Available online at www.sciencedirect.com

SciVerse ScienceDirect

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 281 (2011) 51–67

www.elsevier.com/locate/entcs

Multiobjective Harmony Search Algorithm Proposals

Juan Ricart[1] Germán Hüttemann[2] Joaquín Lima[3]
Benjamín Barán[4]

Facultad Politécnica
Universidad Nacional de Asunción
San Lorenzo, Paraguay

Abstract

Harmony Search metaheuristic is successfully used in several applications of science and engineering. However, its effectiveness in solving multiobjective optimization problems using the concepts of Pareto optimality, remains unproved. This paper presents two proposals of the Harmony Search metaheuristic for multiobjective optimization, using the ZDT functions as a test bed. Performance metrics for experimental results show that the proposals are competitive even when compared to NSGA-II evolutionary algorithm.

Keywords: Harmony Search, Multiobjective Optimization, Musicians Improvisation, Pareto Dominance.

1 Introduction

Most real-world engineering optimization problems are multiobjective in nature, since they normally have several objectives that must be optimized (minimized or maximized) at the same time. Usually, these objectives are in conflict, that is, improving one worsens others. For this reason, it is not always possible to find a unique optimal solution for these problems, as it is possible with mono-objective optimization problems. Instead of finding a unique optimal solution, the goal in multiobjective optimization problems is to find a set of good compromises or "trade-off" solutions, from which an expert chooses one or several acceptable solutions for the tackled problem [2].

[1] Email: juangabriel.ricart@gmail.com
[2] Email: ghuttemann@gmail.com
[3] Email: joaquinlima@gmail.com
[4] Email: bbaran@pol.una.py

doi:10.1016/j.entcs.2011.11.025

Harmony Search Algorithm (HS) is a recent metaheuristic inspired by the music improvisation process. It was proposed as a mono-objective problem optimization metaheuristic and, since its creation, it has shown to be effective and convenient in several science and engineering applications, as seen in [14]. The natural and efficient application of HS algorithm to multiobjective optimization problems is a proposed extension of the existing research. Furthermore, the use of this algorithm to solve complex multiobjective problems, like NP-hard ones, is considered as a future challenge [22].

This paper presents two proposals of the HS algorithm to solve general multiobjective optimization problems. To prove the effectiveness of the proposed algorithms, ZDT functions [24] are used as test bed and results of those algorithms are compared to the solutions obtained with NSGA-II [4].

The paper is organized as follows: Section 2 presents the definition of multiobjective optimization problem and other related concepts. Section 3 explains the Harmony Search metaheuristic in detail, making first a comparison to musical improvisation process, and then, describing each part of the algorithm structure. Section 4 presents a brief review of works that treat HS algorithm and multiobjective optimization. Section 5 presents the adaptations of the mono-objective HS algorithm for both proposals of this paper. Next, section 6 presents the set of tests and the experimental results. Finally, section 7 presents conclusions and future work.

2 Multiobjective Optimization Problems

Formally, a multiobjective optimization problem can be defined as follows:

Definition 2.1 (Multiobjective Optimization Problem):

$$\text{Optimize } \mathbf{y} = \mathbf{F}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})] \tag{1}$$

$$\text{subject to } g(\mathbf{x}) = [g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_m(\mathbf{x})] \geq 0 \tag{2}$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_n] \in \mathbf{X} \subseteq \mathbb{R}^n, \mathbf{y} = [y_1, y_2, \ldots, y_k] \in \mathbf{Y} \subseteq \mathbb{R}^k$

$\mathbf{x}$ is a n-dimensional vectorial decision variable, $\mathbf{y}$ is a k-dimensional objective vector, $\mathbf{X} \subseteq \mathbb{R}^n$ denotes the decision space, $\mathbf{Y} \subseteq \mathbb{R}^k$ denotes the objective space. Though, a multiobjective optimization problem has n decision variables, m constraints and k objectives. *Optimization* refers then to the maximization or minimization of the k objective functions. Throughout the remainder of this paper, the treated multiobjective optimization problems are exclusively minimization problems.

The existence of more than one optimum (or trade-off) solution in multiobjective optimization problems ($k > 1$) makes necessary a different notion of optimum. The most commonly accepted notion of optimum is a proposal known as *Pareto Optimum* [2]. The fundamental concepts to understand the notion of *Pareto Op-*

timum are: *Pareto Dominance*, *Pareto Optimality*, *Pareto Optimal Set* and *Pareto Optimal Front*, defined as follows [20]:

Definition 2.2 (Pareto Dominance): a vector $\mathbf{u} = [u_1, u_2, \ldots, u_k]$ dominates another vector $\mathbf{v} = [v_1, v_2, \ldots, v_k]$ if and only if $\mathbf{u}$ is partially better than $\mathbf{v}$, i.e., $\forall i \in \{1, 2, \ldots, k\}$, $u_i \leq v_i \wedge \exists i \in \{1, 2, \ldots, k\}$ such that $u_i < v_i$. In this case, we say that vector $\mathbf{u}$ dominates (or it is better than) vector $\mathbf{v}$, which is denoted as $\mathbf{u} \succ \mathbf{v}$. Another two used notations are: $\mathbf{u} \prec \mathbf{v}$, meaning than $\mathbf{u}$ is dominated by $\mathbf{v}$, and $\mathbf{u} \sim \mathbf{v}$, meaning that neither vector $\mathbf{u}$ dominates $\mathbf{v}$ ($\mathbf{u} \not\succ \mathbf{v}$) nor $\mathbf{v}$ dominates $\mathbf{u}$ ($\mathbf{v} \not\succ \mathbf{u}$) and therefore, they are not comparables, i.e. $\mathbf{u} \sim \mathbf{v}$ if $\mathbf{u} \not\succ \mathbf{v} \wedge \mathbf{u} \not\prec \mathbf{v} \wedge \mathbf{u} \neq \mathbf{v}$.

