Harmony Search for Multi-objective Optimization

Lucas M. Pavelski, Carolina P. Almeida and Richard A. Gonçalves

Computer Science Department - DECOMP
State University in the Midwest of Paraná - UNICENTRO
Guarapuava, PR, Brazil

lmpavelski@yahoo.com.br, carol@unicentro.br, richard@unicentro.br

Abstract—This paper investigates the efficiency of Harmony Search based algorithms for solving multi-objective problems. For this task, four variants of the Harmony Search algorithm were adapted in the Non-dominated Sorting Genetic Algorithm II (NSGA-II) framework. Harmony Search is a recent proposed music-inspired metaheuristic while NSGA-II is a very successful evolutionary multi-objective algorithm. The four proposed methods are tested against each other using a set of benchmark instances proposed in CEC 2009. The best proposed algorithm is also compared with NSGA-II. The preliminary results are very promising and stand the proposed approach as a candidate to the State-of-art for multi-objective optimization, encouraging further researches in the hybridization of the Harmony Search and Multi-objective Evolutionary Algorithms.

Keywords—Harmony Search; Multi-objective Optimization; NSGA-II;

I. INTRODUCTION

Many real-world problems can be formulated as a non-linear optimization having multiple objectives, the so called Multi-objective Optimization Problems (MOPs). However, the objectives are often conflicting, leading to the problem of finding a set of optimal solutions, called Pareto-optimal front. A good approximation of the Pareto-optimal front helps the Decision Maker in choosing the best compromise solution among all solutions [1][2].

The Evolutionary Algorithms (EAs) are nature-inspired stochastic search techniques based on natural selection and biological evolution mechanisms [3]. In the field of Multi-objective Optimization, the EAs have become a valuable tool, mostly because their population based approach, simple design and flexibility. The so called Multi-objective Optimization Evolutionary Algorithms (MOEAs) are now an established field of research and application [4][5]. The Non-dominated Sorting Genetic Algorithm (NSGA-II)[6] is a popular MOEA that uses concepts of Pareto-dominance and crowding distance to find approximations to the Pareto-optimal front.

Recently, a new heuristic algorithm for optimization was proposed by [7], the Harmony Search (HS). The HS algorithm mimics the improvisation of music players and has already been applied to many problems and some variants were proposed.

This paper investigates the performance of HS operators hybridized with the NSGA-II framework applied to MOPs. In Section II, some background on MOEAs and HS is presented. The proposed multi-objective HS algorithms are described in Section III. The algorithms were tested in ten unconstrained problems instances taken from CEC 2009 [8] and the results are presented in Section IV. Finally, in Section V some conclusions and future works are given.

II. BACKGROUND

A. Multi-objective Optimization

The Multi-objective Optimization (MO) generalizes Single-objective Optimization (SO) for a finite number of objectives ($M > 1$), often self-conflicting. Therefore, MOPs are more flexible for real-world applications and can be mathematically expressed in the following way [4]:

\[
\begin{align*}
\text{Min./Max.} & \quad f_m(x), \quad m = 1, \ldots, M; \\
\text{subject to} & \quad g_j(x) \geq 0, \quad j = 1, \ldots, J; \\
& \quad h_k(x) = 0, \quad k = 1, \ldots, K; \\
& \quad x_i^{(L)} \leq x_i \leq x_i^{(U)} \quad i = 1, \ldots, n.
\end{align*}
\]

where the problem constraints ($g_j$ and $h_k$), together with lower ($x_i^{(L)}$) and upper bounds ($x_i^{(U)}$), form the decision space ($\Omega$) and the images of the $M$ objective functions form the objective space ($\mathcal{Y}$) in $\mathbb{R}^M$.

For every solution $x \in \Omega$, there exists a corresponding point $y \in \mathcal{Y}$. When the objective space $\mathcal{Y}$ has many dimensions, witch is the case for MOPs, the task of comparing a number of solutions become complex. In SO, the solutions of a problem are compared using the objective function itself. Nevertheless, in MO the Pareto dominance concept is frequently used in the comparison of solutions [5].

