applied sciences

MDPI

Article

A Novel Multi-Objective Harmony Search Algorithm with Pitch Adjustment by Genotype

Daniel Molina-Pérez, Edgar Alfredo Portilla-Flores, Eduardo Vega-Alvarado *, Maria Bárbara Calva-Yañez and Gabriel Sepúlveda-Cervantes

Instituto Politécnico Nacional-CIDETEC, Ciudad de México 07700, Mexico; dmolinap1800@alumno.ipn.mx (D.M.-P.); aportilla@ipn.mx (E.A.P.-F.); mvalenciar@ipn.mx (M.B.C.-Y.); gsepulvegac@ipn.mx (G.S.-C.)

* Correspondence: evega@ipn.mx

Abstract: In this work, a new version of the Harmony Search algorithm for solving multi-objective optimization problems is proposed, MOHSg, with pitch adjustment using genotype. The main contribution consists of adjusting the pitch using the crowding distance by genotype; that is, the distancing in the search space. This adjustment automatically regulates the exploration–exploitation balance of the algorithm, based on the distribution of the harmonies in the search space during the formation of Pareto fronts. Therefore, MOHSg only requires the presetting of the harmony memory accepting rate and pitch adjustment rate for its operation, avoiding the use of a static bandwidth or dynamic parameters. MOHSg was tested through the execution of diverse test functions, and it was able to produce results similar or better than those generated by algorithms that constitute search variants of harmonies, representative of the state-of-the-art in multi-objective optimization with HS.

Keywords: Harmony Search; metaheuristics; multi-objective optimization; crowding

Citation: Molina-Pérez, D.; Portilla-Flores, E.A.; Vega-Alvarado, E.; Calva-Yáñez, M.B.; Sepúlveda-Cervantes, G. A Novel Multi-Objective Harmony Search Algorithm with Pitch Adjustment by Genotype. *Appl. Sci.* **2021**, *11*, 8931. https://doi.org/10.3390/app11198931

Academic Editor: Zong Woo Geem

Received: 31 August 2021
Accepted: 20 September 2021
Published: 25 September 2021

1. Introduction

Most real engineering problems are multi-objective in nature, since commonly they present several objective functions to be optimized that are compromised with each other, that is, the improvement of one produces the deterioration of another. Therefore, without a function priority, a unique solution cannot be determined as in a single objective optimization. Instead, multi-objective optimization seeks to obtain the best compromises. The best compromised solutions are known as non-dominated and form the Pareto-optimal front [1].

Harmony Search (HS) is a metaheuristic algorithm developed by Geem et al. [2], emulating musical improvisation. It was proposed as a mono-objective algorithm and has been successfully applied in practical and scientific problems [3–6]. HS integrates three resources for the quantitative optimization process: use of a harmonic memory (HM), pitch adjustment, and randomization [7]. The implementation of multi-objective HS is a growing trend in scientific research, and its use for solving highly complex problems is considered a future challenge [8]. The first multi-objective cases solved by HS [9–11] were engineering problems treated with the weighting function method. In [11,12], the authors began to handle the Pareto front term, without explicitly establishing a multi-objective HS algorithm. In [13], Xu et al. developed for the first time a multi-objective HS to generate a Pareto-optimal front of five points for a robotics model. Ricart et al. [14] established two formal multi-objective proposals known as MOHS1 and MOHS2. Both of them are very similar to the single objective form of HS but with a fundamental difference in the ranking of the harmonies. Sivasubramani et al. [15] presented a proposal similar to MOHS2 (denominated as MOHS3 in this paper) that incorporates the crowding distancing and dynamic pitch adjustment proposed by Mahdavi et al. [16]. Nekooei et al. [17] developed a

multi-objective memetic algorithm by combining HS and PSO. However, it greatly differs from the original structure of HS.

In its original structure, HS is very effective at identifying promising areas in the solution space (exploration) but presents difficulties in carrying out refined searches (exploitation). For this reason, it is required to improve the balance between exploration and exploitation. There are two trends in this sense [18]. The first one is based on the synergistic combination of HS with other metaheuristic algorithms. In [19–21], PSO is applied to give a social component to the pitch adjustment. In [22,23], the authors proposed hybrid algorithms to improve the exploitation in HS by combining it with Artificial Bee Colony (ABC) and Stochastic Hill Climbing. However, these alternatives are often disruptive to the conceptual essence of HS. The second stream focuses on improving the pitch adjustment [24,25]. In [26], pitch adjustments with fixed parameters are proposed, using bandwidths that are proportional to the range of each variable. On the other hand, in [7,16,27], the authors propose the use of dynamic parameters to intensify exploitation as the algorithm execution advances. Gupta [28] proposed an HS variant using non-lineal dynamic parameters with a Gaussian distribution for the Harmony Memory Accepting Rate (HMCR) and bandwidth with a social component. In the case of multi-objective HS, the conventional mono-objective pitch and the dynamic pitch adjustments of Mahdavi et al. [16] have barely been implemented, both based on their effectiveness for single-objective problems. However, Pareto fronts can be generated for different dispersion degrees of the search space, also known as decision space [29]. Currently, most multi-objective HS proposals present a lack of analysis of the behavior of harmonies in the search space during the formation of Pareto fronts.

In this paper, the Multi-Objective HS algorithm MOHSg is proposed. The main contribution consists in the pitch adjustment using the crowding distance by genotype, that is, the distancing in the search space. This adjustment automatically regulates the exploration–exploitation balance of the algorithm, based on the distribution of the harmonies in the search space during the formation of Pareto fronts. Therefore, MOHSg only requires the presetting of the HMCR and the Pitch Adjustment Rate (PAR) for its operation, avoiding the use of the static bandwidth parameter required by the MOHS2 variant, and the dynamic parameters of bandwidth and PAR needed in MOHS3. The proposed algorithm was evaluated with a set of well-known test functions, generating similar or better results than those obtained by the MOHS2 and MOHS3 algorithms. The work is structured as follows: Section 2 presents the pitch adjustment variants reported in the literature, the rankings necessary for the multi-objective approach with a Pareto-optimal front, and the algorithms MOHS2 and MOHS3, while the crowding distancing by genotype and the pitch adjustment are described in Section 3. The proposed MOHSg algorithm is detailed in Section 4, and in Section 5, the experimentation and discussion of the results are carried out. Section 6 corresponds to the final discussion of the work, and Appendix A presents the test functions applied for the algorithm evaluation.

2. Non-Disruptive Multi-Objective HS Algorithms

The objective of this section is to present the MOHS2 and MOHS3 algorithms since they are representative alternatives for multi-objective HS. These algorithms will be used in a comparative analysis with the new MOHSg proposal, since in all cases the original conception of HS is preserved (non-disruptive algorithms), and the differences lie entirely in the pitch adjustment. First, it is necessary to review the pitch adjustment variants reported in the literature and their impact on the multi-objective approach, as well as the description of the ranking that is required for the Pareto-optimal front.

2.1. Pitch Adjustment Variants

Several pitch adjustment techniques have been proposed for HS [20]. In the original version, the pitch adjustment for a new variable x_i that belongs to a solution vector $\vec{x} = (x_1 x_2 x_3 \ldots x_M)$ with M design variables is given by expression (1):

$$x_i = \begin{cases} HM(r,i) + rand(-1,1) \times bw, & probability\ (PAR) \\ HM(r,i), & probability\ (1 - PAR) \end{cases} \quad (1)$$

where N is the number of harmonies; r is a random integer from 1 to N; $rand(-1,1)$ is a random number generated between -1 and 1; bw is the bandwidth; and PAR is the pitch adjustment rate. PAR and bw are preset to fixed values. This scheme is used by Ricart et al. [14] in the MOHS2 algorithm, where diverse problems are solved with fixed values for PAR and bw that are proportional to the range of each variable.

Tuo et al. [26] proposed one of the most used pitch adjustment methods with fixed parameters, with a fixed value of PAR, and a value of bw for each variable given by Equation (2), where L_b and U_b are the lower and upper limits of the specific variable, respectively:

$$bw = \frac{(U_b - L_b)}{1000} \quad (2)$$

Mahdavi et al. [16] presented an improvement to HS through the linear increase of PAR and the exponential decrease of bw in every iteration, as shown in Equations (3) and (4) where PAR_{min} and PAR_{max} are the minimum and maximum values of PAR, respectively; g is the current iteration; NI is the total number of iterations; and bw_{max} and bw_{min} are the maximum and minimum bandwidth, respectively. This encourages exploration in the early iterations of the algorithm and the subsequent intensification of exploitation. This scheme was applied by Sivasubramani et al. [15] to the development of the MOHS3.

$$PAR(g) = PAR_{min} + (PAR_{max} - PAR_{min}) \times \frac{g}{NI} \quad (3)$$

$$bw(g) = bw_{max} \times exp\left[ln\left(\frac{bw_{min}}{bw_{max}}\right) \times \frac{g}{NI}\right] \quad (4)$$

Portilla-Flores et al. [7] proposed a fixed PAR, and a bw that decreases exponentially depending on the range of each variable, as shown in Equation (5), where a is a positive constant in the range from 0 to 1. It generates a bw value for each variable, eliminating the parameters required in [16].

$$bw = \frac{(U_b - L_b)}{g^a} \quad (5)$$

In [27], Wuang and Huang presented a pitch adjustment with decreasing PAR, PAR_{min} set to 0 and PAR_{max} set to 1, as calculated in Equation (6). The pitch adjustment for x_i is given by expression Equation (7). The self-adaptive bandwidth bw is based on harmonic memory awareness, that is, depends on the extreme values of HM , where $min(HM)^i$ and $max(HM)^i$ are the minimum and maximum values of the ith variable in HM, respectively. Thus, the selection of PAR and bw values is eliminated.

$$PAR(g) = PAR_{max} - (PAR_{max} - PAR_{min}) \times \frac{g}{NI} \quad (6)$$

$$x_i = \begin{cases} HM(r,i) + (max(HM)^i - HM(r,i)) \times rand(0,1), & probability\ (0.5 \times PAR) \\ HM(r,i) + (min(HM)^i - HM(r,i)) \times rand(0,1), & probability\ (0.5 \times PAR) \\ HM(r,i), & probability\ (1 - PAR) \end{cases} \quad (7)$$

Omran and Mahdavi, inspired by PSO, proposed the random selection of variables from the best HM vector for the pitch adjustment, eliminating the bw parameter and adding a social dimension to the algorithm [19]. This is indicated in Equation (8), where

x_k^{best} is a random variable of the best solution vector. PAR grows linearly, as shown in Equation (3).

$$x_i = \begin{cases} x_k^{best}, & probability\ (PAR) \\ HM(r,i), & probability\ (1-PAR) \end{cases} \quad (8)$$

As can be seen, only two proposals for pitch adjustment have been brought to the multi-objective extensions of HS. In the case of the fixed pitch adjustment proposal, it is evident that both in single-objective and multi-objective optimization it has the disadvantage of an immobile exploration–exploitation balance during the entire execution. The proposals for pitch adjustment with dynamic bandwidth are based on the fact that as iterations pass, promising areas will be formed, and the exploitation will be more important. However, in the case of multi-objective problems, this behavior does not necessarily occur. Therefore, a proposal for pitch adjustment based on the real harmony distribution in the search space is required, such as the pitch adjustment by genotype.

