The MHSO presented in the paper has been compared with the Genetic Algorithm proposed by Ishibuchi and Murata (1998), that in the following will be indicated as Ishibuchi-Murata GA (IMGA). To this aim their formulation has been adopted with respect to:

- the number of jobs and machines;
- the generation of the initial population;
- the selection of the objectives;
- the setting of the operating parameters (that have been optimized as proposed by the authors themselves).

This choice is mainly due to the fact that the above mentioned algorithm is known to be particularly effective in the context of the specific issue under analysis and that it refers both to two and three criteria optimization problems. Moreover, it is also known that the performances of GAs are seldom exceeded or even achieved by other procedures. Hence, it certainly represents a valuable benchmark.

To begin with, the problems that have been examined in depth can be divided into two large sets, characterized by the number of optimization criteria (respectively 2 and 3) and each subdivided into five

subsets with respect to the number of jobs (we assumed, respectively, n = 10, 20, 30, 50 and 100) and machines (we assumed, respectively, m = 5, 10, 15, 20 and 30). This choice is mainly due to the fact that the IMGA has been tested by the authors (Ishibuchi and Murata, 1998) on similar problems (actually, they reported results for the two cases n = 10, m = 5 and n = 20, m = 10) and, therefore, this allows to perform a straightforward comparison. It is noteworthy that the IMGA has been tested with different sets of operating parameters (population size, crossover and mutation probabilities, elitism) with the aim of finding their best combinations before starting the comparative process. However, for smaller problems (n = 10, m = 5 and n = 20, m = 10) the values proposed by the above mentioned authors still showed to be the best ones. For bigger problems (n = 30, m = 15, n = 50, m = 20 and n = 100, m = 30) the best combinations were chosen after a significant number of tests.

The first set refers to a two-objectives optimization, where the adopted criteria are the *makespan* and the *maximum tardiness*, whereas the latter is aimed to a three-objectives optimization, being the *total flowtime* the last criterion. The initial population, the processing times and the due dates have been calculated as proposed in Ishibuchi and Murata (1998) and each set of these data has been used unaltered both when running the MHSO and the IMGA with the aim of allowing an easy comparison of the algorithms. In brief, the processing time of all jobs in a sequence have been specified as random numbers in the interval [1, 99] (or, in other words, they have been extracted from a discrete uniform distribution in the interval [1, 99]), whereas the due date of the single job has been evaluated at first on the basis of a randomly generated sequence. Indeed, given the completion time of the i-th job, namely CTM(i,m), the corresponding due date has been assumed equal to the following:

$$DD(i) = CTM(i,m) + random[-100,100]$$

Also, the fitness functions used for the linear scaling roulette wheel selection strategy and the strategy itself (as formulated by Goldberg, 1989) have been borrowed by the same work. The former is represented by the weighted sum of the adopted criteria, or, concisely:

$$f(s) = -\sum_{i=1}^{k} \omega_i f_i(s)$$

where k is the number of criteria, ω_i is the weight, $f_i(s)$ is the evaluation of the *i-th* criterion for the sequence s and, finally, f(s) is the overall fitness function. In the present work the same weights (constant multipliers) proposed by Ishibuchi and Murata (1998) have been adopted since the variance of the makespan is much smaller than that of the maximum tardiness (and differs from that of the total flowtime too), suggesting a normalization process aimed to handle the scheduling criteria equally. Furthermore, after several tests, they showed to be very effective and sensibly enhanced the whole algorithm. If Ψ designates the whole HM, it is

certainly possible to evaluate its worst solution $f_{worst}(\Psi)$. In the case of a minimization problem this corresponds to the higher value of the fitness function. Hence, the selection probability can be defined as:

$$P(s) = \frac{f(s) - f_{worst}(\Psi)}{\sum_{s \in \Psi}[f(s) - f_{worst}(\Psi)]}$$

This value is used both when selecting two candidate sequences for the crossover required by the Large Recovery Procedure and when a single sequence is picked up to be overwritten by a better solution. In this last circumstance, owing to the fact that it is necessary to substitute the worst candidate with the higher probability, the selection is performed referring to the complementary of the above calculated chance ratio.

5.1 The 10 jobs – 5 machines problem with 2 objectives

A first class of 100 problems has been generated for n = 10 and m = 5, as this is the first test performed by Ishibuchi and Murata. This class is also interesting in that it allows to perform the exhaustive search and, therefore, to refer to a significant benchmark. The GA parameters have been set as follows:

- *PopSize* = 20;
- crossover probability = 0.9;
- mutation probability = 0.3.

The number of non-dominated chromosomes re-inserted within the population at each step has been fixed to 3 and the local search depth to 2. Finally, the Maximum Number of Iterations (MNI) has been put equal to 10000, in order to evaluate *PopSize* x MNI = 200000 chromosomes.
As an example, referring to the processing times and the due dates reported in Table 1 and Table 2, the IMGA required about 2.55 seconds on a Core 2 Duo T7200 processor (2.0 GHz) with 2 GB RAM.

TABLE 1 HERE

TABLE 2 HERE

On average, running 100 times the same problem, the algorithm found 12 non-dominated heuristic Pareto points for a total of 62 sequences, as depicted by small orange triangles in Figure 4. Instead, the exhaustive solution (represented in Figure 4 by diamonds) required about 91.45 seconds to weigh up 10! sequences, found 11 border points for a total of 111 sequences and showed that 4 out of 12 of the border points found by the IMGA were dominated by the optimal border.

FIGURE 4 HERE

The same problem was in turn submitted to a MHSO characterized by the following parameters:

- HMS = 12;
- HMCR = 0.95;
- PAR = linearly increasing between 0.05 and 0.35;
- LPRR = 0.0001;
- HRR = 0.7.

Referring to the maximum number of iterations and considering that the MHSO generates a single sequence at each step, whereas the IMGA generates *PopSize* chromosomes at each iteration, it was decided to set the value equal to *PopSize* × MNI. Actually, due to the fact that the MHSO performs the Harmony Recovery Procedure during the late iterations and that this involves a crossover procedure on the whole HM, it generally would examine a greater number of solutions than IMGA. Therefore, a constraint has been set to force the number of evaluated sequences to coincide with that of the IMGA. The algorithm required about 4.91 seconds to evaluate 200000 sequences and found the whole Pareto border (11 points) corresponding to the exhaustive solution (Figure 5), for a total of 96 sequences (86.5% of the complete solution set).

FIGURE 5 HERE

On average, the IMGA required about 2.5 seconds to evaluate 200000 chromosomes, obtaining 7.9 Pareto points and a total of 51.7 non-dominated sequences. Actually, a simple comparison with the solutions found by the MHSO showed that IMGA was able to find an average of 4.8 non-dominated Pareto points and a total of 42.5 non-dominated sequences, the rest being taken over by the MHSO frontier. Indeed, the MSHO required about 5.13 seconds to discover 7.1 Pareto border points and 69.8 non-dominated sequences. In this case, the solutions always coincided with those found by the exhaustive (except two cases, where the MHSO missed one point) and were never dominated by those found by the IMGA.
To better illustrate these results, a subset constituted by the first 10 trials has been reported in detail in Table 3.

TABLE 3 HERE

Briefly, the PB column reports the Pareto border points (*i.e.*, the non dominated solutions) found both by MHSO and IMGA, whereas the EPP columns shows the "equipollent" non-dominated points (two different sequences can be said "equipollent" if they share the same Pareto Point, or, in other words, if they give the same solutions for all the scheduling criteria). The third column reports the total number of non-dominated sequences (*i.e.*, the sum of PB and EPP) and Time indicates the seconds that the algorithms took to get the solutions. Further, Act PB and Act EP refer to the compared non dominated borders found by both

algorithms. Concisely, they show how many solutions are not dominated by those found by the other algorithm. Finally, the last column indicates whether the sequences coincide with those found by the exhaustive procedure. A second trial (the same sets of data) has been carried out modifying the MNI for the IMGA (this also modifies the maximum number of iterations for the MHSO). To improve the IMGA performances, it was decided to let run the algorithm for exactly 100000 iterations. For instance, with respect to the first problem (extensively discussed above), the algorithm required on average (over 100 runs) 25.91 seconds to evaluate 2000000 chromosomes and discovered all the 11 Pareto points and a total of 81 non-dominated solutions on 111 (72.97%). On the other hand, the MHSO needed about 66.72 seconds to elaborate 2000000 harmonies and to find all the 11 Pareto points and a total of 106 non-dominated sequences on 111 (95.49%). It emerged, however, that the MHSO spent most of the time in verifying the "equipollent" non-dominated memory. Therefore, a test has been performed introducing a constraint on the number of iterations: whereas the MNI for the IMGA has been left unaltered, the MHSO has been forced to stop after 1/3 of the original value. Results confirmed that the algorithm was still able to find the whole Pareto border in most cases, along with a greater number of "equipollent" solutions than the IMGA in less time. Indeed, on average, it found more than the 98% of the "equipollent" solutions evaluated in the previous test in 17.75 seconds. A further test has been performed increasing the IMGA population size to 30 chromosomes, but results did not differ significantly from those described above.