Given the solutions $x^1$ and $x^2$, and the corresponding objective vectors $y^1$ and $y^2$ resulting from the MOP’s functions $f_i$, $i = 1, \ldots, M$. One say that $y^1$ dominates $y^2$ ($y^1 \prec y^2$) if no component $y_i^1$ of $y^1$ is worse than the corresponding component $y_i^2$ of $y^2$ and at least one $y_i^1$ is better, for $i = 1, \ldots, M$ and $j \in \{1, \ldots, M\}$. Thus, a partial order over the set of solutions is obtained [1].

As can be seen, there may be multiple optimal solutions to a multi-objective function. The set of all optimal solutions is called Pareto-optimal set and the set of corresponding images in the objective space is called Pareto-optimal Front.
A Pareto-set approximation is a finite set of mutually non-dominated solutions. The goal of the MO is to find or approximate the Pareto set, consequently, helping the Decision Maker choose the best compromise solution [5].

B. Multi-objective Evolutionary Algorithms

The Multi-objective Evolutionary Optimization, usually iteratively guides the population, or current Pareto-set approximation, towards the decision space to the Pareto-optimal set. However, the approximation may not contain all solutions of the Pareto-optimal set. Hence, the two main issues when designing a MOEA are [5][9]: accomplish fitness assignment and selection, respectively, in order to guide the search towards the Pareto-optimal set and maintain a diverse population in order to prevent premature convergence achieving a well distributed trade-off Pareto-optimal front approximation.

On most of the state-of-art MOEAs, the first issue is approached maintaining the non-dominated solutions towards the selection phase, in a procedure called non-dominated sorting. The diversity is maintained with many different ways, using clustering methods, user preferences, population operators and others [5][2]. The use of elitism in MOEAs was also regarded to be a good strategy [9].

C. Non-Dominated Sorting Genetic Algorithm II

One of the most popularly used MOEAs is the NSGA-II [4]. The NSGA was first proposed in [10], and later a computationally fast second version (NSGA-II) was proposed in [6]. The NSGA-II uses the genetic operators, Pareto domination relation and density estimation to select the same prior number of individuals from both populations.

The elitist sorting is made assigning ranks of non-dominated solutions to the elitists. First all non-dominated solution in the population are set with rank 1, then considering these were removed, the next non-dominated solutions receive rank 2 and so on. The lowest ranks are carried through the next generation. When the number of individuals of a certain rank exceeds the size of the population, these individuals are sorted by means of crowding distance (cd). The crowding distance is computed as the mean perimeter of the cuboid surrounding the solution.

The NSGA-II then uses the non-dominated sorting elitist selection to move solutions to the Pareto-optimal set, and the crowding distance to ensure diversity. The multi-objective versions of HS algorithm proposed on this paper were based on this NSGA-II framework. Also, NSGA-II was used as a comparative.

D. Harmony Search

The HS is a population metaheuristic that has been gained attention by many motives, like the easy implementation and parameters that directly controls the exploration by randomization, elitism degree and exploitation [11]. In addition, HS considers all the present solutions in the creation of a new one, rather than considering only a few as in the Genetic Algorithm [7].

The HS was inspired by the way that musicians improvise a new harmony. Just like musical performances seek a fantastic melody on determined aesthetic estimation by the set of sounds played on each practice, the optimization seeks the global optimum of a function by its components variables on each iteration [7]. A pseudocode for the HS is presented in Algorithm 1. It is composed of three phases: Harmony Memory (HM) initialization (line 2), improvisation of a new harmony (line 4) and memory update (line 7) [12]. Basically it initializes HMS harmonies at random; then, for NI generations, improvise a new harmony, or solution; if the new solution is better then the worst one in HM, the worst is replaced (memory update).

Algorithm 1 Harmony Search algorithm

1: function HARMONYSEARCH
2: \( HM = x_i \in \Omega, i \in \{1, \ldots, HMS\} \)
3: for \( t = 0, \ldots, NI \) do
4: \( x_{new} = \text{IMPROVISE}(HM) \)
5: \( x_{\text{worst}} = \min_{x_i} f(x_i), x_i \in HM \)
6: if \( f(x_{new}) > f(x_{\text{worst}}) \) then
7: \( HM = (HM \cup \{x_{new}\}) \setminus \{x_{\text{worst}}\} \)
8: end if
9: end for
10: end function

The improvisation scheme of the original HS is show by Algorithm 2, where \( r, r_1, r_2 \) and \( r_3 \) are uniform random variables. The parameters are: the Harmony Memory Consideration Rate (HMCR), Pitch Adjustment Rate (PAR) and the Bandwidth (BW). The improvisation can also be divided in three phases: (1) memory consideration (line 4), where an existing harmony component is copied from the memory; (2) pitch adjustment (line 6), which perturbates the chosen component and (3) random selection (line 9) that randomly generates a new component [12].