2.2. Ranking

The main difference between the mono-objective HS and the multi-objective proposals is the ranking of harmonies. In mono-objective problems, the ranking of solutions by their quality is based on the objective function. However, there is a number of ranking strategies to impulse the generation of higher-quality Pareto fronts for multi-objective problems [30]. MOHS2 uses a non-dominated ranking, while MOSH3 and the proposed MOSHg algorithm apply the ranking presented by Deb et al. [31], which consists in:

1. Non-dominated ranking.
2. Ranking based on crowding distance by phenotype.

The non-dominated ranking classifies the solutions depending on their level of dominance, while the ranking by crowding is used as a second criteria for a second-level ranking related to the overcrowding of the solutions.

2.2.1. Non-Dominated Ranking

Given two solutions x_1 and x_2, x_1 dominates x_2 if and only if x_1 is not worse than x_2 for every objective function, and x_1 is strictly better than x_2 at least in one objective function. For the minimization case, it is equivalent to:

$$f_i(x_1) \leq f_i(x_2),\ i = 1, \ldots N$$

$$f_i(x_1) < f_i(x_2),\ for\ some\ i \in [1, \ldots N]$$

where N is the number of variables in a harmony.

If x_2 is not dominated by x_1 and x_1 is not dominated by x_2, then the solutions are non-dominated by each other. The number of times that a solution x_i is dominated by other solutions corresponds to its dominance level. The different dominance level can be obtained by a non-dominated ranking, that is, the different Pareto fronts of each solution [31–33].

Algorithm 1 presents the non-dominated ranking of HM. As can be seen, the dominance levels of each solution are saved in vector $Front$, and the non-dominated elements have a $Front$ value of 1 (optimal Pareto front).

Algorithm 1 Non-dominated ranking of HM

set N as number of harmonies;
$Front = \text{ones}(N,1)$ *vector of length N initialized with 1's*;
for $i \leftarrow 1$ *to* N **do**
  for $j \leftarrow 1$ *to* $N+1$ **do**
    if $HM(j,:) \prec HM(i,:)$ **then**
      $Front(i) = Front(i) + 1$; /* *where* $\prec$ *denotes dominance*
    else
      if $HM(i,:) \prec HM(j,:)$ **then**
        $Front(j) = Front(j) + 1$;

2.2.2. Crowding Distance by Phenotype

The crowding distance provides an estimate of the density of solutions surrounding a solution. It is used in multi-objective optimization to increase diversity, giving priority to the most dispersed solutions (with the highest crowding value). In other words, it encourages exploration in sparsely populated areas. It constitutes a second criterion for the ranking, since it ranks the solutions of the same front decreasingly with respect to the crowding distance. When this distance works in the space of the objective functions, it stimulates dispersion on the Pareto fronts, and is known in the field of bio-inspired algorithms as phenotype crowding [29,34]. It is expressed in Equation (9), where Nof is the number of objective functions; f_i^{j+1} and f_i^{j-1} are the neighbor points (posterior and previous, respectively) of the analyzed point $f_i^{(j)}$ with $j = 2, \ldots, (N-1)$; $f_i max$ and $f_i min$ are the maximum and minimum values of the ith objective function, respectively.

$$Cr = \sum_{i=1}^{Nof} \left(\frac{f_i^{j+1} - f_i^{j-1}}{f_i max - f_i min} \right) \tag{9}$$

Algorithm 2 corresponds to the crowding distance by phenotype, where Cr is the vector of crowding distancing, In is the position index of the ranked objective-function vector, Nof is the number of objective functions, and Fo is the matrix of values of the objective functions. Note that the crowding distance operates between solutions of the same Pareto front, that is, the crowding of the solutions of the first front only takes into account the solutions of the first front, and so on. The extreme solutions of each front will be infinite crowding values.

Algorithm 2 Crowding distance by phenotype

set N_{rank} as number of fronts;
$Cr = \text{zeros}(N,1)$ *vector of length N initialized with 0's*;
for $v \leftarrow 1$ *to* N_{rank} **do**
  $N_v = \text{sum}(Front == v)$; /**number of solutions of front v*
  $Fr_v = (Front == v)$; /**index vector of front v*
  $Cr = \text{zeros}(N_v, 1)$; /**crowding values of front v;*
  for $j \leftarrow 1$ *to* Nof **do**
    $Fo_v = Fo(Fr_v, j)$;
    $Fo_v = Fo_v / (max(Fo_v) - min(Fo_v))$; /**normalization*
    $In = \text{sort}(Fo_v)$; /**determination of neighboring points;*
    $Cr_v(In(1)) = Cr - v(In(end)) = \infty$; /**extreme values;*
    for $k \leftarrow 2$ *to* $N-1$ **do**
      $Cr_v(In(k)) = Cr_v(In(k)) + Fo_v(In(k+1)) - Fo_v(In(k-1))$;
  $Cr(Fr_v) = Cr_v$

2.3. MOHS2 and MOHS3 Algorithms

The MOHS2 and MOHS3 variants retain the original conception of the HS algorithm. Both alternatives were developed in a multi-objective way through the rankings described in the previous section. Both algorithms generate a random harmonic memory HM_1 in the

first iteration, and by means of this memory usage, pitch adjustment and randomization operators, they generate a new harmonic memory HM_2, with the same dimensions as the initial one. Both memories form an extended matrix HM_{1-2} that is ranked and truncated in half to form the HM_1 for the next iteration.

Algorithm 3 corresponds to MOHS2 [14]. Steps 1–4 constitute the initialization of the algorithm. The main cycle cover steps 5–14, where the new harmonic memory HM_2 is generated. Although MOHS2 uses only non-dominated ranking, in this work, the ranking proposed by Deb et al. [31] (step 15) is applied, because of the improvements experienced by the conformation of the fronts with the crowding distancing criterion. Finally, the harmonic memory HM_1 of the next iteration is formed in step 16.

Algorithm 3 MOHS2

```
1  define objective functions;
2  define HMCR, PAR, fw;                    /* fw: proportion parameter of bandwidth
3  generate randomly initial HM_1;
4  evaluate the objective functions with HM;
5  while g < max number of iterations do
6      while i <= N do
7          if rand < HMCR then
8              r = rand(1,k) ;
9              if rand < PAR then
10                 bw = (Ub_i − Lb_i) × fw/100;
11                 new X(i) = HM(r,i) + rand(−1,1) × bw(r,i);
12             else
13                 new X(i) = HM(r,i)
14         else
15             new X(i) = rand(Lb_i, Ub_i)
16     rank according to Deb et al. [31] to H_{1−2} = HM_1 ∪ HM_2 ;
17     truncate H and assign to HM_1
```

The MOHS3 pseudo-code [15] is shown in Algorithm 4. Steps 1–4 constitute its initialization (note that MOHS3 uses dynamic parameters in pitch adjustment according to Mahdavi et al. [16]). The main cycle cover steps 5–17, where the new harmonic memory HM_2 is generated. As mentioned before, the criterion of Deb is used to rank HM_{1-2} in step 18. Finally, the harmonic memory HM_1 of the next iteration is formed in step 19.

Algorithm 4 MOHS3

```
1  define objective functions;
2  define HMCR, PAR_min, PAR_max, bw_min, bw_max;
3  generate randomly initial HM_1;
4  evaluate the objective functions with HM ;
5  while g ≤ max number of iterations do
6      PAR = PAR_min + (PAR_max − PAR_min) × g/NI ;
7      bw = bw_max × exp(ln(bw_min/bw_max) × g/NI) ;
8      while H_2 is not complete do
9          while i ≤ N do
10             if rand < HMCR then
11                 r = rand(1,k) ;
12                 if rand < PAR then
13                     new X(i) = HM(r,i) + rand(−1,1) × bw(r,i);
14                 else
15                     new X(i) = HM(r,i)
16             else
17                 new X(i) = rand(Lb_i, Ub_i)
18     rank according to Deb et al. [31] to H_{1−2} = HM_1 ∪ HM_2 ;
19     truncate H and assign to HM_1
```

3. Proposed Pitch Adjustment by Genotype

The proposed pitch adjustment is inspired by genotype crowding distancing, which is also used in multi-objective optimization. These two operations are described below.

3.1. Crowding Distance by Genotype

When the crowding distance operates in the space of the decision variables, it encourages dispersion in the search space. In the field of bio-inspired algorithms, this operation is known as genotype crowding [29,34], and is expressed in Equation (10); where M is the number of variables; x_i^{j+1} and x_i^{j-1} are the neighboring points (after and before, respectively) of the point that is analyzed, x_i^j, for $j = 2, \ldots, (N-1)$; $x_i min$ and $x_i max$ are the minimums and maximums of the variable ith of the HM.

$$Cr = \sum_{i=1}^{M} \left(\frac{x_i^{j+1} - x_i^{j-1}}{x_i max - x_i min} \right) \tag{10}$$

It is important to note that the proximity of two solutions on the objective space does not necessarily imply proximity in the search space, as shown in Figure 1. Pareto fronts can be generated for different dispersion degrees of the search space. In [29], a deep analysis of multi-objective problems solved by NSGA with crowding by genotype and phenotype can be found.

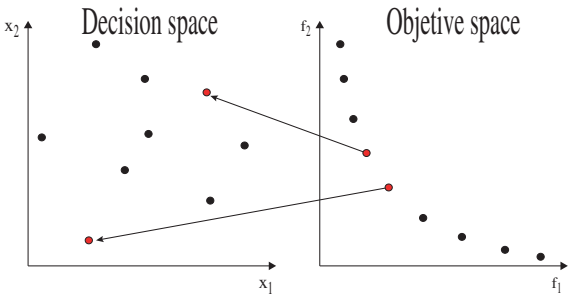

Figure 1. An example with two neighboring solutions in the front that are distant from each other in the search space.

The crowding distance by genotype is determined in a very similar way to the crowding by phenotype previously exposed, with the difference that it is calculated between the variables, as described in Algorithm 5. The number of objective functions Nof was substituted by the number of variables M in step 7, while in steps 8–10 and 13, the matrix of objective functions was replaced by HM.