5.2 The 20 jobs – 10 machines problem with 2 objectives

The following step entailed the analysis of a two-objective scheduling with $n = 20$ and $m = 10$. The trial was carried out based on 20 runs. Following the indications given by Ishibuchi and Murata the IMGA settings were left unchanged with respect to the previous test, with the exception of the populations size, fixed to 30, and the MNI that was set to 50000 iterations in order to evaluate 1500000 chromosomes. Referring to the MHSO, only the memory size was changed to 24 harmonies.

As the exhaustive algorithm cannot be applied to this case, in order to compare the quality of the solutions two supplementary tools were used: (*i*)a graphical representation of the non-dominated Pareto frontier size evolving over time (*Non-dominated growing rate*, NDGR) and (*ii*) the Esbensen quality estimation (Esbensen, 1996). The former allows to get a direct and intuitive overview of the ability of the algorithm to find new solutions and the speed with which the non-dominated frontier improves (Figure 6), while the latter evaluates the average quality of the solutions. In addition, it has been introduced since it has been successfully used by Ishibuchi and Murata to assess the quality of the IMGA solutions and due to the fact that it represents a valuable tool to evaluate the quality also in a three-dimensional objective space (*i.e.*, when dealing with 3 objective problems).

FIGURE 6 HERE

Briefly, following Esbensen the quality of a set of non-dominated solution can be estimated by randomly generating a large number (k) of weights and - using these values - by calculating the objective function over the entire non-dominated population. Greater values (smaller in *modulus*) indicate better solution sets. In brief, it is possible to write:

$$Q(S) = \frac{1}{k}\sum_{i=1}^{k} max\left\{-\sum_{j=1}^{m}\omega_j^i f_j(s)\right\}$$

where:

- Q(S) is the quality of the whole set of non-dominated solutions;
- k is the large number of weights sets (in the following k will be assumed equal to 10000);
- m is th number of criteria;
- ω_j^i indicates the weight for the *j-th* criterion within the *i-th* set. Weights are randomly generated and are normalized to 1;
- $f_j(s)$ is the evaluation of the *j-th* criterion for the sequence s.

The IMGA behaved well as usual, being able to find good non-dominated borders. However, the MHSO always outperformed it. On average, over 20 tests, the IMGA found 16.2 Pareto points and a total of 36.7 solutions in 42.83 seconds. On the other hand, the MHSO found 12.65 Pareto points and a total of 188.6 solutions in 65.78 seconds. However, despite of the little advantage with respect to the computing time, the true non-dominated Pareto points discovered by the IMGA were only 2.66 (with a total of non-dominated solutions equal to 1.73), being the remaining dominated by those found by the MHSO. To confirm this tendency, the Esbensen index resulted equal to -718.98 for the IMGA and to -532.45 for the MHSO. Only a few common Pareto points were found during these tests.

To better illustrate these results, a subset constituted by the first 10 trials has been reported in detail in Table 4.

TABLE 4 HERE

The graphical representation of the NDGR shows, on average, that the MHSO finds many non-dominated solutions very early, at the beginning of the run, and, later on, it improves the Pareto border continuously. At a later stage, the algorithm, thanks to the *Harmony Recovery* procedure, begins to explore new border portions finding other good solutions. On the contrary, the IMGA tends to maintain the non-dominated border unchanged after a certain amount of iterations and, often, is unable to improve it anymore. This behavior is clearly visible in Figure 7.

FIGURE 7 HERE

5.3 The 10 jobs – 5 machines and 20 jobs – 10 machines problems with 3 objectives

The second set of tests was aimed to solve the scheduling problem with respect to three objectives. After some runs necessary to set the parameters correctly, both the IMGA and the MHSO were modified only with respect to the population size (respectively, 30 chromosomes and 18 harmonies) as changes in the other parameters proved to be ineffective. To show the results of the 10 jobs – 5 machines problem, a subset constituted by the first 10 runs has been reported in detail in Table 5.

TABLE 5 HERE

The outcomes prove that the MHSO again outperformed the IMGA, being able to find a larger amount of true non-dominated Pareto points. The influence the *Harmony Recovery* procedure was even greater than in the previous cases. It emerges that the MHSO often finds a border that coincides with the one found by the exhaustive procedure, whereas the IMGA rarely identifies more than 50% of it.

The following step involved the analysis of a three-objective scheduling with $n = 20$ and $m = 10$. The test was carried out based on 20 runs. The IMGA settings were left unchanged with respect to the previous test, with the exception of the populations size, fixed to 30, and the MNI that was set to 50000 iterations in order to evaluate 1500000 chromosomes. Referring to the MHSO, only the memory size was increased to 36 harmonies. A subset constituted by the first 10 runs has been reported in detail in Table 6. Corresponding examples of NDGRs are shown in Figure 8.

TABLE 6 HERE

FIGURE 8 HERE

Again, it emerged that the MHSO spent most of the time in verifying the "equipollent" non-dominated memory and, therefore, it took more time to complete the job. A test has been performed introducing a constraint on the number of iterations: whereas the MNI for the IMGA has been left unchanged, the MHSO has been forced to stop after half of the original value. Results confirmed that the algorithm was still able to find the whole Pareto border in most cases, along with a greater number of "equipollent" solutions than the IMGA in less time. The outcomes have been reported in Table 7 and the NDGR corresponding to the first run is shown in Figure 9.

TABLE 7 HERE

FIGURE 9 HERE

Figure 10 shows the Pareto borders (in two dimensions, with respect to the *makespan* and the *tardiness* criteria) for the MHSO and the IMGA found at the end of the first run, that required similar computation times (45.66 and 43.73 seconds respectively). After comparing the two sets of data the whole IMGA border results to be dominated, confirming that even when evaluating half sequences with respect to the IMGA, the MHSO is able to appraise a very good solution.

FIGURE 10 HERE

This is also supported by the average values of the Esbensen index. Indeed, for the IMGA it remains almost unchanged (as obvious, since no modifications were introduced within the algorithm), while for the MHSO it grows a bit (in *modulus*). This means a worse solution with respect to the unconstrained case, but the MHSO still performs better than the IMGA.

5.4 The 30 jobs – 15 machines problem with 2 and 3 objectives

To further verify the algorithm capabilities additional tests were carried out on larger problems. Owing to the fact that Ishibuchi and Murata (1998) do not report data for these sets of problems some efforts were devoted to test the IMGA and to fix the parameters that give the best results before starting the comparative analysis. In particular, it emerged that the most sensible parameter is the population size, whereas the crossover and the mutation probabilities are effective in a limited range around the values proposed by the authors (these were, therefore, maintained almost unchanged).

In the case of $n = 30$ and $m = 15$ the best performance for the IMGA was reached with a population size of 40 chromosomes and 50000 iterations. The crossover and mutation probabilities have been taken equal to 0.9 and 0.35 respectively. The number of non-dominated chromosomes re-inserted within the population at each step has been fixed to 3 and the local search depth to 2 as for smaller problems. Comparison was carried out with a MHSO having a population of 36 harmonies. Also, the MHSO was forced to stop after evaluating 40% of the solutions evaluated by the IMGA, with the aim of making it more performing in terms of computational time (as already stated, after a number of iterations the MHSO spends most of the time in verifying the "equipollent" non-dominated memory and for large problems this penalizes it in terms of time). The test consisted of 100 runs and results showed that the IMGA required on average 23.69 seconds to find 23 non dominated Pareto points and 41 equipollent solutions (the corresponding Esbensen quality estimation was equal to -1468.60). On the other hand, the MHSO found on average 28 Pareto points and 63 equipollent solutions in 18.03 seconds (with an Esbensen estimation of -1432.06). Comparing the two Pareto borders it emerged, however, that the MHSO again outperformed the IMGA: on average the former found 9 Pareto border points and 22 equipollent solutions that the IMGA did not discover or dominate. On the other hand, the latter found 4 Pareto points and 5 equipollent solutions that were not present within the solutions

obtained by the MHSO. The IMGA gave better results (a better Pareto border and the corresponding Esbensen estimation) only 7 times on 100 runs. Results are summarized in Table 8, where ND is the column of the non-dominated solutions, Eqp that of the equipollent non-dominated points for the two algorithms. On the other hand, the Exclusive ND and Eqp represent the non dominated solution that each algorithm did not discover or dominate with respect to the other.

TABLE 8 HERE

The same test was repeated introducing the third criterion and modifying the population size parameter both for the IMGA and the MHSO (respectively, 50 chromosomes and 40 harmonies). The other parameters were left unchanged (after some tests these proved to be the most efficient), with the exception of the MHSO that was forced to stop after evaluating 50% of the solutions evaluated by the IMGA. The MHSO performed even better than in the previous case, being able to find on average 86 non dominated points and 42 equipollent solutions, whereas the IMGA found 92 non dominated points and 56 equipollent solutions. However, comparing the Pareto borders it emerged clearly that on average the former found 51 Pareto border points and 2 equipollent solutions that the IMGA did not discover, while the latter found 9 Pareto points and 5 equipollent solutions that were not present within the solutions obtained by the MHSO. The average Esbensen quality estimations were, respectively, -16905.71 and -17501.43.