Algorithm 2 Harmony Search improvisation function

1: function IMPROVISE(HM) : \( x_{new} \)
2: for \( i = 0, \ldots, n \) do
3: if \( r_1 < \text{HMCR} \) then
4: \( x_{new}^{i} = x_{k}^{i}, k \in \{1, \ldots, HMS\} \)
5: if \( r_2 < \text{PAR} \) then
6: \( x_{new}^{i} = x_{new}^{i} \pm r_3 \times BW \)
7: end if
8: else
9: \( x_{new}^{i} = x_{L}^{i} + r \times (x_{U}^{i} - x_{L}^{i}) \)
10: end if
11: end for
12: end function
As can be seen, HS algorithm has a simple structure and easy implementation. Its customization is motivated to the interested researchers [12]. Hence, many algorithms for improvisations schemes and parameters adjustments were proposed, this paper covers: Improved Harmony Search (IHS) [13], Global-best Harmony Search (GHS) [14] and Self-adaptive Global-best Harmony Search (SIGHS) [15].

1) Improved Harmony Search: The IHS, proposed in [13], provides the fine-tuning of the parameters PAR and BW, making them change dynamically with generation number (t) and the total number of improvisations (NI), as shown in equations 1 and 2.

\[
PAR(t) = PAR_{\min} + \frac{(PAR_{\max} - PAR_{\min})}{NI} \times t \quad (1)
\]

\[
BW(t) = BW_{\max}e^{-\left(\frac{BW_{\min}}{BW_{\max}}\right)^{NI} \times t} \quad (2)
\]

2) Global-best Harmony Search: First proposed in [14] and inspired by the concepts of swarm intelligence and Particle Swarm Optimization, the GHS involves the best harmony in the improvisation of new ones. For this purpose, the original pitch adjustment phase is modified such that the new harmony can mimic the best harmony in the HM (\(x_{\text{best}}\)), as shown in equation 3. The BW parameter is not used and PAR is adjusted dynamically, as in IHS, by the equation 1.

\[
x_{i}^{\text{new}} = x_{k}^{\text{best}}, \quad k \in (1, 2, \ldots, n) \quad (3)
\]

3) Self-adaptive Global-best Harmony Search: Proposed in [15], the SGHS also involves the best harmony in the improvisation and in addition, provides self-adaptation to the PAR and HMCR parameters and modifies the improvisation scheme [15]. The modified memory consideration and pitch adjustment phases are shown in equations 4 and 5, and the BW parameter is changed dynamically as in equation 6.

\[
x_{i}^{\text{new}} = x_{i}^{k} \pm r \times BW, \quad k \in (1, \ldots, HMS), \quad r \in U(0, 1) \quad (4)
\]

\[
x_{i}^{\text{new}} = x_{i}^{\text{best}} \quad (5)
\]

\[
BW(t) = \left\{ \begin{array}{ll}
BW_{\max} - \frac{BW_{\min} - BW_{\min}}{NI} & \text{if } t < NI/2, \\
BW_{\min} & \text{otherwise}
\end{array} \right. \quad (6)
\]

The parameters HMCR and PAR are normal random values defined in the ranges [0.9, 1] and [0, 1], with \(\sigma\) of 0.01 and 0.05 and \(\mu\) of HMCRm and PARm, respectively. The auto-adaptation of HMCRm and PARm is performed at each \(lp\) (learning period), defined as a number of improvisations, obtaining the mean of HMCR and PAR values from the harmonies that were included in HM in the \(lp\) generations.