Algorithm 5 Crowding distance by genotype

```
1  set N_rank as number of fronts;
2  Cr = zeros(N,1) vector of length N initialized with 0's;
3  for v ← 1 to N_rank do
4      N_v = sum(Front == v);                    /*number of solutions of front v
5      Fr_v = (Front == v) ;                     /*index vector of front v
6      Cr = zeros(N_v,1);                        /*crowding values of front v;
7      for j ← 1 to M do
8          HM_v = HM(Fr_v, j) ;
9          HM_v = HM_v/(max(HM_v) − min(HM_v));   /*normalization
10         In = sort(HM_v);                      /*determination of neighboring points;
11         Cr_v(In(1)) = Cr − v(In(end)) = ∞;     /*extreme values;
12         for k ← 2 to N − 1 do
13             Cr_v(In(k)) = Cr_v(In(k)) + HM_v(In(k + 1)) − HM_v(In(k − 1));
14     Cr(Fr_v) = Cr_v
```

3.2. Pitch Adjustment by Genotype

According to the previous subsection, the Pareto-optimal front can be reflected in the search space both by dispersed solutions and by close solutions that form promising areas. This can be determined by means of crowding distance by genotype, opening two possibilities:

1. Intensify exploitation as the algorithm advances, but this time depending on the conformation of promising areas.
2. Maintain higher exploration in dispersed solutions and higher exploitation in promising areas, with the objective of achieving a better balance of the algorithm, based on the behavior of HM during the formation of the Pareto-optimal front.

Therefore, a pitch adjustment based on the crowding distance by genotype is proposed as established in expression (11), where bw is obtained from Equation (12):

$$x_i = \begin{cases} HM(r,i) + rand(-1,1) \times bw(r,i), & probability\ (PAR) \\ HM(r,i), & probability\ (1-PAR) \end{cases} \tag{11}$$

$$bw(r,i) = \frac{x_i^{j+1} - x_i^{j-1}}{2} \tag{12}$$

For the upper and lower extremes of each front, bw is given by Equations (13) and (14), respectively:

$$bw(r,i) = x_i - x_i^{j-1} \tag{13}$$

$$bw(r,i) = x_i^{j+1} - x_i \tag{14}$$

Since the bandwidth $bw(r,i)$ is the crowding distance component of the variable x_i for the solution vector in the position r, Equation (12) includes no summation operator. That is, x_i may vary by adjusting the pitch up to a range equal to the distance from the neighboring points. An example considering an HM with four harmonies is shown in expression (15), where a non-dominated ranking was performed and harmonies 1,2,4 turned out to be non-dominated (*Front 1*) while harmony 3 was dominated only once (*Front 2*):

$$HM = \begin{bmatrix} 9 & 7 & 1 \\ 3 & 1 & 9 \\ 6 & 2 & 4 \\ 2 & 5 & 8 \end{bmatrix} \begin{matrix} \rightarrow & Harmony\ 1 & (Front\ 1) \\ \rightarrow & Harmony\ 2 & (Front\ 1) \\ \rightarrow & Harmony\ 3 & (Front\ 2) \\ \rightarrow & Harmony\ 4 & (Front\ 1) \end{matrix} \tag{15}$$

Suppose that a variable x_2 in the new harmony is created by pitch adjustment, and the parameter r is randomly generated with a value 4. The substitution of these values in expression (11) produces Equation (16), for calculating x_2:

$$x_2 = HM(4,2) + rand(-1,1) \times bw(4,2) \tag{16}$$

Vector 4 is a part of *Front* 1. Since the analyzed variable is $x_2^j = HM(4,2) = 5$, the neighboring points are 7 and 1, from harmonies 1 and 2, respectively. Equations (17) and (18) are generated by substituting these values in Equations (12) and (16). Note that solution vector 3 is not considered because it belongs to *Front* 2.

$$bw(4,2) = \frac{(x_2^{j+1} - x_2^{j-1})}{2} = \frac{(7-1)}{2} \tag{17}$$

$$x_2 = 5 + rand(-1,1) \times 3 \tag{18}$$

4. Proposed HS (MOHSg)

The effectiveness of metaheuristic algorithms is driven by two fundamental components: exploration and exploitation. The proper balance between these two components

greatly influences the algorithm efficiency. Highly exploit-focused algorithms explore only a fraction of the search space and tend to become stuck in the local optimum. On the other hand, highly scan-focused algorithms converge very slowly, and the solution time can be very long.

HS proposals with a fixed exploration–exploitation balance are unable to adjust their behavior as required by the problem. On the other hand, the proposals with variable parameters do not contemplate the distribution of solutions in the search space, which, as seen above, can have different configurations for the formation of the Pareto-optimal front. Instead, the proposal presented in this paper is capable of balancing the exploration and exploitation by adjusting the pitch, based on the distribution of the solutions in the search space. Therefore, it does not require parameters such as the static bandwidth used in MOHS2, nor dynamic parameters such as those required by MOHS3.

Description

The main differences between MOHSg (described by Algorithm 6) and the original mono-objective HS version lie in the HM ranking and in the pitch adjustment operation by genotype. The algorithm requires the presetting of the harmonic memory consideration $HMCR$ and pitch adjustment PAR parameters. The execution starts generating a random initial harmonic memory HM_1. For each iteration, a new harmonic memory HM_2 is generated with the same dimensions as the initial one and is made up variable by variable applying the memory usage operators, pitch adjustment by genotype and randomization. Both memories are combined to form an extended matrix HM_{1-2} that is ranked according to the ranking criteria of Deb et al. [31]. Finally, the ranked matrix is truncated in half to form the HM_1 of the next iteration.

Algorithm 6 Proposed algorithm MOHSg

define objective functions;
define $HMCR$, PAR;
generate randomly an initial HM_1;
evaluate the objective functions with HM_1 ;
rank HM_1 according to Deb et al. [31] ;
generate bw by Cr, **applying the crowding with genotype described in Algorithm 5**;
while $g \leq$ *max number of iterations* **do**
  while H_2 *is not complete* **do**
    while $i \leq N$ **do**
      if $rand < HMCR$ **then**
        $r = rand(1, k)$;
        if $rand < PAR$ **then**
          $new\ X(i) = HM(r, i) + rand(-1, 1) \times bw(r, i)$;
        else
          $new\ X(i) = HM(r, i)$
      else
        $new\ X(i) = rand(Lb_i, Ub_i)$
  rank $H_{1-2} = HM_1 \cup HM_2$, **according to Deb et al. [31]**;
  truncate H_{1-2} and assign to HM_1 ;
  generate bw by Cr with genotype

5. Experimentation and Results

In this section, the MOHS2, MOHS3 and MOHSg algorithms are used to solve six problems reported in the literature [35,36], designated as P1 to P6, that are specifically designed for measuring the performance of multi-objective optimization algorithms. The problems were selected taking into account the variety of their characteristics, such as (I) the shape of the Pareto-optimal fronts (PFs) and the Pareto-optimal sets (PSs), (II) the coexistence of local and global PSs (multi-modality) (III) the number of decision variables and objective functions. In all the problems presented, it is possible to determine the real PS and the real PF, which allows the evaluation of the results obtained by each algorithm. The algorithms were programmed in Matlab R2018a on a Windows 10 platform. Computational experi-

ments were performed on a PC with a 2.67 GHz Intel(R) Core (TM) i7 processor and 8 GB of RAM. The test problems are included in Appendix A.

5.1. Performance Indicators

Every problem was solved with 20 independent runs of each algorithm. An execution includes 10,000 evaluations of the objective function, with the exception of problem P6, which required 20,000 evaluations per run. In fact, the number of objective-function evaluations measures the computational cost. For the graphic appreciation of the results, the integrations of PSs and PFs of the 20 executions are made. In each figure, the number of solution vectors that make up each front is specified. Additionally to the graphical appreciation, the performance indicators described in the following subsection were applied for the quantitative comparison of the results. The performance indicators were calculated for each execution, and presented through its mean value (average), best value (best), worst value (worst), and standard deviation (Std. Dev). The best results are highlighted. Considering that metrics can be misleading in multi-objective optimization according to Coello and Cortés [37], it can be illuminating to consider two main factors: (I) if the solutions belong to the real PF, and (II) how uniform the distribution of solutions along the Pareto front is.

5.1.1. Error Ratio, ER

This parameter (Equation (19)) was proposed by Van Veldhuizen [38] to indicate the percentage of solutions from the non-dominated vectors that are not in the real PF, where n is the number of non-dominated solution vectors that were generated, and e_i is 0 or 1 if the vector is non-dominated by the PF real or not, respectively. The ideal value is $ER = 0$ since every vector generated by the algorithm belongs to the real PF.

$$ER = \frac{\sum_{i=1}^{n} e_i}{n} \tag{19}$$

5.1.2. Spacing, S

This indicator was proposed for Schott [39] as a way to measure the variance of neighboring vectors in the PF. It is calculated by Equations (20) and (21), where d is the media of all d_i, with $i, j = 1, \ldots, n$,

$$S = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (d - d_i)^2} \tag{20}$$

$$d_i = min_j(|f_1^i - f_1^j| + |f_2^i - f_2^j|) \tag{21}$$

5.1.3. Inverter Generational Distance, GD

This indicator was introduced by Van Veldhuizen and Lamont [40] as a way of estimating how far the elements of the PF obtained by the algorithm are from the real PF. It is represented by Equation (22), where n is the number of generated vectors, and d_i is the Euclidian distance between the generated vectors and the ones of the real PF.

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{22}$$

Coello and Cortés [37] recommend using the real PF as a reference, that is, each vector of the real PF is measured with its nearest neighbor of the PF obtained to avoid measurement problems when the generated front has few members, and this is what it is known as Inverted generational distance. In this work, GD was used both in the search space (GDx) and in the space of objective functions (GDf).

5.2. Parameter Tuning

In order to achieve a reasonably good performance for the three algorithms, several previous experiments were carried out, modifying their parameters until producing the best performances. For the case of MOHS2 and MOHS3, it started from the parameters recommended in the original developments [14,15]. As can be verified in Table 1, for MOHS2, only the *HMCR* parameter was varied, while MOHS3 kept the parameters proposed in the original work. For the case of P6, which has three objective functions (many-objective), the parameters of the three algorithms were modified in order to obtain a better exploration, as shown in Table 2.

Table 1. Parameters for problems P1 to P5.

	Value		
Parameter	**MOHSg**	**MOHS2**	**MOHS3**
$HMCR$	0.95	0.70	0.85
PAR_{min} PAR_{max}	0.80	0.70	0.20 2.00
bw_{min} bw_{max}	—	1%	0.45 0.90

Table 2. Parameters for problem P6.