Definition 2.3 (Pareto Optimality): a solution $\mathbf{x} \in \mathbf{X}$ is said to be Pareto optimal with respect to $\mathbf{X}$ if and only if it does not exist a solution $\mathbf{x'} \in \mathbf{X}$ for which $\mathbf{v} = \mathbf{F}(\mathbf{x'}) = [f_1(\mathbf{x'}), f_2(\mathbf{x'}), \ldots, f_k(\mathbf{x'})]$ dominates $\mathbf{u} = \mathbf{F}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})]$.

Definition 2.4 (Optimal Pareto Set): for a multiobjective optimization problem $\mathbf{F}(\mathbf{x})$, the optimal Pareto set is defined as the set of all non-dominated solutions with respect to $\mathbf{X}$, i.e. $\mathbf{P} = \{\mathbf{x} \in \mathbf{X} \mid \nexists \mathbf{x'} \in \mathbf{X} \text{ for which } \mathbf{F}(\mathbf{x'}) \succ \mathbf{F}(\mathbf{x})\}$.

Definition 2.5 (Optimal Pareto Front): for a multiobjective optimization problem $\mathbf{F}(\mathbf{x})$ and an optimal Pareto set $\mathbf{P}$, the optimal Pareto front is defined as the set: $\mathbf{PF} = \{\mathbf{u} = \mathbf{F}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})] \mid \mathbf{x} \in \mathbf{P}\}$.

3 The Harmony Search metaheuristic

Before introducing the HS algorithm and each of its parts, a brief comparison to the processes of musical improvisation (and optimization) will be given in section 3.1. Then, the structure of the algorithm together with each of its parts will be treated in section 3.2.

3.1 Music improvisation and optimization

The Harmony Search metaheuristic is an emerging optimization algorithm inspired by the underlying principles of music improvisation. When musicians make up a harmony, they usually test various pitch combinations stored in their memories. The process of searching for optimal solutions to engineering problems is analogous to this efficient search for a perfect state of harmony [21]. Table 1 presents a comparison between music improvisation and optimization.

Figure 1 shows the structure of the Harmony Memory (HM), which is the core of the HS algorithm, as well as an analogy between music improvisation and optimization. Consider a jazz trio consisting of a saxophone, a double bass and a guitar. There are few favorite pitches in the memory of each musician: saxophonist, {C, D, E}; double bassist, {E, F, G}; and guitarist, {G, A, B}. If the saxophonist chooses randomly {C} from its memory, the double bassist chooses {E}, and the guitarist chooses {G}, the new harmony {C, E, G} is made. If this harmony is better than

Comparison factor	Music improvisation	Optimization
Best state	Perfect harmony	Global optimum
Estimated by	Aesthetic standard	Objective function
Estimated with	Instrument pitches	Variable values
Processing unit	Each practice	Each iteration

Table 1
Comparison between music improvisation and optimization

the worst in HM, then the new harmony replaces it. This process is repeated until perfect harmony is reached.

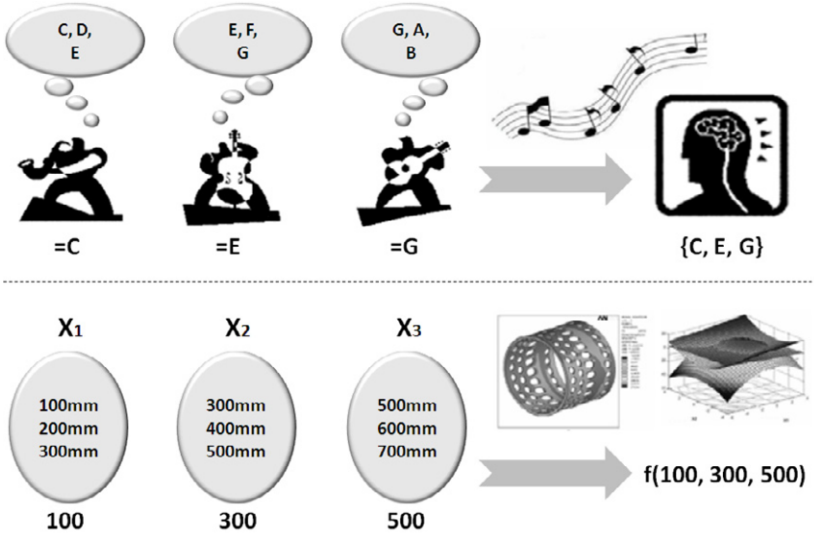

Fig. 1. Harmony memory structure and analogy between music improvisation and optimization