5.5 The 50 jobs – 20 machines problem with 2 and 3 objectives

In the case of n = 50 and m = 20 with 2 objectives the best performance was obtained with a population size of 60 chromosomes and 50000 iterations for the IMGA (again, the crossover and mutation probabilities have been taken equal to 0.9 and 0.35 respectively. The number of non-dominated chromosomes re-inserted within the population at each step has been fixed to 3 and the local search depth to 2) and 40 harmonies for the MHSO. Again, the MHSO was forced to stop after evaluating 50% of the solutions evaluated by the IMGA, with the aim of making it less time consuming. The test consisted of 100 runs and results showed that the IMGA required on average 73.12 seconds to find 27 non dominated Pareto points and 21 equipollent solutions (the corresponding Esbensen quality estimation was equal to -2373.04). On the other hand, the MHSO found on average 24 Pareto points and 33 equipollent solutions in 65.71 seconds (with an Esbensen estimation of -2300.85). Comparing the two Pareto borders it emerged again that the MHSO worked better than the IMGA: on average the former found 11 Pareto border points and 31 equipollent solutions that the IMGA did not discover. On the other hand, the latter found 7 Pareto points and 11 equipollent solutions that were not present within the solutions obtained by the MHSO. The IMGA gave better results (a better Pareto border and the corresponding Esbensen estimation) only 5 times on 100 runs.

The runs were repeated for the three objective problem. This time, all parameters were left unchanged with respect to the previous test, as these proved to be the most efficient in a significant number of tests. The

MHSO performed satisfactorily, being able to find on average 31 non dominated points and 67 equipollent solutions, whereas the IMGA found 115 non dominated points and 77 equipollent solutions. Comparing the Pareto borders it emerged that on average the MHSO found 31 Pareto border points and 2 equipollent solutions that the IMGA did not discover or dominate, while the latter found 13 Pareto points and 0 equipollent solutions that were not present within the solutions obtained by the MHSO. The average Esbensen quality estimations were, respectively, -43090.63 and -46445.67.

5.6 The 100 jobs – 30 machines problem with 2 and 3 objectives

Finally, the algorithm was applied to the case of n = 100 and m = 30 with 2 and 3 criteria. Referring to the IMGA the best performance was obtained with a population size of 60 chromosomes and 40000 iterations. Again, the crossover and mutation probabilities have been taken equal to 0.9 and 0.35 respectively. The number of non-dominated chromosomes re-inserted within the population at each step has been fixed to 3 and the local search depth to 2. The same problem was in turn submitted to a MHSO characterized by the following parameters:

- HMS = 40;
- HMCR = 0.65;
- PAR = linearly increasing between 0.05 and 0.15;
- LPRR = 0.0001;
- HRR = 0.85.

The MHSO was forced to stop after evaluating 60% of the solutions evaluated by the IMGA to reduce the computing time. With respect to the two-criteria problem, the test consisted of 100 runs and results showed that the IMGA required on average 98.56 seconds to find 36 non dominated Pareto points and 11 equipollent solutions (the corresponding Esbensen quality estimation was equal to -5123.74). On the other hand, the MHSO found on average 25 Pareto points and 9 equipollent solutions in 109.93 seconds (with an Esbensen estimation of -5068,77). Comparing the two Pareto borders it emerged again that the MHSO worked better than the IMGA: on average the former found 13 Pareto border points and 7 equipollent solutions that the IMGA did not discover. On the other hand, the latter found 5 Pareto points and 10 equipollent solutions that were not present within the solutions obtained by the MHSO. All other points found by the IMGA were dominated by the MHSO border. The IMGA gave better results (a better Pareto border and the corresponding Esbensen estimation) 9 times on 100 runs.

The runs were subsequently repeated for the three objective problem. All parameters were left unchanged with respect to the previous test, as these proved to be the most efficient in a significant number of runs. The MHSO found on average 168 non dominated points and 5 equipollent solutions, whereas the IMGA found 143 non dominated points and 2 equipollent solutions. However, comparing the Pareto borders it emerged that on average the MHSO found 76 Pareto border points and 1 equipollent solutions that the IMGA did not

discover or dominate, while the latter found 39 Pareto points and 0 equipollent solutions that were not present within the solutions obtained by the MHSO. The average Esbensen quality estimations were, respectively, -162157.12 and -169322.37. For such large problems, during the tests the IMGA showed to be somewhat faster than the MHSO. Indeed, the IMGA required on average about 102 seconds to run, whereas the MHSO, evaluating 60% of the solutions calculated by the IMGA, needed about 123 seconds to complete. Results of this test are summarized in Table 9.

5.7 Kruskal-Wallis hypothesis test for homogeneity

To complete the validation of the algorithm, a Kruskal-Wallis hypothesis test for homogeneity (Law and Kelton, 1991) was performed to show that the non dominated Pareto border found by the MHSO outperforms the IMGA. The test was applied to the whole set of problems and showed that the MHSO obtained efficient solutions. Fox example, in the case of the 50 jobs, 20 machines and two criteria problem based on 50 runs (*i.e.*, 50 different problems) for each algorithm, the statistic (T) was evaluated for two independent samples (k = 2) and at a level α = 0.99. Results clearly showed that the null hypothesis can be rejected, being T = 28.77 and $\chi^2_{2-1,1-0.99}$ = 6.635 (where $\chi^2_{k-1,1-\alpha}$ is the upper 1 – α critical value for a Chi-square distribution with k – 1 degrees of freedom). When the test was performed with respect to the 100 jobs, 30 machines problem there was evidence that the null hypothesis can be rejected again, being T = 21.15. The difference between T and $\chi^2_{k-1,1-\alpha}$ is even greater for smaller problems. Consequently, the Pareto border found by the MHSO is actually different from the one found by the IMGA and, considering the Esbensen quality estimations, it is possible to affirm that the modified Harmony Search behaved better that the Genetic Algorithm.

Conclusions

The paper has presented a modified Harmony Search Optimization (MHSO) algorithm applied to multi-criteria permutation flowshop scheduling problems with due dates. Also, the results of a comparative analysis with respect to the Genetic Algorithm (GA) proposed by Ishibuchi and Murata (1998) are presented, with the aim of showing its remarkable performances. This choice is due to the fact that the mentioned GA is known to be particularly effective in the solution of the above mentioned problem. It is also known that in similar situations the performances of GAs are seldom exceeded or even achieved by other procedures representing, therefore, a valuable benchmark.

The problems that have been examined in depth can be divided into two large sets, characterized by the number of optimization criteria (respectively 2 and 3) and each subdivided into two subsets with respect to the number of jobs (respectively, n = 10, 20, 30, 50 and 100) and machines (we assumed, respectively, m =

5, 10, 15, 20 and 30). The first set refers to a two-objectives optimization, where the adopted criteria are the *makespan* and the *maximum tardiness*, whereas the latter is aimed to a three-objectives optimization, being the *total flowtime* the third criterion.

The outcomes show that the MHSO outperforms the IMGA, being able to find a larger amount of true non-dominated Pareto points. When applied to small sized problems (n = 10), the exhaustive evaluation has shown that the MHSO often finds the whole border. As the exhaustive algorithm cannot be applied to larger problems, in order to compare the quality of the solutions two supplementary tools have been introduced: (*i*)a graphical representation of the non-dominated Pareto frontier size evolving over time and (*ii*) the Esbensen quality estimation (Esbensen, 1996). To complete the validation of the algorithm a Kruskal-Wallis hypothesis test for homogeneity has been performed to show that the non dominated Pareto borders found by the MHSO actually outperform those found by the IMGA.

Due to the fact that the MHSO takes more time than the GA when evaluating the same number of sequences, constraints have been introduced to the number of iterations of the MHSO to verify its performances even in unfavorable situations. Results clearly show the validity of the proposed algorithm in that it still outperforms the GA in several tests.

With respect to potential future works, the obtained results suggest to enhance the neighborhood search procedure to speed up the algorithm and to adapt it to other problems such as, for instance, layout optimization and network analysis (in particular Hub-and-Spoke problems). Also, it would be useful to develop the algorithm integrating it with other metaheuristic methods (GAs, Simulated Annealing, *etc.*) in an attempt to further improve its overall performance.

References

Braglia, M., Grassi, A., (2009), A new heuristic for the flowshop scheduling problem to minimize makespan and maximum tardiness, *International Journal of Production Research*, 47(1), 273-288.

Campbell, H.G., Dudek R.A., Smith M.L., 1970, A heuristic algorithm for the n-job m-machine sequencing problem, *Management Science*, 16B, 630-637.