### III. PROPOSED MULTI-OBJECTIVE HARMONY SEARCH ALGORITHMS

This paper proposes a multi-objective version of the original HS algorithm using the NSGA-II framework, named Non-dominated Sorting Harmony Search (NSHS). It also proposes multi-objective versions for the IHS, GHS and SGHS algorithms named NSIHS, NSGHS, NSSGHS, respectively. In order to analyze the HS operators and its variants in a MOP, some changes in the original algorithms were done. Algorithm 3 shows the general scheme of NSHS. The two major changes carried by all HS variants were:

- For each generation, the amount of harmonies in the memory is doubled (lines 5 to 8), including all improvised ones, instead of the HS selection where the worst harmony is replaced. The motive that lead to this change is that it would be computationally expensive if, for each new harmony, a multi-objective selection were made in the entire memory;
- In the selection phase, the non-dominated sorting, the crowding distance and the elitism were applied, as in NSGA-II algorithm framework [6] (line 10).

**Algorithm 3 Non-dominated Sorting Harmony Search**

1: function NSHS
2: \[HM = x_{i} \in \Omega, \; i \in (1, \ldots, HMS)\]
3: for \(j = 0, \ldots, NI / HMS\) do
4: \[HM^{\text{new}} = \emptyset\]
5: for \(k = 0, \ldots, HMS\) do
6: \[x^{\text{new}} = \text{IMPROVISE}(HM)\]
7: \[HM^{\text{new}} = HM^{\text{new}} \cup x^{\text{new}}\]
8: end for
9: \[HM = HM \cup HM^{\text{new}}\]
10: end for
11: end function

The main change was in the selection phase and the improvisation of a new harmony vary with the different HS variants, e.g. Original HS, IHS, GHS and SGHS. For each variant, some minor changes were made, some relevant remarks are:

- **IHS:** Similar to the original version, just considering the generation (t) as the amount of harmonies improvised at the moment. The IHS was already successfully applied by [16] in the multi-objective Environmental/Economic Dispatch problem with a similar approach to the one analyzed in this paper.
- **GHS:** The choice for the best is made for each new harmony, choosing, with equal probability, a harmony from the actual non-dominated set.
- **SGHS:** The best harmony choice is performed as in GHS but the HMCRm and PARm parameters are adapted each generation \(j\), considering the mean of the
Table I

<table>
<thead>
<tr>
<th>HS and Variants Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMC R</td>
</tr>
<tr>
<td>NSHS</td>
</tr>
<tr>
<td>NSIHS</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>NSGHS</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>NSSGHS</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

**PAR and HMC R values of the different improvised harmonies, i.e. where (cd ≥ 0).**

IV. EXPERIMENTAL RESULTS

This section presents the results of the computational experiments performed with 10 instances from the CEC 2009 multi-objective benchmark [8]. The functions are general unconstrained (bound constrained) MOPs, with different characteristics, named UF1, UF2 to UF10. The MOPs contain 2 (UF1 to UF7) or 3 (UF8 to UF10) objective functions and search space dimensions (n) equal to 30.

Firstly, the proposed algorithms (NSHS, NSIHS, NSGHS and NSSGHS) were compared to determine the best proposed approach. Then the best proposed algorithm was compared with the NSGA-II algorithm.

The population size, or HMS in the case of HS and variants, was set to 200, and the number of evaluations of the objective functions was fixed to 150000. The parameters for the NSHS and variants are presented in Table I, where Δx is the range of values of search space (i.e. \( x(U) - x(L) \)).

The PAR values for NSHS, NSIHS and NSGHS were empirically determined while the HMC R value used was the one suggested in [14]. The NSGA-II uses polynomial mutation with probability of 1/n, binary tournament for mating selection and simulated binary crossover with probability of 0.7. Each algorithm was executed 30 independent times for each considered instance.

A. Quality Indicators and Statistical Tests

In this work, only non-parametric statistical tests were applied due to the non-normality of the various results.

The first step in assessing the relative quality of approximation sets generated by different multi-objective evolutionary algorithms was the dominance ranking, as suggested in [17]. The Mann-Whitney test was used to determine if there was statistical difference between the dominance ranking of a pair of algorithms. If no significant difference is inferred in the dominance ranking, as occurred in all the cases considered, then three quality indicators were applied: hypervolume, additive unary-ε and \( R_2 \) [18]. To calculate the dominance ranking and the quality indicators, the PISA framework [19] was used.