In a real optimization context, each musician is replaced by each decision variable, and favorite pitches by favorite variable values. If each variable represents the diameter of a pipe between two nodes in a water distribution network, each has a certain number of favorite diameters. If the first variable, x_1 chooses {100 mm} from {100 mm, 200 mm, 300 mm}, the second variable x_2 chooses {300 mm} from {300 mm, 400 mm, 500 mm} and the third variable x_3 chooses 500 mm from {500 mm, 600 mm, 700 mm}, these values make a new solution vector, {100 mm, 300 mm, 500 mm}. If the solution vector is better than the worst vector in HM, then the new vector replaces the worst one in HM. This process is repeated until a stopping criterion for termination of the algorithm is reached or the global optimum is found.

3.2 The structure of the algorithm

In order to explain the HS algorithm in greater detail, it is required to idealize the process of improvisation done by an expert musician. When a musician is improvising, he can choose among three options: (1) to execute any pitch from memory; (2) to execute a pitch adjacent to any other in his memory; (3) to execute a random pitch from the range of all possible pitches. Similarly, when each decision variable picks a value, there are three options: (1) to pick any value from the memory; (2) to pick a value adjacent to any value in the memory; (3) to pick a random value from the domain of all possible values. Geem et al. [12] formalized these three options to create a new metaheuristic in 2001, and the three corresponding components were: *memory use or consideration, pitch adjustment and randomness.* These three options are employed in the HS algorithm using two parameters: memory consideration rate, and pitch adjustment rate.

Having explained the three main components of the HS algorithm: harmony memory (HM), harmony memory consideration rate (HMCR) and pitch adjustment rate (PAR), the following subsections explain each step that comprises the HS algorithm.

3.2.1 Problem formulation

As is presented in section 1, the HS algorithm was initially conceived for solving optimization problems where a single objective is considered. Therefore, to apply the canonical HS to a problem, it must be formulated as a mono-objective optimization problem, with one objective function and several constraints:

$$f(x) = f(x_1, x_2, \ldots, x_n) \tag{3}$$

subject to

$$h_i(x) = 0;\ i = 1, \ldots, p \tag{4}$$

$$g_i(x) \geq 0;\ i = 1, \ldots, q \tag{5}$$

$$x_i \in X_i = \{x_i(1), x_i(2), \ldots, x_i(K_i)\} \quad \text{or} \quad x_i^L \leq x_i \leq x_i^U \tag{6}$$

The HS algorithm searches all the solution space to find the optimal solution vector $x = (x_1, x_2, \ldots, x_n)$ that optimizes (minimizes or maximizes) the objective function in equation 3. If the problem has equality or inequality conditions these may be considered as constraints, as in equations 4 and 5. If the decision variables have discrete values, the set of possible values is given by $x_i \in X_i = \{x_i(1), x_i(2), \ldots, x_i(K_i)\}$, where K_i is the number of different values in the definition space for variable i, on the other hand, if the variables have continuous values, the set of possible values is given by $x_i^L \leq x_i \leq x_i^U$.

3.2.2 Parameter configuration

Once the problem formulation is ready, the parameters of the algorithm must be configured with values. Furthermore, besides the two parameters already mentioned, HMCR and PAR, the HS algorithm has other parameters such as: harmony memory size (HMS), maximum amount of improvisations or iterations (Maximum Improvisations, MI) and pitch range variability (Fret Width, FW [11]) that operate altogether with PAR in pitch adjustment.

Memory consideration is important because it ensures that good solutions are considered as elements of new solutions. If this parameter is too low, only few good solutions are selected, and convergence may be slow. If this parameter is extremely high (close to 1), the memory values are mostly used and other alternatives are not well explored, resulting in not very good solutions. Therefore, in order to use the memory effectively, HMCR $\in [0.70; 0.95]$ [22].

Pitch adjustment is similar to the mutation operator in genetic algorithms. A low value for PAR together with a narrow value for FW can make the convergence of the HS algorithm slow, given the limitation in exploration to a single portion of the search space. On the other hand, a very high value for PAR together with a wide value for FW may cause solutions to disperse around a few potential optima as in random search. For these reasons, usually, PAR$[0.1; 0.5]$ and FW, generally, is bounded between 1% and 10% of all the range of variable values [22].

3.2.3 Memory initialization

After the problem has been formulated and the parameters properly configured, a random configuration process is performed on the memory.

The HS algorithm initially improvises several solutions randomly. The number of solutions must be at least equal to HMS. Nonetheless, it may be higher, as much as double or triple [5]. Then, the best HMS solutions are selected. HM may be viewed as the following matrix:

$$\mathrm{HM} = \left[\begin{array}{cccc|c} x_1^1 & x_2^1 & \cdots & x_n^1 & f(x^1) \\ x_1^2 & x_2^2 & \cdots & x_n^2 & f(x^2) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_1^{HMS} & x_2^{HMS} & \cdots & x_n^{HMS} & f(x^{HMS}) \end{array}\right] \tag{7}$$

3.2.4 Improvisation

As it was mentioned in subsection 3.2, there are three options among which the HS algorithm may choose when performing an improvisation:

(i) **Random selection:** When HS determines the x_i^{new} value for a new solution $x^{new} = [x_1^{new}, x_2^{new}, \ldots, x_n^{new}]$, it randomly chooses a value from the range of all possible values $\{x_i(1), x_i(2), \ldots, x_i(K_i)\}$ or $x_i^L \leq x_i \leq x_i^U$ with a probability of $(1 - \mathrm{HMCR})$.