Cao, D., Chen, M., 2003, Parallel flowshop scheduling using Tabu search, *International Journal of Production Research*, 41(13), 3059-3073.

Carlier, J., Rebai, I., 1996, Two branch-and-bound algorithms for the permutation flow shop problem, *European Journal of Operational Research*, 90, 238-251.

Cheng, J., Kise, H., Matsumoto, H., 1997, A branch-and-bound algorithm with fuzzy inference for a permutation flow shop scheduling problem, *European Journal of Operational Research*, 96, 578-590.

Cheng, J., Steiner, G., Stephenson, P., 2001, A computational study with a new algorithm for the three-machine permutation flow-shop problem with release time, *European Journal of Operational Research*, 130, 559-575.

Dannenbring, D.G., 1977, An evaluation of flow shop sequencing heuristics, *Management Science*, 23, 1174-1182.

Esbensen, H., 1996, *Defining solution set quality*, Elect. Res. Lab., College Eng., University of California, Berkeley, Memorandum, UCB/ERL M96/1.

Grabowski, J., Wodecki, M., 2004, A very fast Tabu Search algorithm for the permutation flow shop problem with makespan criterion, *Computers and Operations Research*, 31, 1891-1909.

Gangadharan, R., Rajendran, C., 1994, A simulated annealing heuristic for scheduling in a flowshop with bicriteria, *Proc. Of the 16 th International Conference on Computers and Industrial Engineering*, 345-348.

Geem, Z.W., Kim, J.H., Loganathan, G.V., 2001, A new heuristic optimization algorithm: Harmony Search, *Simulation*, 76(2), 60-68.

Geem, Z.W., Lee, K.S., Park, Y., 2005, Application of harmony Search to vehicle routing, *American Journal of Applied Sciences*, 2(12), 1552-1557.

Geem, Z.W., 2006, Optimal cost design of water distribution networks using Harmony Search, *Engineering Optimization*, 38(3), 259-280.

Goldberg, D.E., 1989, *Genetic Algorithms in Search, Optimization and Machine Learning*, Reading, MA, Addison-Wesley.

Gupta, J.N.D., Stafford, Jr.E.F., 2006, Flowshop scheduling research after five-decades, *European Journal of Operational Research*, 169, 699-711.

Haouari, M., Ladhari, T., 2003, A branch-and-bound-based local search method for the flow shop problem, *Journal of the Operational Research Society*, 54, 1076-1084.

Hejazi, S.R., Saghafian, S., (2005), Flowshop-scheduling problems with makespan criterion: a review, *International Journal of Production Research*, 43(14), 2895-2929.

Ignall, E., Schrage, L.E., 1965, Application of branch and bound technique to some flow-shop problems, *Operations Research*, 13(3), 400-412.

Ishibuchi, H., Murata, T., 1998, A multi-objective genetic local search algorithm and its application to flowshop scheduling, *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and reviews*, 28(3), 392-403.

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

Khan, B.S.H., Prabhaharan, G., Asokan, P., 2007, A grasp algorithm for m-machine flowshop scheduling problem with bicriteria of makespan and maximum tardiness, *International Journal of Computer Mathematics*, 84(12), 1731-1741.

Kim, J.H., Geem, Z.W., Kim, E.S., 2001, Parameter estimation of the nonlinear Muskin-gum model using Harmony Search, *Journal of the American Water Resources Association*, 37(5), 1131-1138.

Kim, S.H., Yoo, W.S., Oh, K.J., Hwang, I.S., Oh, J.E., 2006, Transient analysis and leakage detection algorithm using GA and HS algorithm for a pipeline system, *Journal of Mechanical Science and Technology*, 20(3), 426-434.

Ladhari, T., Haouari, M., 2005, A computational study of the permutation flow shop problem based on a tight lower bound, *Computers and Operations Research*, 32(7), 1831-1847.

Lageweg, B.J., Lenstra, J.K., Rinnooy Kan A.H.G., 1978, A general bounding scheme for the permutation flow-shop problem, *Operations Research*, 26, 53-67.

Law, A.M. and Kelton, W.D., *Simulation Modeling and Analysis*, McGraw-Hill, New York, NY, 1991.

Lee, K.S., Geem, Z.W., 2004, A new structural optimization method based on the Harmony Search algorithm, *Computers and Structures*, 82(9 – 10), 781-798.

Lin, S.W:, Gupta, J.N.D., Ying, K.C., Lee, Z.J., 2009, Using simulated annealing to schedule a flowshop manufacturing cell with sequence-dependent family setup times, *International Journal of Production Research*, 47(12), 3205-3217.

Murata, T., Ishibuchi, I., Tanaka, H., 1996, Multi-objective Genetic Algorithm and its applications to flowshop scheduling, *Computers and Industrial Engineering*, Vol. 30, No. 4, 957-968.

Nawaz, M., Enscore, Jr.E.E., Ham, I., 1983, A heuristic algorithm for the m-machine, n-job flow shop sequencing problem, *Omega*, 11, 91-95.

Nowicki, E., Smutnicki, C., 1996, A fast tabu search algorithm for the flow shop problem, *European Journal of Operational Research*, 91, 160-175.

Osman, I.H., Potts, C.N., 1989, Simulated annealing for permutation flow-shop scheduling, *Omega*, 17, 551-557.

Pan, J.C.H, Fan, E.T, 1997, Two-machine flowshop scheduling to minimize total tardiness, *International Journal of Systems Science*, 28(4), 405-414.

Pinedo, M., *Scheduling: theory, algorithms, and systems*, Englewood Cliffs, NJ, Prentice-Hall, 1995.

Ponnanbalam, S.G., Jagannathan, H., Kataria, M., Gadicherla, A., 2004, A TSP-GA multi objective algorithm for the flowshop scheduling, *International Journal of Advanced Manufacturing Technology*, 23, 909-915.

Potts, C.N., 1980, An adaptive branching rule for the permutation flow-shop problem, *European Journal of Operational Research*, 5, 19-25.

Rajendran, C., 1994, A heuristic for scheduling in flowshop and flowline-based manufacturing cell with multi-criteria, International Journal of Production Research, 32(11), 2541-2558.

Rajendran, C., Ziegler, H., 2005, Two ant-colony algorithms for minimizing total flowtime in permutation flowshops, *Computers and Industrial Engineering*, 48, 789-797.

Rajkumar, R., Shahabudeen, P., 2009, An improved genetic algorithm for the flowshop scheduling problem, *International Journal of Production Research*, 47(1), 233-249.

Schaffer, J.D., 1985, Multiple objective optimization with vector evaluated genetic algorithms, *Proc. Of the 1 th ICGA*, 93-100.

Srikanth, K.I., Barkha, S., 2004, Improved genetic algorithm for the permutation flowshop scheduling problem, *Computers and Operations Research*, 31(4), 593-606.

Stevens, J.W., Gemill, D.D., (1997), Scheduling a two-machine flowshop with travel times to minimize maximum lateness, *International Journal of Production Research*, 35(1), 1-15.

Tasgetiren, M.F., Liang, Y.C., Sevkli, M., Gencylmaz, G., 2007, A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem, *European Journal of Operational Research*, 177, 1930-1947.

Tavakkoli-Moghaddam, R., Rahimi-Vahed, A., Hossein Mirzaei, A., 2007, Solving a multi-objective no-wait flowshop problem by a hybrid multi-objective immune algorithm, *Multiprocessor Scheduling: Theory and Applications*, Itech Education and Publishing, Vienna, Austria.

T'Kindt, V., Billaut, J.C., 2001, Multicriteria scheduling problems: a survey, *Operations Research*, 35, 143-163.

Tseng, L.Y., Lin, Y.T., 2009, A hybrid genetic local search algorithm for the permutation flowshop scheduling problem, *European Journal of Operational Research*, 198(1), 84-92.

Yandra, Tamura, H., 2007, A new multiobjective genetic algorithm with heterogeneous population for solving flowshop scheduling problems, *International Journal of Computer Integrated Manufacturing*, 20(5), 465-477.