	Value		
Parameter	**MOHSg**	**MOHS2**	**MOHS3**
$HMCR$	0.40	0.50	0.70
PAR_{min} PAR_{max}	0.60	0.10	0.80 2.00
bw_{min} bw_{max}	—	1%	0.45 0.90

5.3. Analysis of Results

5.3.1. Problem P1

Figure 2 shows the integrated PFs resulting from the solution of problem P1, as well as its real PF (convex front). The legend indicates the number of points that make up each front. Note that all the algorithms converged to the solution of the real PF. However, in the detail view, points appear belonging to the three algorithms (specially the MOHS3 algorithm) that do not belong to the real PF, that is, they are dominated by that front. In the search space, it can be observed that the PS has an abrupt change in behavior (Figure 3). The algorithms found most of the solutions in the left region of the set, while in the right side, a higher dispersion is observed, especially for the proposed algorithm MOHSg.

Table 3 shows the statistical analysis of the behavior of the algorithms for P1, where the best values are indicated in bold type. As can be seen in the error rates, an average of 21.8% of the vectors obtained by MOHSg do not belong to the real PF, while for MOHS2 and MOHS3 they constitute 28% and 41%, respectively. Likewise, it can be observed that, in general, the PFs and PSs nearest to the real PF and PS were obtained by MOHS3, with the drawback that approximately half of its points are dominated by the real PF. The best distributed fronts correspond to the proposed MOHSg algorithm, followed by MOHS2 and MOHS3.

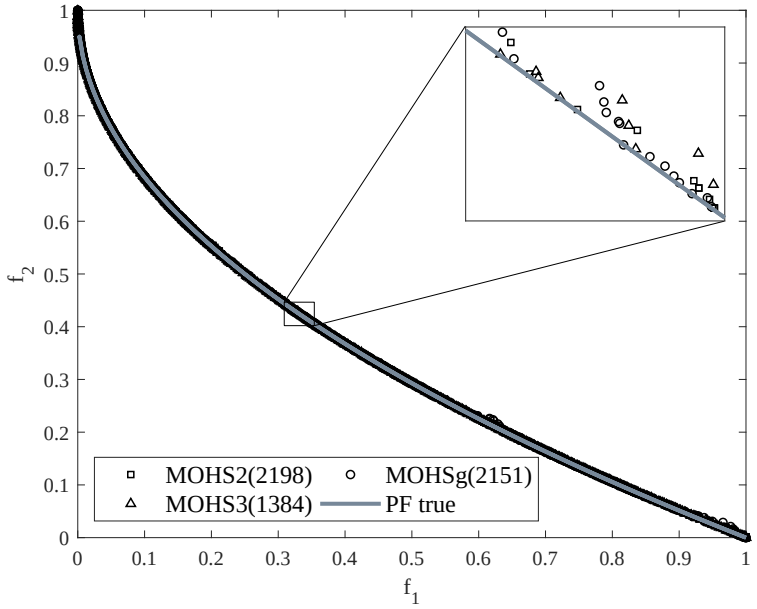

Figure 2. Pareto fronts for problem P1.

Then, it is concluded that for this test function, MOHS3 presented a poor performance because it produced the PFs with the lowest population and with the highest proportions dominated by the real PF. MOHSg had a slight superiority in both the error ratio and the uniform distribution with respect to MOHS2. However, this last algorithm surpassed the proposed one in terms of the proximity in relation to PS and real PF. Thus, MOHS2 and MOHSg presented a similar performance for this problem.

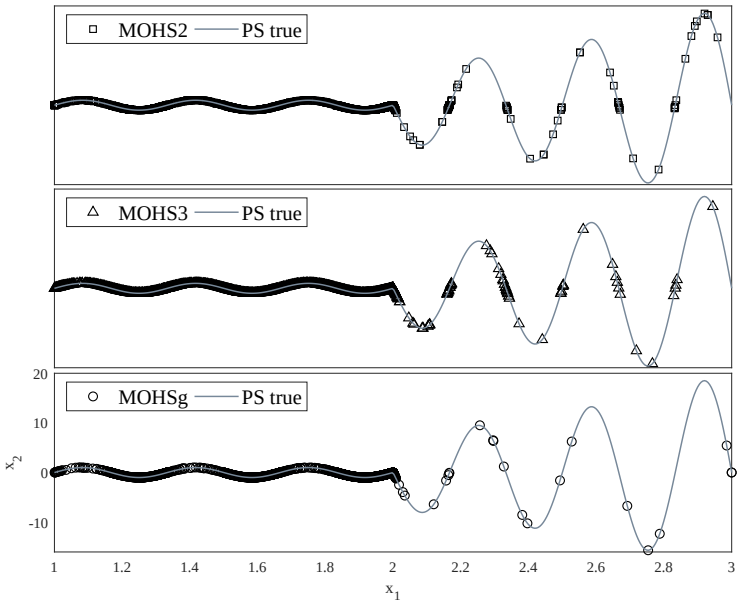

Figure 3. Pareto sets for problem P1.

5.3.2. Problem P2

In Figure 4, the concave PF generated for problem P2 can be seen. The most populated front corresponds to the MOHS2 algorithm, followed by MOHS3 and MOHSg very close to each other. Note that in this problem, the concave front is made up of two PSs in the search space, as shown in Figure 5. The PFs and PSs obtained are more populated and better distributed than in P1, as can be seen in the results tabulated below.

Table 4 presents the statistical analysis of the algorithms. It can be observed that the best error ratio corresponds to the MOHS2 algorithm with 7.65% (mean value) followed by MOHSg with 11.95% and MOHS3 with 12.83%. The fronts and sets nearest to the real PF and PS correspond to those obtained by MOHS3, followed by the results of MOHS2 and MOHSg. The best distribution of the PFs corresponds to MOHS2, followed by the proposed algorithm MOHSg and MOHS3. In this problem, MOHS2 produced the most populated integrated front, and the fronts with the lowest error ratio and with the best uniform distribution of all the variants. Therefore, it can be concluded that MOHS2 had the best overall performance for this test function.

Table 3. Statistic analysis for problem P1.

		MOHS2	MOHS3	MOHSg
Error ratio	Average	2.800×10^{-1}	4.105×10^{-1}	$\mathbf{2.183 \times 10^{-1}}$
	Best	$\mathbf{1.500 \times 10^{-1}}$	3.350×10^{-1}	$\mathbf{1.500 \times 10^{-1}}$
	Worst	4.300×10^{-1}	4.950×10^{-1}	$\mathbf{2.900 \times 10^{-1}}$
	Std. Dev.	7.222×10^{-2}	4.322×10^{-2}	$\mathbf{3.945 \times 10^{-2}}$
GDf	Average	1.145×10^{-2}	$\mathbf{4.401 \times 10^{-3}}$	1.545×10^{-2}
	Best	6.990×10^{-3}	$\mathbf{3.662 \times 10^{-3}}$	7.658×10^{-3}
	Worst	2.291×10^{-2}	$\mathbf{5.560 \times 10^{-3}}$	2.469×10^{-2}
	Std. Dev.	3.748×10^{-3}	$\mathbf{5.009 \times 10^{-4}}$	4.787×10^{-3}
GDx	Average	18.21×10^{-1}	$\mathbf{17.85 \times 10^{-1}}$	21.31×10^{-1}
	Best	11.47×10^{-1}	$\mathbf{7.248 \times 10^{-1}}$	8.722×10^{-1}
	Worst	$\mathbf{25.56 \times 10^{-1}}$	32.58×10^{-1}	36.73×10^{-1}
	Std. Dev.	$\mathbf{3.746 \times 10^{-1}}$	7.091×10^{-1}	7.283×10^{-1}
Spacing	Average	1.389×10^{-2}	17.39×10^{-1}	$\mathbf{1.151 \times 10^{-2}}$
	Best	5.869×10^{-3}	$\mathbf{5.434 \times 10^{-3}}$	7.874×10^{-3}
	Worst	5.500×10^{-2}	1.097×10^{-1}	$\mathbf{2.276 \times 10^{-2}}$
	Std. Dev.	1.269×10^{-2}	31.22×10^{-1}	$\mathbf{3.980 \times 10^{-3}}$

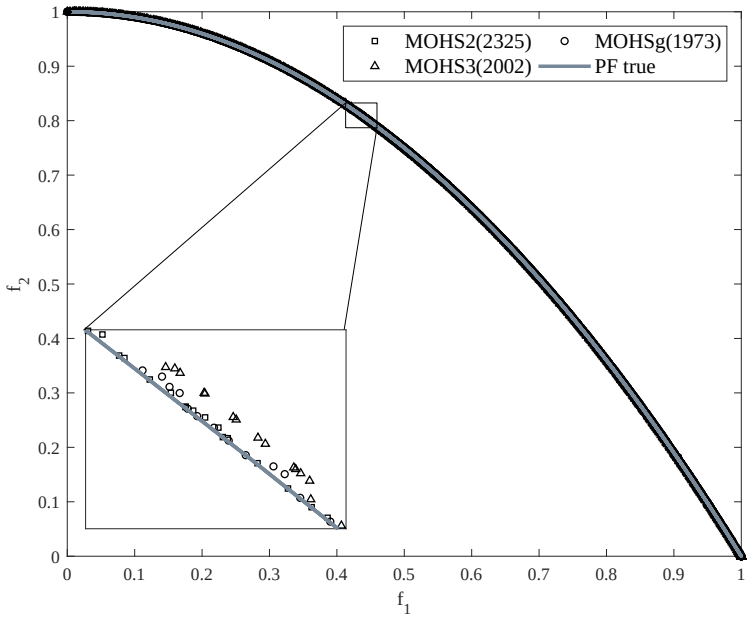

Figure 4. Pareto fronts for problem P2.

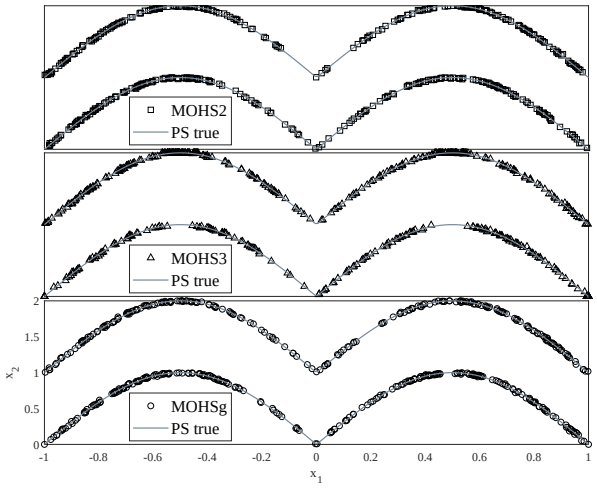

Figure 5. Pareto sets for problem P2.