(ii) **Memory consideration:** When HS determines the value of x_i^{new}, it randomly chooses the x_i^j value from HM ($j = 1, 2, \ldots, \text{HMS}$) with a probability equal to HMCR. The random index j may be calculated using an uniform distribution $\text{U}(0, 1)$:

$$j \leftarrow \text{int}(\text{U}(0,1) * \text{HMS}) + 1 \tag{8}$$

(iii) **Pitch adjustment:** After the value of x_i^{new} has been randomly chosen from HM in the process previously described, it may be adjusted to neighboring values adding or substracting a given amount, with probability PAR. For discrete variables, if $x_i(k) = x_i^{new}$, the pitch adjustment is $x_i(k + m)$, where $m \in \{-1, 1\}$. For continuous variables, the pitch adjustment is $x_i^{new} + \Delta$, where $\Delta = \text{U}(-1, 1) * \text{FW}(i)$.

The three basic operations previously mentioned may be expressed as follows:

$$x_i^{new} \leftarrow \begin{cases} \begin{cases} x_i \in \{x_i(1), x_i(2), \ldots, x_i(K_i)\} \\ x_i \in [x_i^L, x_i^U] \end{cases} & \text{w.p. } (1 - \text{HMCR}) \\ x_i \in \text{HM} = \{x_i^1, x_i^2, \ldots, x_i^{\text{HMS}}\} & \text{w.p. HMCR} * (1 - \text{PAR}) \\ \begin{cases} x_i(k+m) & \text{if } x_i(k) \in \text{HM} \\ x_i + \Delta & \text{if } x_i \in \text{HM} \end{cases} & \text{w.p. HMCR} * \text{PAR} \end{cases} \tag{9}$$

where *w.p.* stands for "*with probability*".

3.2.5 Memory update

If the new solution x^{new} is better than the worst solution in HM in terms of the objective function value, the new solution is included in HM and the worst is discarded:

$$(x^{new} \in \text{HM}) \wedge (x^{worst} \notin \text{HM}) \tag{10}$$

3.2.6 Termination

If the HS algorithm meets the stopping criterion (for instance, has reached the maximum amount of iterations or the maximum execution time), the process is terminated; otherwise, another solution is improvised. The pseudocode of the algorithm is presented in algorithm 1 [22].

4 Multiobjective problem solving using Harmony Search with approaches not based on Pareto dominance

Although in current bibliography we can find a few works that propose the application of the HS algorithm for solving specific multiobjective problems, most of them

Algorithm 1 Pseudocode of the Harmony Search algorithm

Input: f(x), HMCR, PAR, HMS, MI, FW
Output: x^{best} in HM
  Randomly initialize HM
  while stopping criterion is not satisfied **do**
    for each variable x_i **do**
      if $U(0,1) <$ HMCR **then**
        $x^{new} \leftarrow x_i^j$ where $j \leftarrow \text{int}(\text{U}(0,1) * \text{HMS}) + 1$
        if $U(0,1) <$ PAR **then**
          Update x_i^{new} with $x_i(k+m)$ or $x_i^{new} + \Delta$
        end if
      else
        $x^{new} \leftarrow$ random value
      end if
    end for
    if x^{new} is better than x^{worst} **then**
      Replace x^{worst} with x^{new} in HM
    end if
  end while

either do not use the notion of *Pareto Optimality* or do not explicitly detail the proposed multiobjective method.

Geem et al. [13] applied the HS algorithm to vehicle routing, specifically to the *School Bus Routing Problem*, minimizing the number of buses and the total traveling time of all buses, while satisfying the capacity constraints of the buses and the time windows at each stop.

Geem and Hwangbo [9] used the HS algorithm to optimize the design of satellite heat pipes. This problem consists in finding the dimensions of pipes and their operating temperature to minimize the total mass of the pipes and to maximize thermal conductance.

Geem [10] also used the HS algorithm to optimize project planning, attempting to minimize the time and cost in the *Time-Cost Trade-Off Problem*.

In the previous mentioned works, a single objective function that results from the weighted aggregation of functions that correspond to each objective to optimize, is minimized. Furthermore, the first two works do not apply the notion of *Pareto optimality*. However, Geem [10] mentions an implementation of a ranking among Pareto optimum solutions, but without elaborating on the method.

Gao et al. [7] proposed a modified version of the HS algorithm for mono-objective optimization problems with constraints, using the *Pareto optimality* concept. The function to be optimized results from the aggregation of the objective function and the weighted sum of constraint functions. The application of the *Pareto optimality* concept allows to rank the non-feasible solutions, that is, those that violate one or more constraints, allowing them to evolve, like feasible solutions.