Table captions

	Machine 1	*Machine 2*	*Machine 3*	*Machine 4*	*Machine 5*
Job 1	85	31	3	84	18
Job 2	98	8	94	69	7
Job 3	26	75	30	24	43
Job 4	18	10	9	25	11
Job 5	44	13	96	45	7
Job 6	42	10	50	39	87
Job 7	96	42	81	13	28
Job 8	15	84	33	27	4
Job 9	96	39	14	97	93
Job 10	49	72	83	81	87

Table 1 – Processing times for a 10 jobs – 5 machines problem

Due 1	92
Due 2	258
Due 3	321
Due 4	432
Due 5	599
Due 6	666
Due 7	772
Due 8	852
Due 9	928
Due 10	928

Table 2 – Due dates for a 10 jobs – 5 machines problem

MHSO							
PB	EPP	Tot	Time	Act PB	Act EP	%	
11	83	94	3,91	11	83	84,68%	Y
10	11	21	4,34	10	11	100,00%	Y
6	15	21	5,12	6	15	100,00%	Y
6	233	239	4,98	6	233	32,61%	Y
5	76	81	7,34	5	76	60,45%	Y
8	11	19	3,23	8	11	95,00%	N
8	2	10	4,12	8	2	100,00%	Y
4	88	92	5,25	4	88	46,94%	Y
7	35	42	3,62	7	35	68,85%	Y
7	68	75	5,22	7	68	57,69%	Y
7,20	62,20	69,40	4,71	7,20	62,20	0,75	

Exhaustive	
PB	Tot
11	111
10	21
6	21
6	733
5	134
9	20
8	10
4	196
7	61
7	130

IMGA							
PB	EPP	Tot	Time	Act PB	Act EP	%	
9	25	34	2,51	8	25	29,73%	N
13	5	18	2,46	6	5	52,38%	N
6	9	15	2,36	5	9	66,67%	N
8	145	153	2,56	4	145	20,33%	N
5	61	66	2,41	5	61	49,25%	Y
7	5	12	2,47	2	3	25,00%	N
8	1	9	2,52	5	1	60,00%	N
3	70	73	2,52	3	70	37,24%	N
8	21	29	2,45	5	21	42,62%	N
7	54	61	2,53	6	54	46,15%	N
7,40	39,60	47,00	2,48	4,90	39,40	0,43	

Table 3 – Outcomes of 10 runs for a 10 jobs – 5 machines – 2 objectives problem

MHSO							
PB	EPP	Tot	Time	Act PB	Act EP	Esbensen	Common
14	15	29	67,08	10	21	-549,70	2
15	75	90	65,80	15	75	-521,37	0
20	291	311	66,59	20	291	-742,65	0
14	8	22	70,81	13	18	-668,94	1
6	38	44	65,48	6	38	-309,03	0
6	976	982	65,68	6	976	-405,84	0
12	280	292	65,09	12	280	-554,89	0
9	43	52	65,97	9	43	-555,69	0
11	14	25	66,02	11	14	-494,78	0
20	40	60	66,73	20	40	-477,93	0
12,70	178,00	190,70	66,53	12,20	179,60	-528,08	

IMGA						
PB	EPP	Tot	Time	Act PB	Act EP	Esbensen
14	15	29	43,68	2	0	-673,75
14	12	26	42,58	2	0	-751,94
20	23	43	42,62	2	0	-839,73
9	10	19	42,25	6	7	-756,75
11	5	16	53,61	0	0	-655,87
12	61	73	42,01	0	0	-656,41
16	26	42	42,83	3	6	-711,91
17	30	47	42,92	4	0	-736,60
7	13	20	43,16	2	6	-664,06
21	7	28	43,12	3	0	-784,04
14,10	20,20	34,30	43,88	2,40	1,90	-723,11

Table 4 – Outcomes of 10 runs for a 20 jobs – 10 machines – 2 objectives problem

MHSO							Common	Exhaustive	
PB	EPP	Tot	Time	Act PB	Act EP	%		PB	Tot
119	3	122	5,50	119	3	96,83%	21	123	126
85	1	86	4,44	85	1	87,76%	33	96	98
34	1	35	4,19	34	1	83,33%	12	40	42
41	0	41	5,66	41	0	97,62%	16	42	42
52	0	52	5,03	52	0	100,00%	21	52	52
47	0	47	4,41	47	0	100,00%	6	47	47
27	0	27	5,01	27	0	93,10%	14	29	29
9	0	9	7,11	9	0	100,00%	4	9	9
49	0	49	4,34	49	0	94,23%	8	51	52
33	0	33	4,39	33	0	100,00%	17	33	33
49,60	0,50	50,10	5,01	49,60	0,50	0,95			

IMGA						
PB	EPP	Tot	Time	Act PB	Act EP	%
87	0	87	2,83	21	0	16,67%
54	1	55	2,81	34	1	35,71%
27	0	27	2,59	16	0	38,10%
30	0	30	2,58	16	0	38,10%
36	0	36	2,69	21	0	40,38%
27	0	27	2,62	7	0	14,89%
23	2	25	2,78	14	0	48,28%
9	0	9	2,52	4	0	44,44%
43	0	43	2,78	11	0	21,15%
30	4	34	2,64	17	0	51,52%
36,60	0,70	37,30	2,68	16,10	0,10	0,35

Table 5 – Outcomes of 10 runs for a 10 jobs – 5 machines – 3 objectives problem

MHSO							
PB	EPP	Tot	Time	Act PB	Act EP	Esbensen	Common
96	7	103	89,27	96	7	-1384,46	0
63	0	63	89,83	63	0	-792,25	0
111	0	111	88,33	111	0	-988,98	0
190	0	190	90,39	190	0	-982,77	0
77	1	78	88,86	77	1	-972,37	0
91	1	92	88,19	91	1	-733,55	1
184	4	188	90,80	184	4	-1078,09	0
157	6	163	88,20	157	6	-1149,36	0
97	0	97	88,69	97	0	-685,71	0
82	1	83	89,78	82	1	-1038,82	0
114,80	2,00	116,80	89,23	114,80	2,00	-980,64	

IMGA						
PB	EPP	Tot	Time	Act PB	Act EP	Esbensen
61	3	64	45,3	3	3	-3848,02
31	0	31	43,83	5	0	-3683,66
71	0	71	44,59	8	0	-3728,62
101	0	101	45,92	4	0	-3657,03
52	0	52	44,16	0	0	-3934,26
42	0	42	43,94	2	0	-3580,59
51	0	51	44,41	2	0	-4041,30
67	0	67	45,19	0	0	-3844,47
46	0	46	45,2	1	0	-3839,29
51	0	51	44,41	3	0	-3641,25
57,30	0,30	57,60	44,70	2,80	0,30	-3779,85

Table 6 – Outcomes of 10 runs for a 20 jobs – 10 machines – 3 objectives problem

MHSO							
PB	EPP	Tot	Time	Act PB	Act EP	Esbensen	Common
108	1	109	45,66	108	1	-739,73	0
82	0	82	44,86	82	0	-969,34	0
117	3	120	44,50	117	3	-1446,53	0
95	21	116	45,81	95	21	-1392,46	0
66	0	66	45,01	66	0	-856,68	0
66	2	68	45,05	66	2	-931,56	0
122	0	122	44,98	122	0	-927,24	0
61	0	61	46,09	61	0	-1203,56	0
54	0	54	45,11	54	0	-685,18	0
122	1	123	46,02	122	1	-1000,75	0
89,30	2,80	92,10	45,31	89,30	2,80	-1015,30	

IMGA						
PB	EPP	Tot	Time	Act PB	Act EP	Esbensen
45	0	45	43,73	0	0	-3730,77
71	0	71	44,94	25	0	-3911,00
72	1	73	46,78	4	0	-3745,36
80	3	83	46,16	20	2	-3759,68
43	0	43	47,28	2	0	-3666,02
36	0	36	44,77	4	0	-3693,97
51	0	51	45,17	0	0	-3623,60
38	0	38	44,48	4	0	-3877,27
37	0	37	44,59	0	0	-3564,01
92	0	92	45,75	0	0	-4132,84
56,50	0,40	56,90	45,37	5,90	0,20	-3770,45

Table 7 – Outcomes of 10 runs for a 20 jobs – 10 machines – 3 objectives problem with constraints on the MHSO iterations

	ND	Eqp	Esbensen	Exclusive ND	Exclusive Eqp
MHSO	28	63	-1458,95	9	22
IMGA	23	41	-1482,00	4	5

Table 8 – Average outcomes of 100 runs for a 30 jobs – 15 machines – 2 objectives problem

	ND	Eqp	Esbensen	Exclusive ND	Exclusive Eqp
MHSO	168	5	-162157,12	76	1
IMGA	143	2	-169322,37	39	0

Table 9 – Average outcomes of 100 runs for a 100 jobs – 30 machines – 3 objectives problem