5.3.3. Problem P3

As can be seen in Figure 6, problem P3 has a local PF and a global PF, which drives the algorithms to be trapped in the local optimum. Both fronts are discontinuous and are divided into four solution regions. In Figure 7, the local and global PSs for this problem can be seen, divided into four regions and with a linear behavior. The most populated PFs corresponded to MOHSg and MOHS2 with 3919 and 3863 points, respectively, while MOHS3 obtained a much lower front with 1013 points. The mentioned figures show the convergence of the three algorithms to global solutions. The lowest error ratio in Table 5 corresponds to MOHSg with 1.43%, followed by MOHS2 with 2.95 % and MOHS3 with an extremely unfavorable 77.9%. The fronts nearest to the real PF and PS also correspond to MOHSg followed by MOHS2 and MOHS3. In the case of distribution, MOHSg had the best average performance, although the lowest standard deviation corresponds to MOHS2. In this problem, MOHSg clearly presented the best performance.

Table 4. Statistic analysis for problem P2.

		MOHS2	MOHS3	MOHSg
Error ratio	Average	$\mathbf{7.650 \times 10^{-2}}$	1.283×10^{-1}	1.195×10^{-1}
	Best	$\mathbf{4.000 \times 10^{-2}}$	8.000×10^{-2}	9.000×10^{-2}
	Worst	$\mathbf{1.150 \times 10^{-1}}$	1.650×10^{-1}	1.650×10^{-1}
	Std. Dev.	$\mathbf{1.623 \times 10^{-2}}$	2.551×10^{-2}	1.884×10^{-2}
GDf	Average	2.635×10^{-3}	$\mathbf{2.457 \times 10^{-3}}$	2.699×10^{-3}
	Best	2.287×10^{-3}	$\mathbf{2.194 \times 10^{-3}}$	2.242×10^{-3}
	Worst	3.027×10^{-3}	$\mathbf{2.850 \times 10^{-3}}$	3.303×10^{-3}
	Std. Dev.	2.208×10^{-4}	$\mathbf{1.627 \times 10^{-4}}$	2.470×10^{-4}
GDx	Average	2.948×10^{-2}	$\mathbf{2.863 \times 10^{-2}}$	3.848×10^{-2}
	Best	$\mathbf{2.594 \times 10^{-2}}$	2.640×10^{-2}	3.085×10^{-2}
	Worst	3.563×10^{-2}	$\mathbf{3.199 \times 10^{-2}}$	4.948×10^{-2}
	Std. Dev.	2.441×10^{-3}	$\mathbf{1.795 \times 10^{-3}}$	3.963×10^{-3}
Spacing	Average	$\mathbf{3.474 \times 10^{-3}}$	2.904×10^{-2}	1.639×10^{-2}
	Best	2.777×10^{-3}	$\mathbf{2.735 \times 10^{-3}}$	2.757×10^{-3}
	Worst	$\mathbf{6.162 \times 10^{-3}}$	1.162×10^{-1}	9.262×10^{-2}
	Std. Dev.	$\mathbf{6.972 \times 10^{-4}}$	3.555×10^{-2}	2.535×10^{-2}

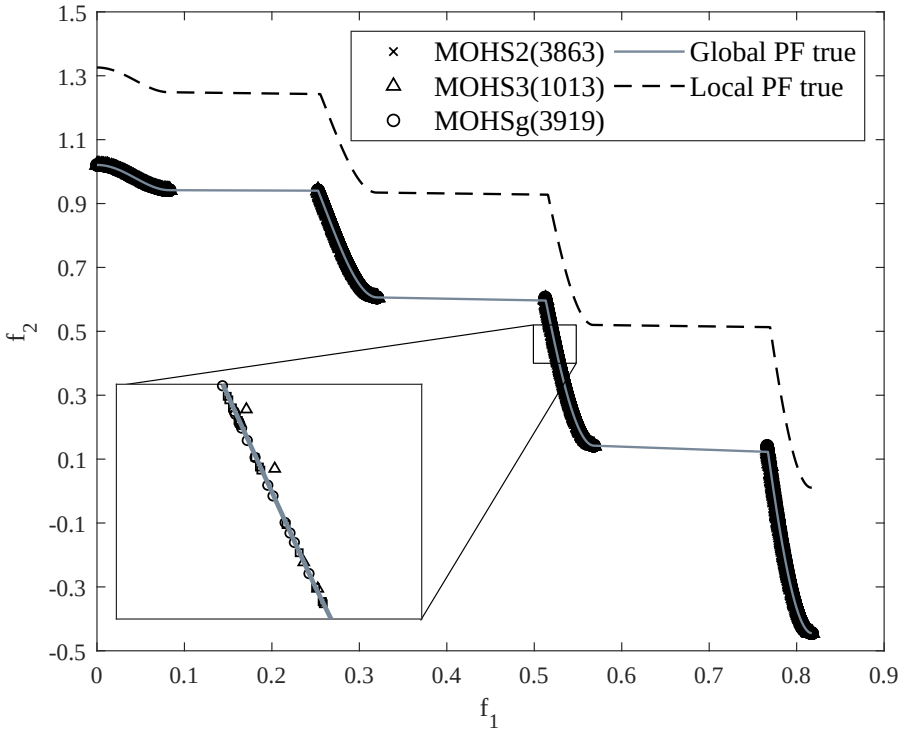

Figure 6. Pareto fronts for problem P3.

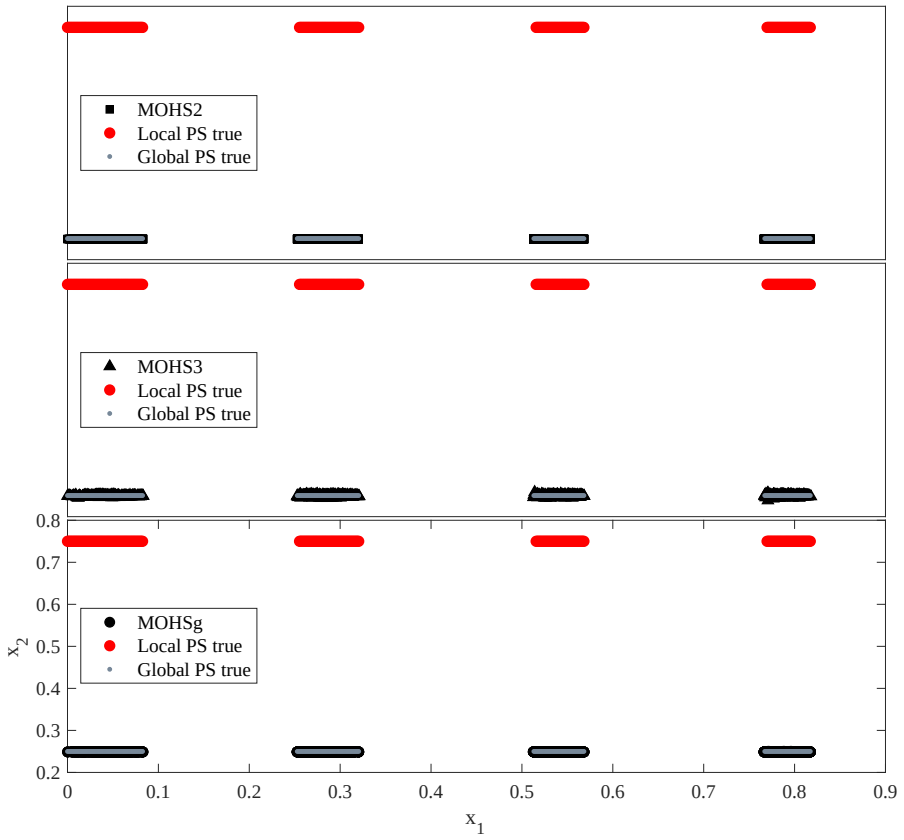

Figure 7. Pareto sets for problem P3.

Table 5. Statistic analysis for problem P3.

		MOHS2	MOHS3	MOHSg
Error ratio	Average	2.950×10^{-2}	7.790×10^{-1}	$\mathbf{1.425 \times 10^{-2}}$
	Best	1.500×10^{-2}	4.400×10^{-1}	$\mathbf{5.000 \times 10^{-3}}$
	Worst	6.500×10^{-2}	9.700×10^{-1}	$\mathbf{4.000 \times 10^{-2}}$
	Std. Dev.	1.297×10^{-2}	1.453×10^{-1}	$\mathbf{7.993 \times 10^{-3}}$
GDf	Average	2.564×10^{-3}	4.829×10^{-3}	$\mathbf{2.414 \times 10^{-3}}$
	Best	2.273×10^{-3}	4.110×10^{-3}	$\mathbf{2.179 \times 10^{-3}}$
	Worst	$\mathbf{2.984 \times 10^{-3}}$	5.993×10^{-3}	3.079×10^{-3}
	Std. Dev.	$\mathbf{1.791 \times 10^{-4}}$	4.899×10^{-4}	2.465×10^{-4}
GDx	Average	1.623×10^{-3}	3.204×10^{-3}	$\mathbf{1.544 \times 10^{-3}}$
	Best	1.522×10^{-3}	1.869×10^{-3}	$\mathbf{1.442 \times 10^{-3}}$
	Worst	1.816×10^{-3}	5.904×10^{-3}	$\mathbf{1.664 \times 10^{-3}}$
	Std. Dev.	6.387×10^{-3}	1.127×10^{-3}	$\mathbf{6.139 \times 10^{-5}}$
Spacing	Average	4.260×10^{-3}	6.460×10^{-2}	$\mathbf{3.005 \times 10^{-2}}$
	Best	3.679×10^{-3}	4.906×10^{-3}	$\mathbf{2.615 \times 10^{-3}}$
	Worst	4.780×10^{-3}	1.322×10^{-1}	$\mathbf{3.843 \times 10^{-2}}$
	Std. Dev.	$\mathbf{2.677 \times 10^{-4}}$	1.759×10^{-2}	2.921×10^{-2}

5.3.4. Problem P4

Like in the previous case, problem P4 has a discontinuous front, but this time divided into 10 regions (Figure 8). In Figure 9, it can be seen that the PS is also divided into 10 linear regions, but in this problem, there is only one global front. As shown in Table 6, the three algorithms generated fronts of approximately 3900 points, for an error ratio in the best of cases of 1.38% for MOHSg followed by 2.18% and 2.70% for MOHS3 and MOHS2, respectively. The fronts and sets nearest to the real PF and PS corresponded to MOHS3, followed by MOHSg and MOHS2. Similarly, the best distributed fronts were obtained using MOHS3, while MOHSg showed very close values.

In this case, the MOHS3 algorithm presented a fairly low error ratio, as well as the fronts and sets nearest to the real PF and PS. It also generated the best uniform distribution values of PF. Therefore, it is concluded that MOHS3 offered the best performance for this test function.