Finally, Xu et al. [21] proposed a multiobjective version of the HS algorithm

for the *design of a reconfigurable mobile robot prototype*. The objectives to be minimized in this problem are the stability of the mobile robot, torque resistance of the rear wheels and the mass of the mobile robot. Although this work uses the notion of *Pareto optimality* and the problem is presented according to definition 2.1, it does not mention how the Pareto ranking of solutions is done nor explains in detail the changes to the original HS to produce a multiobjective version. Even so, to the best of our knowledge, this proposal is the only one that fully complies with the requirements of the formulation and resolution of a truly multiobjective optimization problem.

5 Proposals of multiobjective Harmony Search algorithms based on Pareto dominance

The main difference between the mono-objective HS algorithm and the multiobjective variants proposed in the present work is based on the solution sorting procedure, also known as ranking assignment.

Ranking assignment consists in associating a number to each solution as it is deemed better or worse than the rest, determining an order for the solutions where the best ones have a lower ranking than the worse.

In a mono-objective problem, the ranking of solutions is determined by the objective function. However, several methods were developed for multiobjective problems that incorporate the concepts of *Pareto optimality* [18].

The ranking assignment method chosen for the multiobjective variants of the HS algorithm is the one proposed by Fonseca and Fleming [6], where the ranking of a solution x_i for an iteration t is defined according to the equation:

$$\text{rank}(x_i, t) = 1 + p_i^{(t)} \tag{11}$$

where $p_i^{(t)}$ denotes the number of solutions for the current iteration which dominate the solution in question. So, non-dominated solutions of HM are assigned a ranking equal to 1, while dominated solutions have a ranking equal to $q \in \{2, \ldots, \text{HMS}\}$. Figure 2 shows a hypothetical example that illustrates several points in objective space with two functions to be optimized and where Fonseca-Fleming ranking is assigned to each point.

5.1 MOHS1: Multiobjective Harmony Search, 1st proposal

This multiobjective variant was conceived so as to not require significant changes in the behavior of the original HS algorithm. The pseudocode for this proposal is shown in algorithm 2.

The difference consists in the use of HM as a repository for the best trade-off solutions found at a given point in time, specifying a ranking for them according to the Fonseca-Fleming method.

The algorithm begins by initializing HM with solutions generated randomly. Then it proceeds to calculate the ranking of these solutions, an operation which

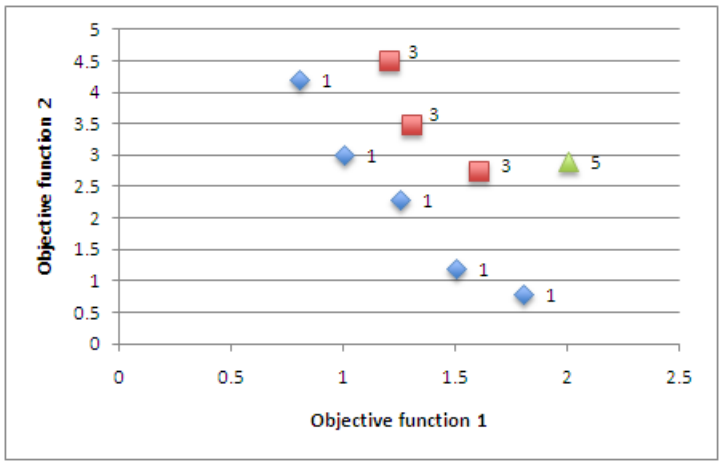

Fig. 2. Fonseca-Flemming ranking for points in a two-dimensional objective space

Algorithm 2 Pseudocode for MOHS1

Input: F(x), HMCR, PAR, HMS, MI, FW
Output: P extracted from HM
  Randomly initialize HM
  while stopping criterion is not satisfied **do**
    Improvise a new solution S
    Calculate the Pareto ranking of S considering HM
    if S has a better ranking than the worst solution in HM **then**
      Update HM with S
    end if
  end while

incurs in an asymptotic cost of $O(\text{HMS}^2)$ to calculate the ranking, comparing each solution to the others.

At each iteration, the algorithm tries to find a new trade-off solution, using the decision variable values of the solutions in HM as considering values. This procedure remains identical to the mono-objective HS algorithm.

For this new generated solution, a ranking is calculated considering the solutions stored in HM. If this ranking turns out to be better than the worst ranked solution in HM, the new solution is stored in HM replacing the worst one, choosing randomly among the worst ones if there is more than one. This procedure incurs in an asymptotic cost of $O(\text{HMS})$ to recalculate the ranking of the solutions in HM.

At each iteration, the non-dominated solutions stored in HM correspond to an approximation to the Pareto set for the problem to be solved. When the stopping criterion is met, the solutions with ranking equal to one (non-dominated solutions) stored in HM are returned as the best approximation to the Pareto optimum for the multiobjective problem to be solved.

5.2 MOHS2: Multiobjective Harmony Search, 2nd proposal

The main idea of this proposal, whose pseudocode is shown in algorithm 3, is to generate a new memory HM_2 at each iteration with the same number of solutions as the original memory, HM_1. From the union of both memories, $H_u \leftarrow HM_1 \cup HM_2$, only half the number of solutions are admitted as components of the memory for the next iteration.