Figure captions

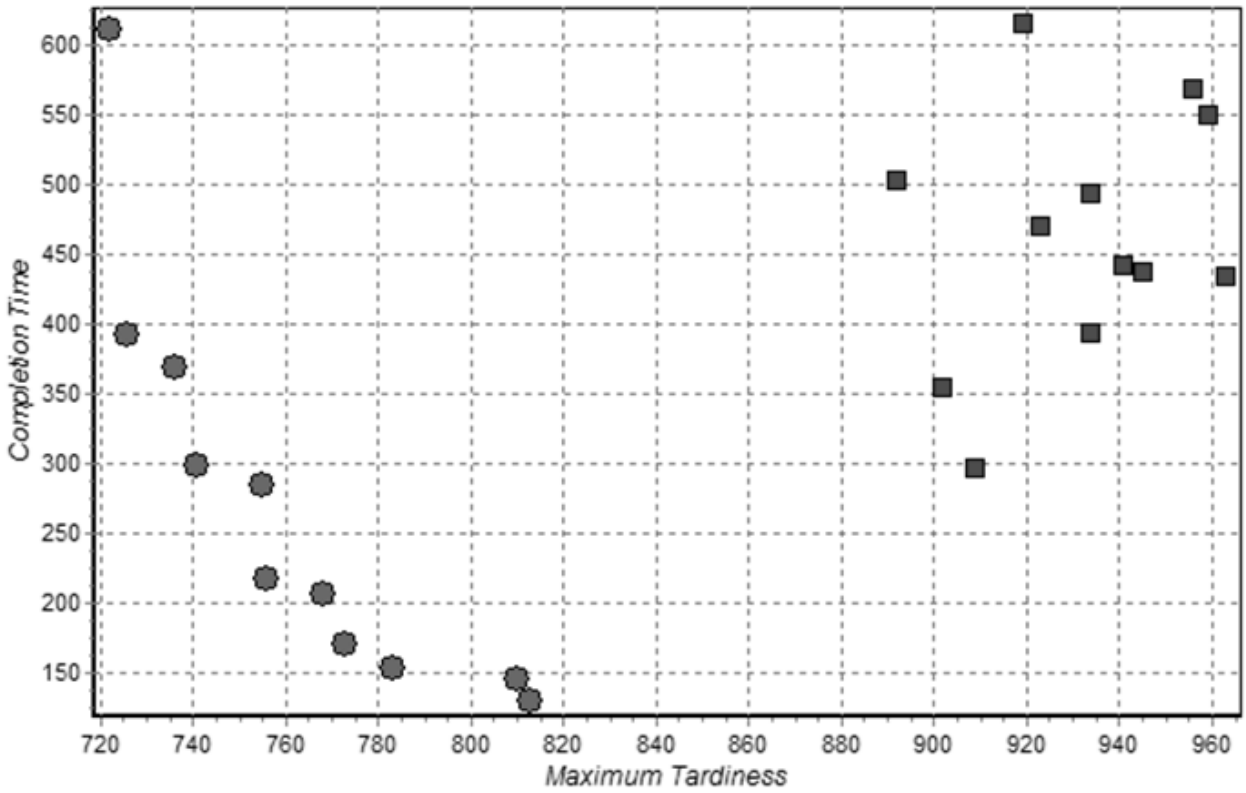


Figure 1 – Dominated (square points) and non-dominated (rounded points) solutions