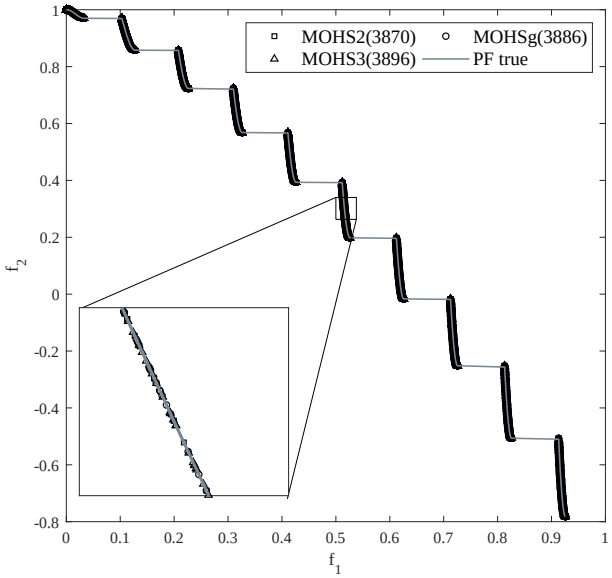

Figure 8. Pareto fronts for problem P4.

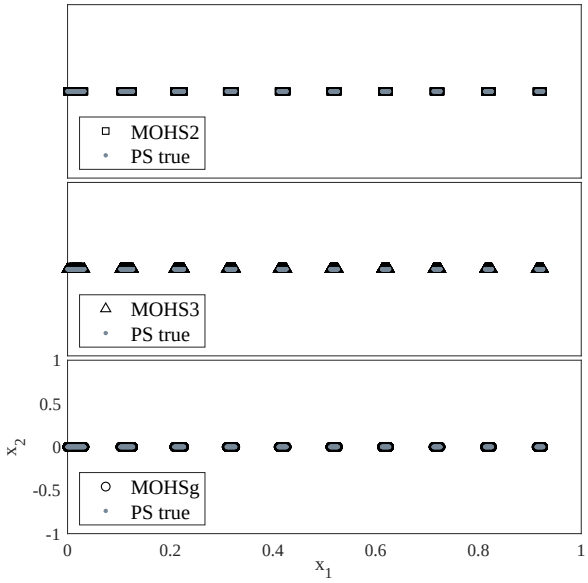

Figure 9. Pareto sets for problem P4.

5.3.5. Problem P5

Unlike the previous cases, problem P5 has three decision variables. In addition, it presents two PSs (one local and one global) that make up a local and global PF, as shown in Figures 10 and 11. The most populated front corresponds to the proposed algorithm MOHSg (3402 points), followed by MOHS2 (2987), while MOHS3 produced a significantly smaller front (1438). As can be seen in Table 7, the error ratio also differs drastically, with the best in 8.95% for MOHSg, followed by 18.58% for MOHS2, and in the worst case MOHS3 returned a value of 70%. The PF closest to the real one was obtained by MOHS2, closely followed by MOHSg, while the PS closest to the real one was obtained by MOHS3. The PFs with the best uniform distribution (mean value) were obtained by MOHSg.

For this problem, MOHSg obtained a much more populated front than MOHS2 or MOHS3, as well as a significantly lower error ratio. In Figure 11, it can be contrasted that MOHSg does not cover all the real PS, and it is also manifested in the GDx values. However, it has the best average value of uniform distribution in the PFs. Therefore, it can be concluded that MOHSg has the best overall performance for this problem.

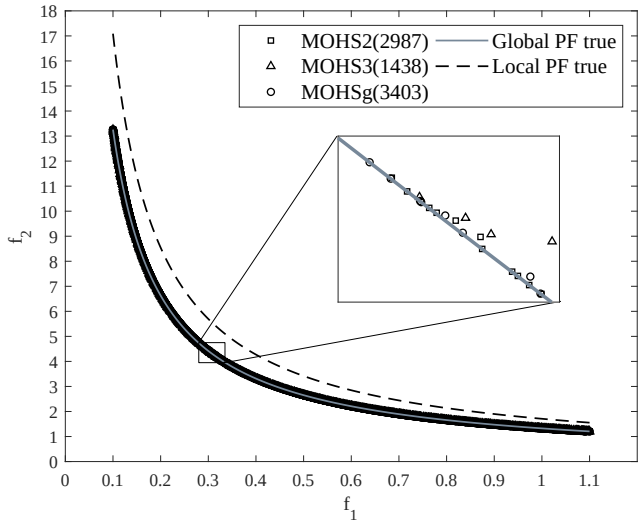

Figure 10. Pareto fronts for problem P5.

Table 6. Statisticalanalysis for problem P4.

		MOHS2	MOHS3	MOHSg
Error ratio	Average	2.725×10^{-2}	2.175×10^{-2}	$\mathbf{1.375 \times 10^{-2}}$
	Best	5.000×10^{-3}	5.000×10^{-3}	**0.000**
	Worst	4.500×10^{-2}	3.500×10^{-2}	$\mathbf{3.000 \times 10^{-2}}$
	Std. Dev.	1.118×10^{-2}	$\mathbf{7.993 \times 10^{-3}}$	8.867×10^{-3}
GDf	Average	3.280×10^{-3}	$\mathbf{2.643 \times 10^{-3}}$	3.203×10^{-3}
	Best	3.036×10^{-3}	$\mathbf{2.502 \times 10^{-3}}$	2.683×10^{-3}
	Worst	3.586×10^{-3}	$\mathbf{2.903 \times 10^{-3}}$	4.612×10^{-3}
	Std. Dev.	1.570×10^{-4}	$\mathbf{1.010 \times 10^{-4}}$	5.389×10^{-4}
GDx	Average	7.234×10^{-4}	$\mathbf{5.239 \times 10^{-4}}$	6.242×10^{-4}
	Best	6.314×10^{-4}	$\mathbf{4.633 \times 10^{-4}}$	5.058×10^{-4}
	Worst	9.297×10^{-4}	$\mathbf{6.222 \times 10^{-4}}$	7.374×10^{-4}
	Std. Dev.	7.805×10^{-5}	$\mathbf{3.946 \times 10^{-5}}$	6.383×10^{-5}
Spacing	Average	5.168×10^{-3}	$\mathbf{4.420 \times 10^{-3}}$	4.479×10^{-3}
	Best	4.811×10^{-3}	$\mathbf{3.359 \times 10^{-3}}$	3.555×10^{-3}
	Worst	$\mathbf{5.547 \times 10^{-3}}$	6.183×10^{-3}	6.416×10^{-3}
	Std. Dev.	$\mathbf{2.198 \times 10^{-4}}$	8.600×10^{-4}	7.011×10^{-4}

Table 7. Statistical analysis for problem P5.

		MOHS2	MOHS3	MOHSg
Error ratio	Average	1.858×10^{-1}	7.023×10^{-1}	$\mathbf{8.950 \times 10^{-2}}$
	Best	6.000×10^{-2}	6.200×10^{-1}	$\mathbf{3.000 \times 10^{-2}}$
	Worst	3.700×10^{-1}	7.550×10^{-1}	$\mathbf{1.650 \times 10^{-1}}$
	Std. Dev.	1.001×10^{-1}	$\mathbf{3.683 \times 10^{-2}}$	4.273×10^{-2}
GDf	Average	$\mathbf{1.287 \times 10^{-2}}$	1.936×10^{-2}	1.356×10^{-2}
	Best	$\mathbf{1.206 \times 10^{-2}}$	1.690×10^{-2}	1.265×10^{-2}
	Worst	$\mathbf{1.457 \times 10^{-2}}$	2.232×10^{-2}	1.418×10^{-2}
	Std. Dev.	$\mathbf{5.929 \times 10^{-4}}$	1.244×10^{-3}	6.003×10^{-4}
GDx	Average	2.528×10^{-2}	$\mathbf{3.960 \times 10^{-2}}$	1.261×10^{-1}
	Best	3.770×10^{-2}	$\mathbf{3.570 \times 10^{-2}}$	9.731×10^{-2}
	Worst	1.345×10^{-1}	$\mathbf{4.479 \times 10^{-2}}$	1.776×10^{-1}
	Std. Dev.	2.842×10^{-2}	$\mathbf{2.497 \times 10^{-3}}$	2.201×10^{-2}
Spacing	Average	3.280×10^{-2}	6.154×10^{-2}	$\mathbf{2.861 \times 10^{-2}}$
	Best	$\mathbf{2.366 \times 10^{-2}}$	4.690×10^{-2}	2.457×10^{-2}
	Worst	5.100×10^{-2}	7.738×10^{-2}	$\mathbf{3.439 \times 10^{-2}}$
	Std. Dev.	8.211×10^{-3}	8.395×10^{-3}	$\mathbf{2.430 \times 10^{-3}}$

5.3.6. Problem P6

The MMF14_a problem has three decision variables and three objective functions, making it a many-objective problem. As explained above, in this problem the number of evaluations was modified to 20,000 and the parameters of the three algorithms were tuned in order to improve the quality of the solutions. Note that this problem has two global surfaces of PSs that correspond to a single concave global PF (Figures 12 and 13). Additionally, the generated PFs converge to the real PF, also covering the two surfaces that make up the PSs. The most populated front was obtained by MOHS3, followed by MOHS2 and MOHSg. The best error ratio corresponds to MOHS3 with 34.95%, while for MOHS2 and MOHSg the error ratios were 47% and 50%, respectively, as indicated in Table 8. The front closest to the real PF was also obtained from the MOHS3 algorithm, while the sets closest to the real PS were obtained by MOHSg. In the case of the uniform distribution of the PF, the best mean value corresponds to

MOHS3 while the best recorded point value was obtained by MOHSg. Thus, MOHS3 produced the best solution, since it generated the most populated front of the three algorithms, as well as the lowest error ratio by a considerable margin. This algorithm also yielded the best mean GDf and uniform distribution values.