Algorithm 3 Pseudocode for MOHS2

```
Input: F(x), HMCR, PAR, HMS, MI, FW
Output: P extracted from HM1
  Randomly initialize HM1
  while stopping criterion is not satisfied do
    Empty HM2
    while HM2 is not filled do
      Improvise a new solution S from HM1
      Store S in HM2
    end while
    Hu ← HM1 ∪ HM2
    Calculate Pareto ranking of Hu using Fonseca-Fleming definition
    Emtpy HM1
    R ← 1
    F ← Extract all solutions from Hu with ranking R
    while HM1 has space ≥ |F| (cardinality of set F) do
      Move all solutions from F to HM1
      R ← R + 1
      F ← Extract all solutions from Hu with ranking R
    end while
    T ← Space left in HM1
    if T > 0 then
      Truncate F to size T
      Move every solution F to HM1
    end if
  end while
```

The proposed algorithm begins in the same way as the mono-objective HS algorithm, generating solutions with random values for the variables until HM_1 is filled.

Each iteration begins by improvising all solutions that will belong to HM_2, using the same selection method as the mono-objective HS algorithm. This is, the values of each calculated solution are generated using the values of decision variables contained in HM_1 as considered values. Once the solutions for HM_2 are generated, the Fonseca-Fleming ranking of H_u is calculated.

Once the ranking assignment procedure is finished, the best solutions of H_u are selected. To accomplish this, firstly the solutions in H_u are grouped in fronts, where each front contains solutions with the same ranking. Then solutions from

this fronts are transfered to HM_1 in ascending order, i.e. solutions in the front with the lowest ranking first, and then those fronts with successively higher rankings. This procedure continues until the number of solutions in a front is equal or larger than the space available in HM_1. If the size of the front exceeds the space available in HM_1, a truncating procedure developed for the SPEA2 algorithm [25] is applied. The truncation reduces the number of solutions until the size of the front is equal to the space available in HM_1. Having completed the transfer of solutions, HM_1 has a new set of solutions for the next iteration.

When the stopping criterion for the algorithm is met, the solutions in HM_1 with ranking equal to one (non-dominated solutions) are returned as the best approximation to the Pareto optimum set.

6 Experimental results

To evaluate the proposed multiobjective HS algorithms, a set of six test functions was selected that contemplate several characteristics, such as convexity, non-convexity, discreteness and non-uniformity, together with multimodal and deceptive characteristics. These functions consider the minimization of two objectives and are known as "*ZDT functions*" (from their creators Zitzler, Deb and Thiele [24]).

Each one of these six functions (ZDT1, ZDT2, ZDT3, ZDT4, ZDT5, and ZDT6) was used to compare the two proposed multiobjective HS algorithms to the well known NSGA-II (Non-dominated Sorting Genetic Algorithm II) evolutionary algorithm created by Deb et al [4], and available at [3]. The chosen evaluation metrics were the ones proposed in [24]:

- M_1^*: Distance between calculated Pareto front and optimal Pareto front.
- M_2^*: Distribution of the calculated Pareto front.
- M_3^*: Extension of the calculated Pareto front.

6.1 Parameter configuration of the compared algorithms

The adopted values for the parameters of the multiobjective HS algorithms proposed: HMCR, PAR, HMS, and FW, were taken following advice from Yang [22], Gao [7], and Geem [8,11]. The values for these variables are shown in table 2.

The values adopted for the parameters of the NSGA-II evolutionary algorithm: population size, crossover probability, mutation probability, distribution index for crossover and distribution index for mutation, were taken from [3]. The values for these variables are shown in table 3.

6.2 Results

Each of the three algorithms was executed 10 times with 10 seconds per execution, for each test function. Experimental results are shown in tables 4 to 6. These three tables show the average values obtained with all 10 executions for each algorithm and each test function, together with the corresponding standard deviation

Parameter	Value
HMCR	0.95
PAR	0.1
HMS	100
FW (ZDT5)	40%
FW (other ZDT)	1%

Table 2
Parameters for the multiobjective HS algorithms

Parameter	Value
Population size	100
Crossover probability	0.9
Mutation probability (ZDT5)	1/b
Mutation probability (other ZDT)	1/n
Distribution index for crossover	15
Distribution index for mutation	20

Table 3
Parameters for the NSGA-II algorithm. n is the number of variables for the problem and b is the total number of bits of all variables (for a binary problem).

in parenthesis. It is worth mentioning that these values are normalized according to Lima [17,16] using the theoretical optimal Pareto front. This way, one is the optimal value and the further the metric value is from one, the worse it is. Furthermore, figure 3 shows a comparison among the calculated Pareto fronts for each of the three algorithms and the optimum Pareto front (PF_{true} [20]) of each ZDT function.

As seen on table 4, neither algorithm outperforms the others in every ZDT function, but the MOHS1 algorithm has the best results on average, followed by the MOHS2 and last NSGA-II. The most notable difference can be observed for the ZDT5 test function, where the MOHS1 and MOHS2 algorithms clearly outperform NSGA-II in an order of magnitude, what significantly contributes to the average. This difference can be best seen in figure 3e. Although small numerical differences exist for the other problems, this is graphically inappreciable, as shown in figures 3a to 3d and 3f.