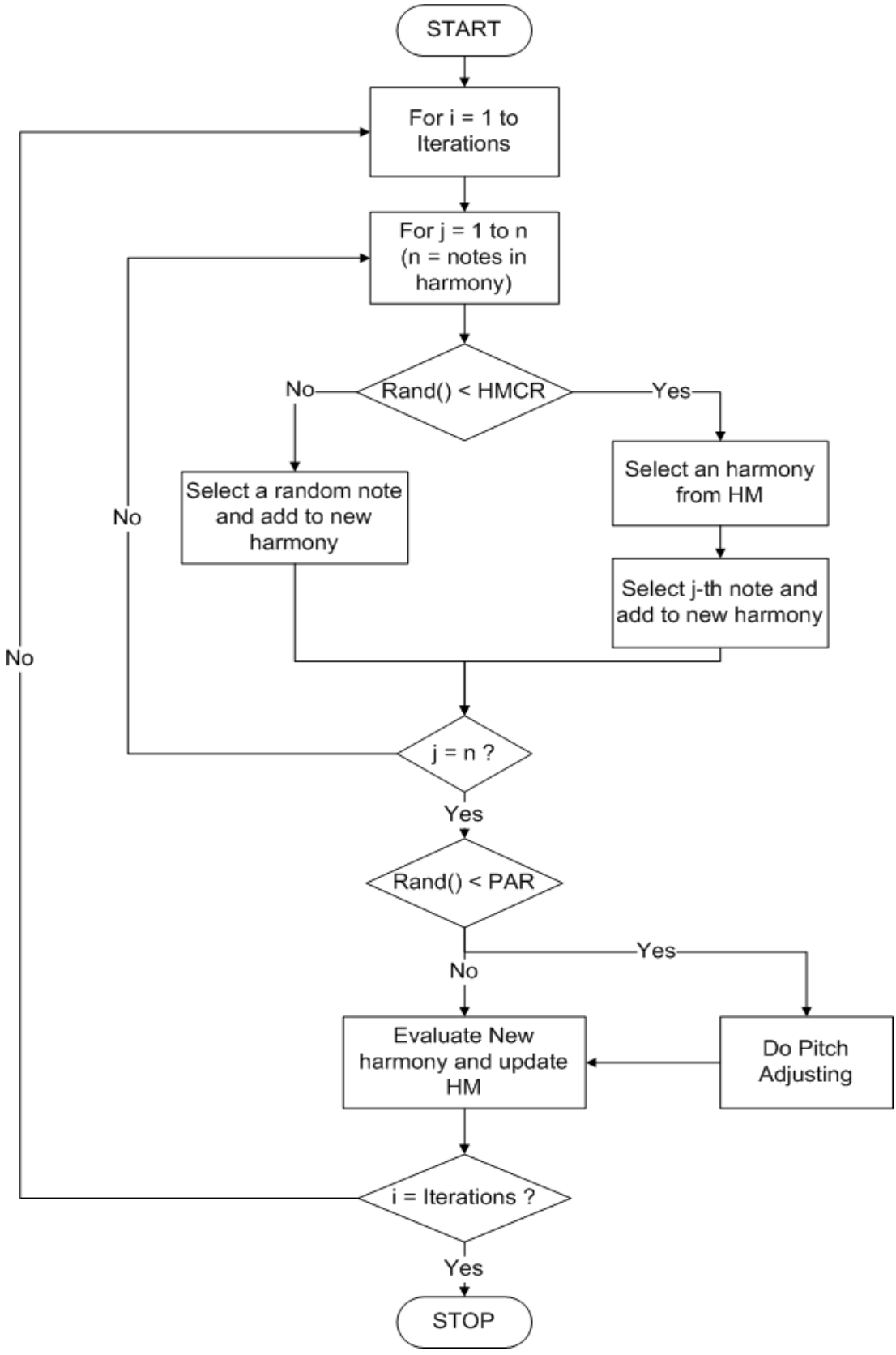


Figure 2 – HSO algorithm flow diagram

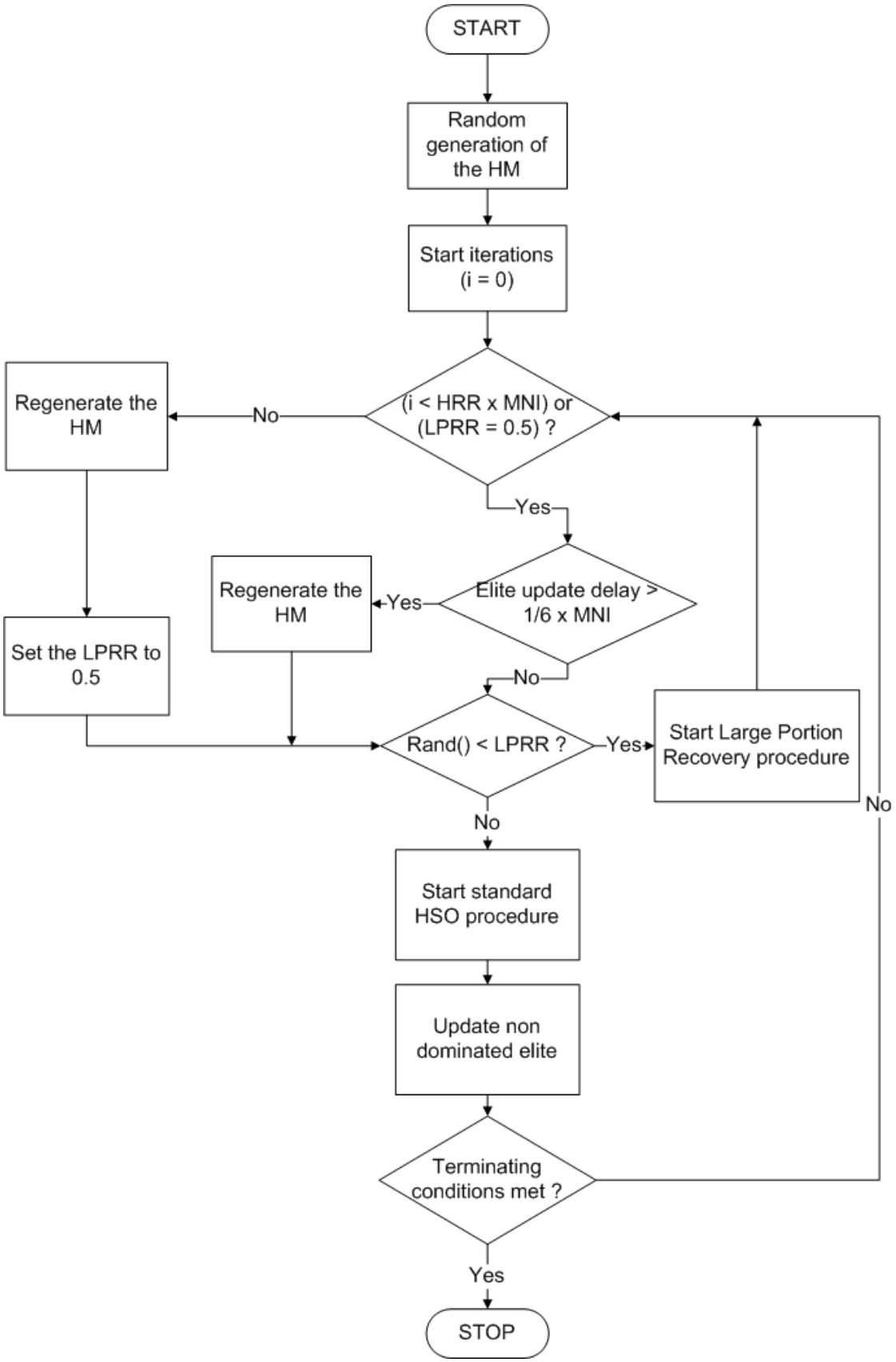


Figure 3 – MHSO algorithm flow diagram

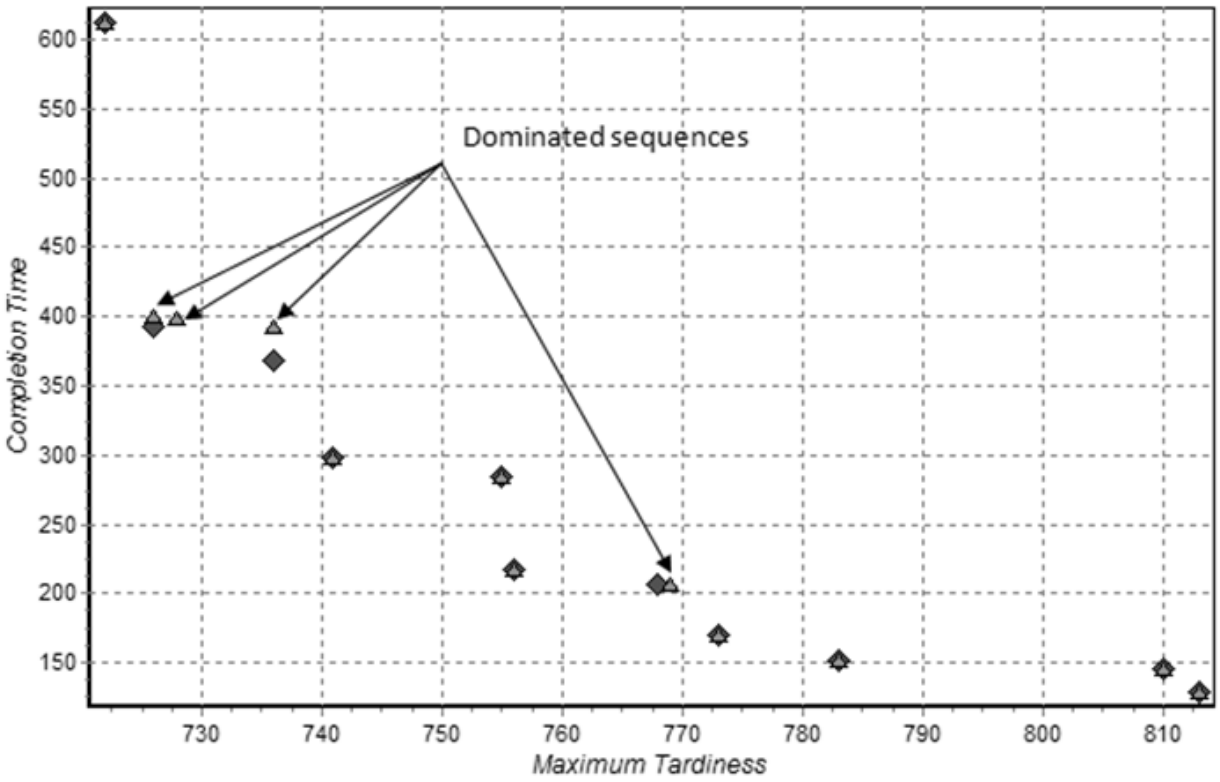


Figure 4 – IMGA and Exhaustive Pareto non-dominated frontier for a two-objective problem

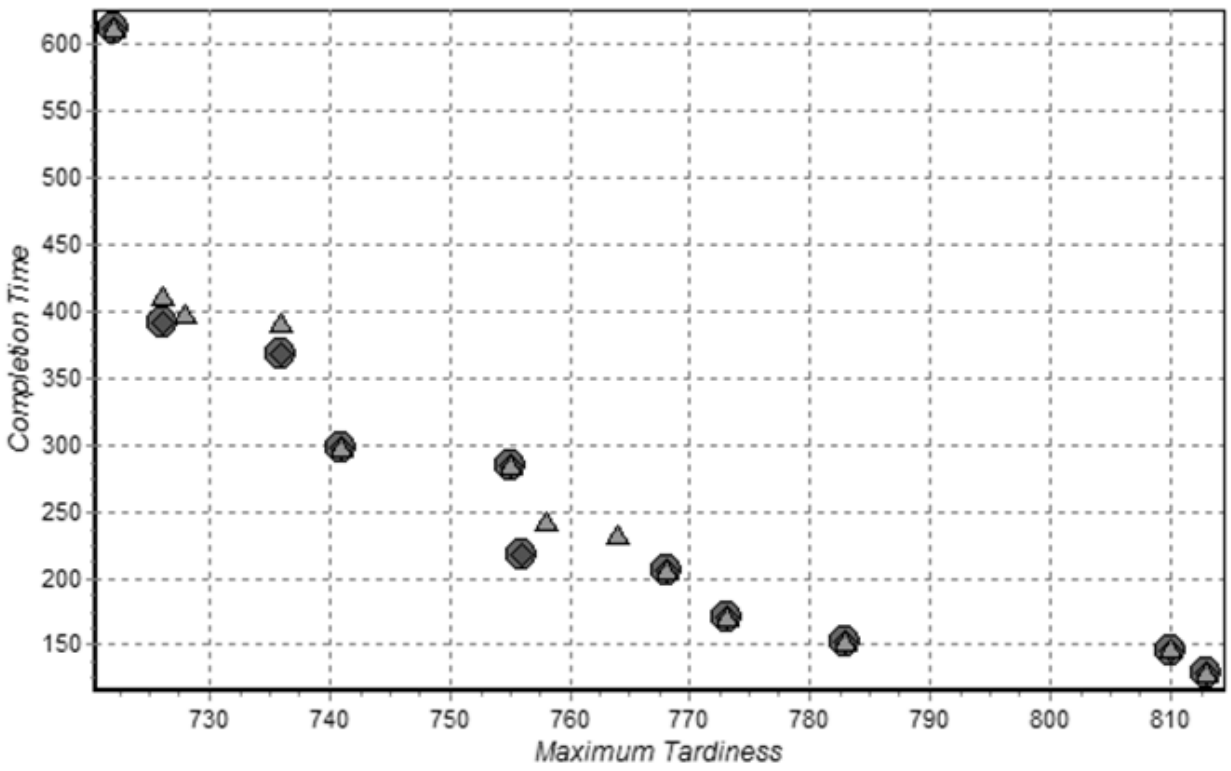


Figure 5 – IMGA (triangles), MHSO (circles) and exhaustive Pareto (diamonds) non-dominated frontier for a two-objective problem

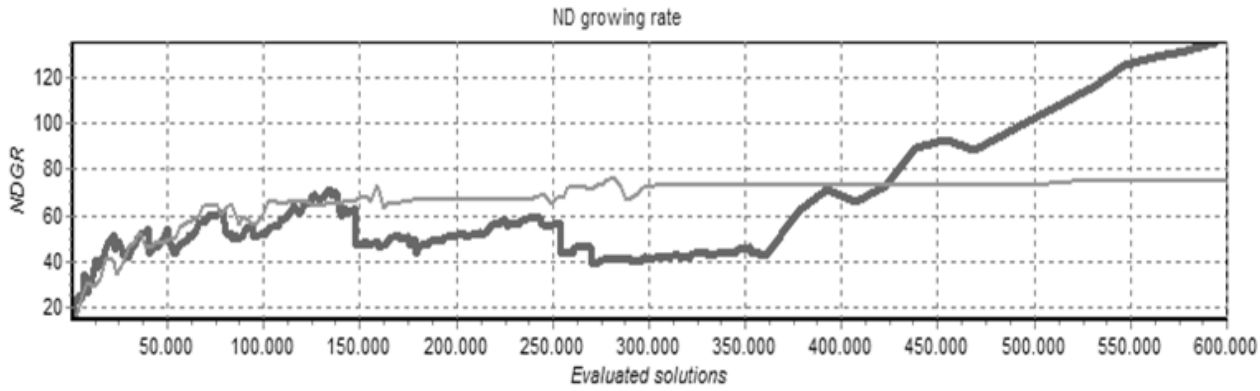


Figure 6 – NDGR for the MHSO (thick line) and for the IMGA in a 20 jobs – 10 machines – 2 objectives problem