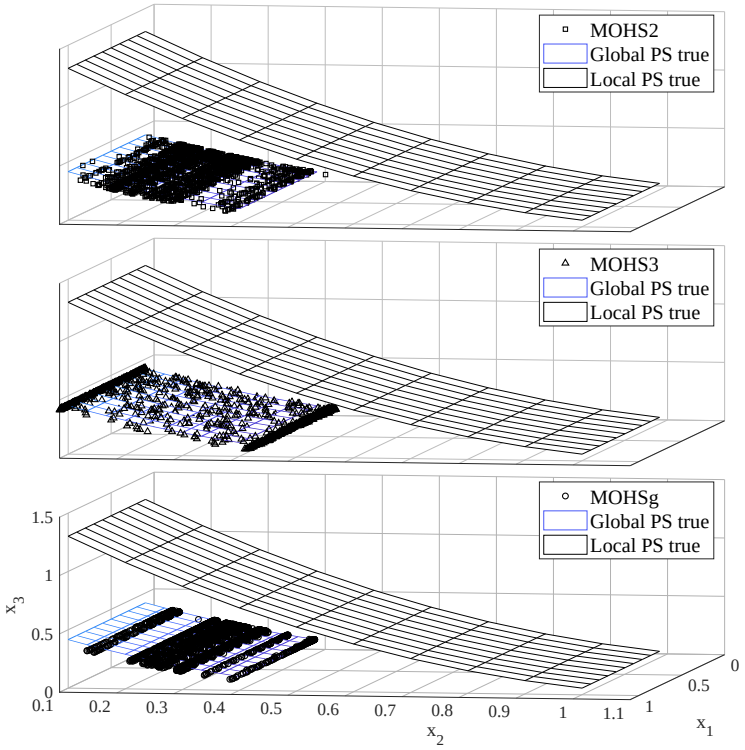

Figure 11. Pareto sets for problem P5.

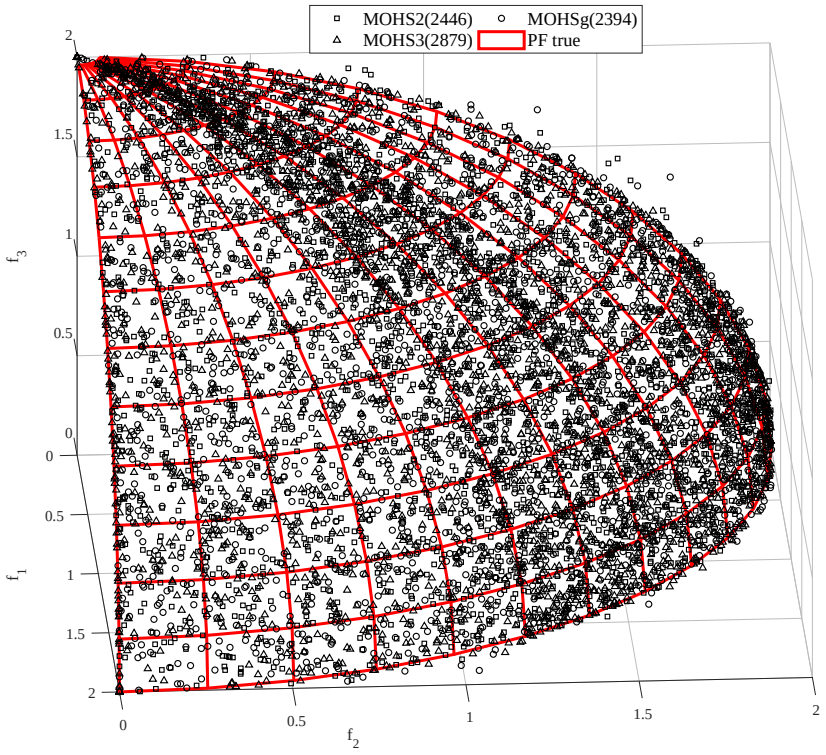

Figure 12. Pareto fronts for problem P6.

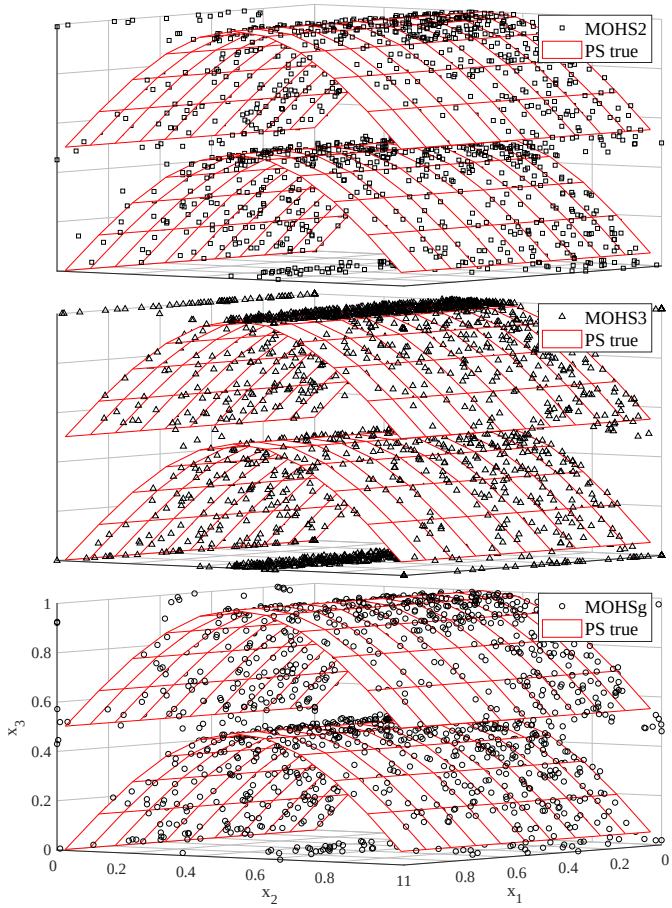

Figure 13. Pareto sets for problem P6.

Table 8. Statistic analysis for problem P6.

		MOHS2	MOHS3	MOHSg
Error ratio	Average	4.710×10^{-1}	$\mathbf{3.495 \times 10^{-1}}$	4.998×10^{-1}
	Best	3.900×10^{-1}	$\mathbf{2.950 \times 10^{-1}}$	4.400×10^{-1}
	Worst	5.300×10^{-1}	$\mathbf{4.000 \times 10^{-1}}$	5.700×10^{-1}
	Std. Dev.	3.813×10^{-2}	$\mathbf{2.946 \times 10^{-2}}$	3.299×10^{-2}
GDf	Average	9.482×10^{-2}	$\mathbf{9.229 \times 10^{-2}}$	9.464×10^{-2}
	Best	9.129×10^{-2}	$\mathbf{8.838 \times 10^{-2}}$	9.073×10^{-2}
	Worst	1.003×10^{-1}	9.854×10^{-2}	$\mathbf{9.848 \times 10^{-2}}$
	Std. Dev.	2.519×10^{-3}	2.913×10^{-3}	$\mathbf{2.135 \times 10^{-2}}$
GDx	Average	7.868×10^{-2}	8.357×10^{-2}	$\mathbf{7.692 \times 10^{-2}}$
	Best	7.240×10^{-2}	7.698×10^{-2}	$\mathbf{7.227 \times 10^{-2}}$
	Worst	8.891×10^{-2}	9.456×10^{-2}	$\mathbf{8.118 \times 10^{-2}}$
	Std. Dev.	4.093×10^{-3}	4.468×10^{-3}	$\mathbf{2.245 \times 10^{-2}}$
Spacing	Average	8.465×10^{-2}	$\mathbf{7.959 \times 10^{-2}}$	8.248×10^{-2}
	Best	7.110×10^{-2}	6.991×10^{-2}	$\mathbf{6.080 \times 10^{-2}}$
	Worst	1.024×10^{-1}	$\mathbf{9.121 \times 10^{-2}}$	1.198×10^{-1}
	Std. Dev.	9.119×10^{-3}	$\mathbf{5.567 \times 10^{-3}}$	1.302×10^{-2}

6. Final Discussion

In this work, a multi-objective HS algorithm (MOHSg) is proposed, whose fundamental contribution consists of the pitch adjustment based on the crowding distancing by

genotype, that is, the crowding distancing that works in the search space. This algorithm is capable of automatically adjusting the exploration and exploitation by adjusting the pitch, based on the distribution of solutions in the search space during the formation of the Pareto front. Therefore, MOHSg only needs the presetting of the harmonic memory and pitch adjustment parameters for its operation, without requiring the static bandwidth parameter of the MOHS2 variant nor the dynamic bandwidth and pitch adjustment parameters needed by the MOHS3 algorithm.

For the test of the proposed algorithm, six multi-objective optimization problems were used with a diversity of characteristics regarding the shape of the Pareto-optimal fronts and Pareto-optimal sets, the coexistence of local and global solutions (multi-modality), and the number of decision variables and objective functions. MOHSg was able to produce similar or better results to those generated by the MOHS2 and MOHS3 algorithms, which constitute HS variants representative of the state-of-the-art in multi-objective optimization. Specifically, MOHSg produced the best results in three of the proposed problems, while in the rest, it registered competent results, excelling in some punctual performance indicators.

From this comparative study, it can be concluded that the harmonic search algorithm based on pitch adjustment by genotype is an effective tool for solving multi-objective optimization problems. As part of future work, the application of the proposed algorithm to the solution of multi-objective problems with functional restrictions is proposed.

Author Contributions: Conceptualization and methodology, D.M.-P. and E.A.P.-F.; validation and formal analysis, M.B.C.-Y. and G.S.-C.; investigation and software, D.M.-P.; writing—original draft preparation, D.M.-P.; writing—review and editing, D.M.-P., E.A.P.-F., E.A.P.-F. and E.V.-A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Acknowledgments: The authors wish to thank Instituto Politécnico Nacional of México for its support via Secretaría de Investigación y Posgrado with the SIP projects 20211567 and 20211583.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the development of the study or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

HM	Harmonic Memory
HS	Harmony Search
MOHS	Multi-Objective Harmony Search

Appendix A

Description of the test multi-objective problems.

Appendix A.1. P1

Minimize

$$\begin{cases} f_1 = |x_1 - 2| \\ f_2 = 1 - \sqrt{|x_1 - 2|} + 2(x_2 - \sin(6\pi|x_1 - 2| + \pi)^2 \end{cases}$$

where $1 \leq x_1 \leq 3$; $-1 \leq x_2 \leq 1$.

Appendix A.2. P2

Minimize

$$\begin{cases} f_1 = |x_1| \\ f_2 = \begin{cases} 1 - x_1^2 + 2(x_2 - \sin(\pi|x_1|))^2 & 0 \le x_2 \le 1 \\ 1 - x_1^2 + 2(x_2 - 1 - \sin(\pi|x_1|))^2 & 1 \le x_2 \le 2 \end{cases} \end{cases}$$

where $-1 \le x_1 \le 1; 0 \le x_2 \le 2$.

Appendix A.3. P3

Minimize

$$\begin{cases} f_1 = x_1 \\ f_2 = g(x_2) \cdot h(f_1, g) \end{cases}$$

$$g(x_2) = 2 - \exp\left[-2\log(2) \cdot \left(\frac{x_2 - 0.1}{0.8}\right)^2\right] \sin^6(2\pi x_2)$$

$$h(f_1, g) = 1 - \left(\frac{f_1}{g}\right)^2 - \frac{f_1}{g} \sin(2\pi q f_1)$$

where q is the number of discontinuities of the front, $q = 4$ for this problem; $0 \le x_1 \le 1$; $0 \le x_2 \le 1$.