Considering the values shown on table 5, we can see that the best algorithm in terms of the distribution metric M_2^* is MOHS2 closely followed by NSGA-II, with MOHS1 in last place. For each problem, the differences in value are small, although MOHS1 has values consistently lower for all problems. Graphically, the distribution metric behaves in a similar way as M_1^*, as can be seen in figure 3.

Fig. 3. Comparison between the calculated Pareto fronts of the three algorithms and the Pareto optimum of each ZDT function.

For the extension metric, as shown in table 6, the NSGA-II algorithm is the best in average, followed by MOHS2 with MOHS1 last. There exists an unusual value that exceed one. This is because the extension of the calculated Pareto front exceeded the extension of the theoretical optimal Pareto front, what can be seen particularly for NSGA-II and ZDT5. In this case, as can be seen in figure 3e, this is due to the fact that NSGA-II finds one solution very far from the rest, which is not necessarily a good characteristic.

In summary, neither of the algorithms outperform the others for every problem and/or every metric, but the proposed MOHS algorithms are very competitive when compared to a well known state-of-the-art alternative as NSGA-II.

Function	NSGA-II	MOHS1	MOHS2
ZDT1	0.9918(0.00)	0.9922(0.00)	0.9922(0.00)
ZDT2	0.9957(0.00)	0.9960(0.00)	0.9962(0.00)
ZDT3	0.9571(0.00)	0.9533(0.00)	0.9529(0.00)
ZDT4	0.9959(0.00)	0.9963(0.00)	0.9688(0.04)
ZDT5	0.0858(0.06)	0.6223(0.10)	0.6314(0.16)
ZDT6	0.9420(0.00)	0.9500(0.00)	0.9385(0.00)
Average(σ)	**0.8281(0.36)**	**0.9184(0.15)**	**0.9133(0.14)**

Table 4
Average results obtained for the normalized M_1^* metric: *Distance between the calculated Pareto front and the optimum Pareto front* (standard deviation is shown in parenthesis).

Function	NSGA-II	MOHS1	MOHS2
ZDT1	0.8262(0.00)	0.8225(0.00)	0.8261(0.00)
ZDT2	0.8247(0.00)	0.8107(0.01)	0.8263(0.00)
ZDT3	0.8346(0.00)	0.8312(0.00)	0.8375(0.00)
ZDT4	0.8271(0.00)	0.8191(0.00)	0.8258(0.00)
ZDT5	0.8022(0.00)	0.7787(0.03)	0.8204(0.02)
ZDT6	0.8241(0.00)	0.6719(0.03)	0.8225(0.00)
Average(σ)	**0.8232(0.01)**	**0.7890(0.06)**	**0.8264(0.01)**

Table 5
Average results obtained for the normalized M_2^* metric: *Distribution of the calculated Pareto front* (standard deviation is shown in parenthesis).

7 Conclusions and further work

The Harmony Search metaheuristic has shown to be efficient in solving various kinds of optimization problems in engineering. Nonetheless, its correct application to the resolution of multiobjective optimization problems remains a task for the future.

This work documents in detail two proposals for a general resolution of multiobjective problems using Harmony Search, considering a representative test bed that is classic in the optimization literature [4,19,25]. Experimental results show that the proposed algorithms are competitive when compared to NSGA-II, currently regarded as one of the most representative algorithm of the state-of-the-art in evolutionary algorithms for multiobjective optimization.

Considering these promising results, the following future course of action is proposed: (1) to study the performance of the algorithms presented in this work using the test bed proposed in CEC 2009 [23]; (2) to make a temporal analysis of the be-

Function	**NSGA-II**	**MOHS1**	**MOHS2**
ZDT1	1.0000(0.00)	0.9943(0.00)	1.0000(0.00)
ZDT2	1.0000(0.00)	0.9936(0.01)	1.0000(0.00)
ZDT3	1.0000(0.00)	0.9983(0.00)	0.9999(0.00)
ZDT4	1.0000(0.00)	0.9916(0.01)	1.0087(0.01)
ZDT5	1.1153(0.00)	0.7224(0.03)	0.8142(0.04)
ZDT6	0.9999(0.00)	1.0000(0.00)	1.0000(0.00)
Average(σ)	**1.0192(0.05)**	**0.9500(0.11)**	**0.9705(0.08)**

Table 6
Average results obtained for the normalized M_3^* metric: *Extension of the calculated Pareto front* (standard deviation is shown in parenthesis).

havior of the algorithms in solving all considered problems, to obtain metric results with respect to execution time; (3) to apply the new proposals to the resolution of classic problems in combinatorial optimization, for instance the *Traveling Salesman Problem* (TSP) [1]; and (4) to validate the MOHS proposals comparing them to other metaheuristics, for instance *Particle Swarm Optimization* (PSO) [15].

References

[1] Applegate, D. L., R. E. Bixby, V. Chvátal and W. J. Cook, "The Traveling Salesman Problem: A Computational Study," Princeton Series in Applied Mathematics, Princeton University Press, 2006.