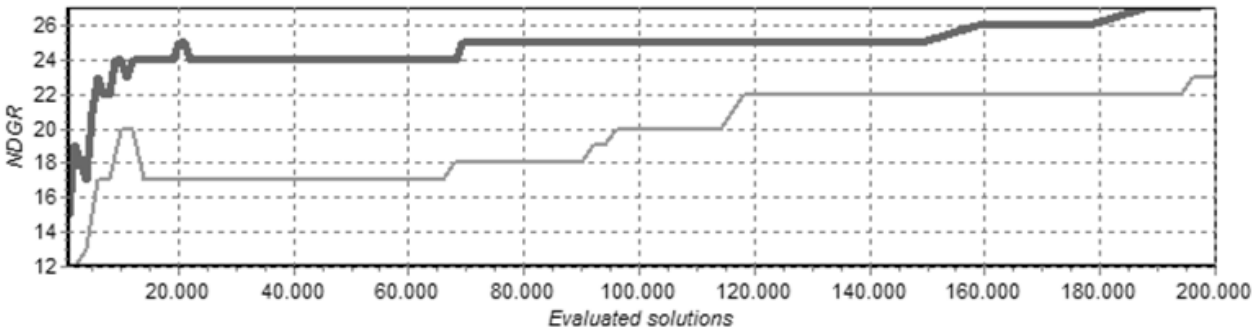


Figure 7 – NDGR for the MHSO (thick line) and for the IMGA in a 20 jobs – 10 machines – 2 objectives problem

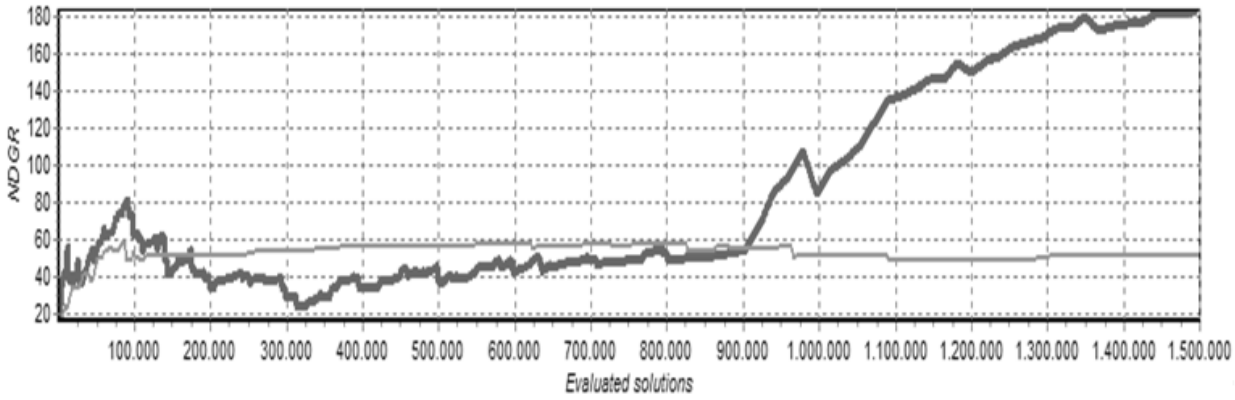


Figure 8 – Example of NDGR for the MHSO (thick line) and for the IMGA in a 20 jobs – 10 machines – 3 objectives problem

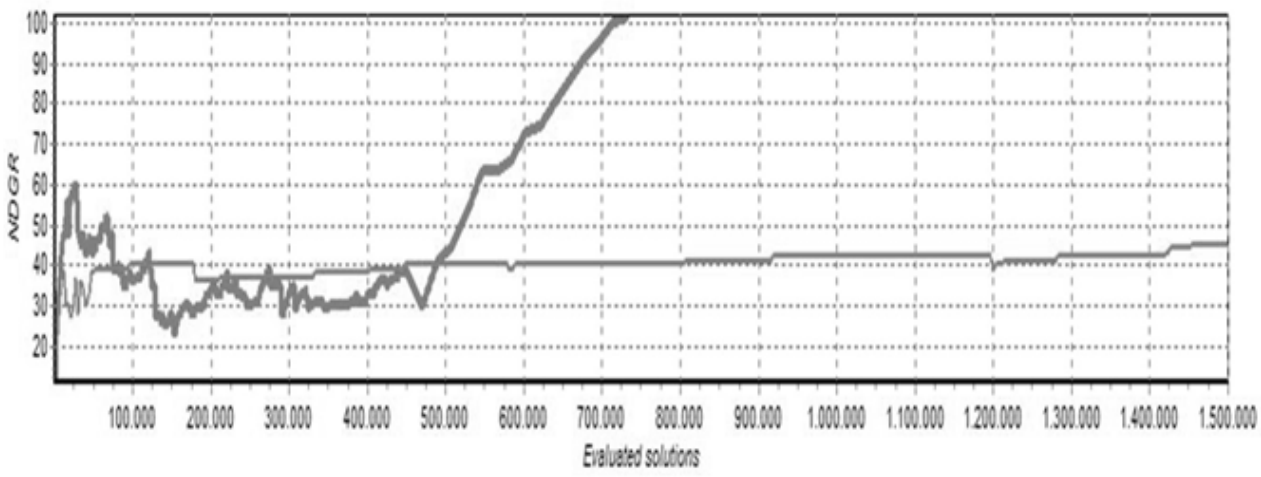


Figure 9 – NDGR for the MHSO (thick line) and for the IMGA in a 20 jobs – 10 machines – 3 objectives problem with constraints on the MHSO iterations

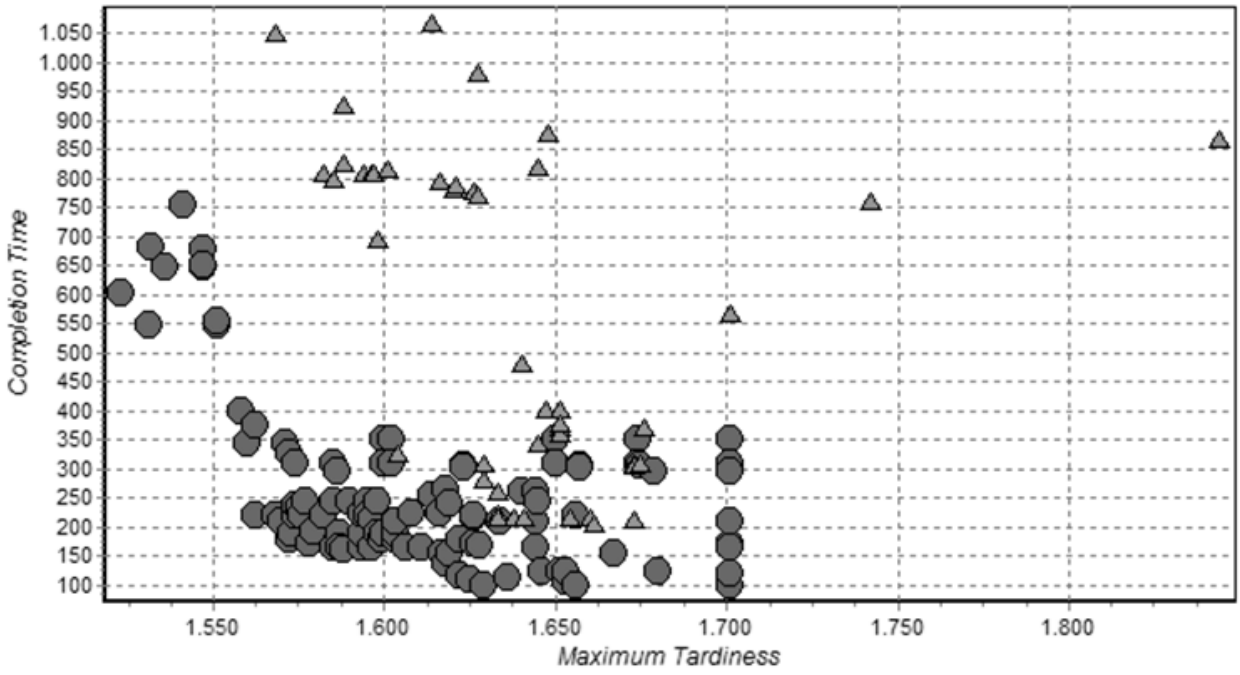


Figure 10 – MHSO (circles) and IMGA (triangles) Pareto borders