Appendix A.4. P4

Minimize

$$\begin{cases} f_1 = x_1 \\ f_2 = g(x_2) \cdot h(f_1, g) \end{cases}$$

$$g(x_2) = 1 + 10x_2$$

$$h(f_1, g) = 1 - \left(\frac{f_1}{g}\right)^2 - \frac{f_1}{g} \sin(2\pi q f_1)$$

where q is the number of discontinuities of the front, $q = 10$ for this problem; $0 \le x_1 \le 1$; $0 \le x_2 \le 1$.

Appendix A.5. P5

Minimize

$$\begin{cases} f_1 = x_1 \\ f_2 = \frac{g(t)}{x_1} \end{cases}$$

$$g(t) = 2 - \exp\left[-2\log(2) \cdot \left(\frac{t - 0.1}{0.8}\right)^2\right] \sin^6(2\pi t)$$

$$t = x_2 + \sqrt{x_3}$$

where $0.1 \le x_1 \le 1.1; 0.1 \le x_2 \le 1.1; 0.1 \le x_3 \le 1.1$.

Appendix A.6. P6

Minimize

$$\begin{cases} f_1 = \cos\left(\frac{\pi}{2}x_1\right)\cos\left(\frac{\pi}{2}x_2\right)(1 + g(x)) \\ f_2 = \cos\left(\frac{\pi}{2}x_1\right)\sin\left(\frac{\pi}{2}x_2\right)(1 + g(x)) \\ f_3 = \sin\left(\frac{\pi}{2}x_1\right)(1 + g(x)) \end{cases}$$

$$g(x) = 2 - \sin^2\left(2\pi\left(x_3 - 0.5\sin(\pi x_2) + \frac{1}{4}\right)\right)$$

where $0 \le x_1 \le 1; 0 \le x_2 \le 1; 0 \le x_3 \le 1$.

References

1. Coello, C. A Short Tutorial on Evolutionary Multiobjective Optimization, Evolutionary Multi-Criterion Optimization. In Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization, Zurich, Switzerlan, 7–9 March 2001; Lecture Notes in Computer Science; Springer: Berlin, Germany, 2001; pp. 21–40. [CrossRef]
2. Geem, Z.; Kim, J.; Loganathan, G. A New Heuristic Optimization Algorithm. *Simulation* **2001**, *76*, 60–68. [CrossRef]
3. Ingram, G.; Zhang, T. Overview of Applications and Developments in the Harmony Search Algorithm. In *Music-Inspired Harmony Search Algorithm*; Geem, Z.W., Ed.; Studies in Computational Intelligence; Springer: Berlin/Heidelberg, Germany, 2009; Volume 191. [CrossRef]
4. Kim, J.H.; Geem, Z.W.; Jung, D.; Yoo, D.G.; Yadav, A. *Advances in Harmony Search, Soft Computing and Applications*; Springer: Cham, Switzerland, 2020.
5. Carbas, S.; Toktas, A.; Ustun, D. *Nature-Inspired Metaheuristic Algorithms for Engineering Optimization Applications*; Springer: Singapore, 2021.
6. Nigdeli, S.M.; Kim, J.H.; Bekdaş, G.; Yadav, A. *Proceedings of 6th International Conference on Harmony Search, Soft Computing and Applications*; Springer: Istanbul, Turkey, 2021.
7. Portilla-Flores, E.A.; Sánchez-Márquez, Á.; Flores-Pulido, L.; Vega-Alvarado, E.; Yañez, M.B.C.; Aponte-Rodríguez, J.A.; Niño-Suarez, P.A. Enhancing the Harmony Search Algorithm Performance on Constrained Numerical Optimization. *IEEE Access* **2017**, *5*, 25759–25780. [CrossRef]
8. Yang, X.S. Harmony Search as a Metaheuristic Algorithm. In *Music-Inspired Harmony Search Algorithm. Studies in Computational Intelligence*; Geem, Z.W., Ed.; Springer: Berlin/Heidelberg, Germany, 2009; Volume 191. [CrossRef]
9. Geem, Z.W.; Lee, K.S.; Park, Y. Application of Harmony Search to Vehicle Routing. *Am. J. Appl. Sci.* **2005**, *2*, 1552–1557. [CrossRef]
10. Geem, Z.W.; Hwangbo, H. Application of Harmony Search to Multi-Objective Optimization for Satellite Heat Pipe Design. In Proceedings of the US-Korea Conference on Science, Technology & Entrepreneurship, Teaneck, NJ, USA, 10–13 August 2006.
11. Geem, Z.W. Multiobjective Optimization of Time-Cost Trade-Off Using Harmony Search. *J. Constr. Eng. Manag. Asce* **2010**, *136*, 711–716. [CrossRef]
12. Gao, X.Z.; Wang, X.; Ovaska, S.J. Harmony Search Methods for Multi-modal and Constrained Optimization. In *Music-Inspired Harmony Search Algorithm. Studies in Computational Intelligence*; Geem, Z.W., Ed.; Springer: Berlin/Heidelberg, Germany, 2009; Volume 191. [CrossRef]
13. Xu, H.; Gao, X.Z.; Wang, T.; Xue, K. Harmony Search Optimization Algorithm: Application to a Reconfigurable Mobile Robot Prototype. In *Recent Advances In Harmony Search Algorithm. Studies in Computational Intelligence*; Geem, Z.W., Ed.; Springer: Berlin/Heidelberg, Germany, 2010; Volume 270. [CrossRef]
14. Ricart, J.; Hüttemann, G.; Lima, J.; Baran, B. Multiobjective Harmony Search Algorithm Proposals. *Electron. Notes Theor. Comput. Sci.* **2011**, *281*, 51–67. [CrossRef]
15. Sivasubramani, S.; Swarup, K.S. Multi-objective harmony search algorithm for optimal power flow problem. *Int. J. Electr. Power Energy Syst.* **2011**, *33*, 745–752. [CrossRef]
16. Mahdavi, M.; Fesanghary, M.; Damangir, E. An improved harmony search algorithm for solving optimization problems. *Appl. Math. Comput.* **2007**, *188*, 1567–1579. [CrossRef]
17. Nekooei, K.; Farsangi, M.M.; Nezamabadi-Pour, H.; Lee, K.Y. An Improved Multi-Objective Harmony Search for Optimal Placement of DGs in Distribution Systems. *IEEE Trans. Smart Grid* **2013**, *4*, 557–567. [CrossRef]
18. Al-Omoush, A.A.; Alsewari, A.A.; Alamri, H.S.; Zamli, K.Z. Comprehensive Review of the Development of the Harmony Search Algorithm and its Applications. *IEEE Access* **2019**, *7*, 14233–14245. [CrossRef]
19. Omran, M.; Mahdavi, M. Global-best harmony search. *Appl. Math. Comput.* **2008**, *198*, 643–656. [CrossRef]
20. Yadav, P.; Kumar, R.; Panda, S.; Chang, C. An Intelligent Tuned Harmony Search Algorithm for Optimisation. *Inf. Sci.* **2012**, *196*, 47–72. [CrossRef]
21. Zou, D.; Wu, J.; Li, S.; Li, Y. A novel global harmony search algorithm for reliability problems. *Comput. Ind. Eng.* **2010**, *58*, 307–316. [CrossRef]
22. Jayalakshmi, P.; Sridevi, S.; Janakiraman, S.A. Hybrid Artificial Bee Colony and Harmony Search Algorithm-Based Metahueristic Approach for Efficient Routing in WSNs. *Wirel. Pers. Commun.* **2021**, 1–17. [CrossRef]
23. Mahafzah, B.A.; Alshraideh, M. Hybrid harmony search algorithm for social network contact tracing of COVID-19. *Soft Comput.* **2021**, 1–23. [CrossRef]
24. Geem, Z.W. *Harmony Search Algorithms for Structural Design Optimization*; Springer: Berlin, Germany, 2009. [CrossRef]
25. Geem, Z.W. *Recent Advances in Harmony Search Algorithm*; Springer: Berlin, Germany, 2010. [CrossRef]
26. Tuo, S.; Yong, L.; Deng, F. A Novel Harmony Search Algorithm Based on Teaching-Learning Strategies for 0–1 Knapsack Problems. *Sci. World J.* **2014**, *2014*, 637412. [CrossRef]
27. Wang, C.; Huang, Y. Self-adaptive harmony search algorithm for optimization. *Expert Syst. Appl.* **2010**, *37*, 2826–2837. [CrossRef]
28. Gupta, S. Enhanced harmony search algorithm with non-linear control parameters for global optimization and engineering design problems. *Eng. Comput.* **2021**, 1–24. [CrossRef]
29. Deb, K. *Multiobjective Optimization Using Evolutionary Algorithms*; Wiley: New York, NY, USA, 2001.
30. Fonseca, C.M.; Fleming, P.J. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evol. Comput.* **1995**, *3*, 1–16. [CrossRef]

31. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
32. Srinivas, N.; Deb, K. Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evol. Comput.* **1994**, *2*, 221–248. [CrossRef]
33. Deb, K.; Goel, T. Controlled Elitist Non-dominated Sorting Genetic Algorithms for Better Convergence. In Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization, Zurich, Switzerland, 7–9 March 2001; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2001. [CrossRef]
34. Matlab, MathWorks: User's Guide (R2018a). 2018. Available online: https://la.mathworks.com/help/gads/genetic-algorithm-options.html (accessed on 6 August 2021).
35. Liang, J.; Qu, B.; Gong, D.; Yue, C. *Problem Definitions and Evaluation Criteria for the CEC 2019 Special Session on Multimodal Multiobjective Optimization*; Scribd Inc.: Wellington, New Zealand, 2019.
36. Deb, K. Multi-objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems. *Evol. Comput.* **1999**, *7*, 205–230. [CrossRef]
37. Coello, C.A.C.; Cortés, N.C. Solving Multiobjective Optimization Problems Using an Artificial Immune System. *Genet. Program. Evol. Mach.* **2005**, *6*, 163–190. [CrossRef]
38. Van Veldhuizen, D.A. Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. Ph.D. Dissertation, Air Force Institute of Technology, Wright-Patterson AFB, OH, USA, 1999.
39. Schott, J.R. Fault Tolerant Design Using Single and Multi-Criteria Genetic Algorithm Optimization. Mater's Thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge, MA, USA, 1995.
40. Van Veldhuizen, D.A.; Lamont, G. On measuring multiobjective evolutionary algorithm performance. In Proceedings of the 2000 Congress on Evolutionary Computation, La Jolla, CA, USA, 16–19 July 2000; Volume 1, pp. 204–211. [CrossRef]