[2] Coello Coello, C., *A Short Tutorial on Evolutionary Multiobjective Optimization*, in: *Evolutionary Multi-Criterion Optimization*, Lecture Notes in Computer Science **1993**, Springer, 2001 pp. 21–40.

[3] Deb, K., *Multiobjective NSGA-II code in C*, accessed July 2010.
URL http://www.iitk.ac.in/kangal/codes/nsga2/nsga2-gnuplot-v1.1.5-64bit.tar.gz

[4] Deb, K., A. Pratap, S. Agarwal and T. Meyarivan, *A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II*, Evolutionary Computation, IEEE Transactions on **6** (2002), pp. 182–197.

[5] Degertekin, S., *Optimum Design of Steel Frames using Harmony Search Algorithm*, Structural and multidisciplinary optimization **36** (2008), pp. 393–401.

[6] Fonseca, C. and P. Fleming, *Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization*, , **423**, Citeseer, 1993, pp. 416–423.

[7] Gao, X., X. Wang and S. Ovaska, *Harmony Search Methods for Multi-modal and Constrained Optimization*, Music-Inspired Harmony Search Algorithm (2009), pp. 39–51.

[8] Geem, Z., *Optimal Cost Design of Water Distribution Networks using Harmony Search*, Engineering Optimization **38** (2006), pp. 259–277.

[9] Geem, Z. and H. Hwangbo, *Application of Harmony Search to Multi-objective Optimization for Satellite Heat Pipe Design*, in: *Proceedings of US-Korea conference on science, technology and entrepreneurship*, Citeseer, Teaneck, NJ, USA, 2006, pp. 1–3.

[10] Geem, Z. W., *Multiobjective Optimization of Time-Cost Trade-Off Using Harmony Search*, Journal of Construction Engineering and Management **136** (2010), pp. 711–716.

[11] Geem, Z. W., *State-of-the-Art in the Structure of Harmony Search Algorithm*, in: *Recent Advances in Harmony Search Algorithm*, Studies in Computational Intelligence **270**, Springer, 2010 pp. 1–10.

[12] Geem, Z. W., J. H. Kim and G. Loganathan, *A New Heuristic Optimization Algorithm: Harmony Search*, Simulation **76** (2001), pp. 60–68.

[13] Geem, Z. W., K. S. Lee and Y. Park, *Application of Harmony Search to Vehicle Routing*, American Journal of Applied Sciences **2** (2005), pp. 1552–1557.

[14] Ingram, G. and T. Zhang, *Overview of Applications and Developments in the Harmony Search Algorithm*, Music-Inspired Harmony Search Algorithm **191** (2009), pp. 15–37.

[15] Kennedy, J. and R. Eberhart, *Particle Swarm Optimization*, , **4**, IEEE, Perth, WA, Australia, 1995, pp. 1942–1948.

[16] Lima, J., "Optimización de Enjambre de Partículas aplicada al Problema del Cajero Viajante Bi-objetivo," Facultad Politécnica, Universidad Nacional de Asunción, San Lorenzo, Paraguay, 2007, Tesis de Grado para el título de Ingeniero en Informática.

[17] Lima, J. and B. Barán, *Optimización de Enjambre de Partículas aplicada al Problema del Cajero Viajante Bi-objetivo*, Inteligencia Artificial: Revista Iberoamericana de Inteligencia Artificial **10** (2006), pp. 67–76.

[18] Mallor, F., P. Mateo Collazos, I. Alberto Moralejo and C. Azcárate Giménez, *Multiobjective Evolutionary Algorithms: Pareto rankings*, in: *VII Jornadas Zaragoza-Pau de Matemática Aplicada y estadística: Jaca (Huesca), 17-18 de septiembre de 2001*, Prensas Universitarias de Zaragoza, 2003, pp. 27–36.

[19] Parsopoulos, K. and M. Vrahatis, *Recent Approaches to Global Optimization Problems through Particle Swarm Optimization*, Natural Computing **1** (2002), pp. 235–306.

[20] Veldhuizen, D. and G. Lamont, *Multiobjective Evolutionary Algorithms: Analyzing the state-of-the-art*, Evolutionary Computation **8** (2000), pp. 125–147.

[21] Xu, H., X. Z. Gao, T. Wang and K. Xue, *Harmony Search Optimization Algorithm: Application to a Reconfigurable Mobile Robot Prototype*, in: *Recent Advances in Harmony Search Algorithm*, Studies in Computational Intelligence **270**, Springer, 2010 pp. 11–22.

[22] Yang, X.-S., *Harmony Search as a Metaheuristic Algorithm*, in: *Music-Inspired Harmony Search Algorithm*, Studies in Computational Intelligence **191**, Springer, 2009 pp. 1–14.

[23] Zhang, Q., A. Zhou, S. Zhao, P. Suganthan, W. Liu and S. Tiwari, *Multiobjective Optimization Test Instances for the CEC 2009 Special Session and Competition*, Technical Report CES-487, University of Essex and Nanyang Technological University (2009).

[24] Zitzler, E., K. Deb and L. Thiele, *Comparison of Multiobjective Evolutionary Algorithms: Empirical results*, Evolutionary Computation **8** (2000), pp. 173–195.

[25] Zitzler, E., M. Laumanns and L. Thiele, *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*, Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland (2001).