In order to present a summary of the results and to evaluate the overall performance of each algorithm, the Mack-Skillings [20] variation of the Friedman test (MS_Friedman) [21][22][23] was applied to the case of multiple observations (replications). The test is based on the ranks of the algorithms (lower ranks are associated with better algorithms). This is a conservative test [24] and can be used to detect differences in algorithms across multiple instances of a problem [25].

The results of the MS_Friedman test were presented through the mean rank value of each indicator for each algorithm in all problem instances. If the MS_Friedman test detected significant difference among at least a pair of algorithms (the p-value was low indicating that null hypothesis was rejected), a post hoc analysis was carried out. The post hoc analysis used corresponds to an asymptotically distribution-free multiple comparison procedure using within-blocks ranks that is designed to make two-sided decisions about individual differences between pairs of algorithms [20]. This procedure was based on a critical value for the difference between the mean ranks of a pair of algorithms.

Different from the MS_Friedman test that was considered for a macro evaluation, i.e., it was used to evaluate the overall performance, a Kruskal-Wallis test was also adopted for a micro evaluation. So the Kruskal-Wallis test was used to detect if there was statistical difference among the indicators found by each algorithm [26] for each instance.

The results of the Kruskal-Wallis test are represented by the p-values of the corresponding statistical test (Tables II, III and IV). P-values lower than 0.05 indicate that the first algorithm is better than the second algorithm (bold face) while p-values greater than 0.95 indicate that the second algorithm is better than the first algorithm (italic face), with 95% confidence.

B. Comparison among the Proposed Algorithms

The results obtained in Kruskal-Wallis for hypervolume, unary-ε and \( R_2 \) indicators are shown in Tables II, III and IV, respectively. The results can be summarized as:

- NSHS was among the best algorithms for solving UF3, UF5, UF6, UF7, UF9 and UF10.
- NSIHS, many times incomparable to NSHS, had a good performance on in UF3, UF4, UF5, UF6 and UF9.
- NSGHS obtained good results on the 3 objective problems, namely UF8, UF9 and UF10.
- NSSGHS performed well on UF1, UF4 and UF7.

The HS variants were then compared throughout MS_Friedman test. For all the indicators, the null hypothesis was rejected (i.e. p-values were near zero). The Table V show the mean values for the ranks obtained, together with the relative order given by the resultant critical difference of 7.29621 under 95% of confidence.
Table II  
**Kruskal-Wallis P-values for the Hypervolume Indicator**

<table>
<thead>
<tr>
<th></th>
<th>NSHS x NSHS</th>
<th>NSHS x NSGHS</th>
<th>NSHS x NSSGHS</th>
<th>NSHS x NSSGHS</th>
<th>NSGHS x NSHS</th>
<th>NSGHS x NSGHS</th>
<th>NSGHS x NSSGHS</th>
<th>NSGHS x NSSGHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>UF1</td>
<td>0.19</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>UF2</td>
<td>0.21</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>UF3</td>
<td>0.09</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>UF4</td>
<td>0.94</td>
<td>0.96</td>
<td>0.0</td>
<td>0.59</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>UF5</td>
<td>0.07</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>UF6</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>UF7</td>
<td>0.02</td>
<td>0.02</td>
<td>0.8</td>
<td>0.55</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>UF8</td>
<td>0.25</td>
<td>1.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>UF9</td>
<td>0.97</td>
<td>0.11</td>
<td>0.01</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>UF10</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.25</td>
<td>0.01</td>
<td>0.04</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table III  
**Kruskal-Wallis P-values for the Unary-ε Indicator**

<table>
<thead>
<tr>
<th></th>
<th>NSHS x NSHS</th>
<th>NSHS x NSGHS</th>
<th>NSHS x NSSGHS</th>
<th>NSHS x NSSGHS</th>
<th>NSGHS x NSHS</th>
<th>NSGHS x NSGHS</th>
<th>NSGHS x NSSGHS</th>
<th>NSGHS x NSSGHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>UF1</td>
<td>0.89</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>UF2</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>UF3</td>
<td>0.26</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>UF4</td>
<td>0.89</td>
<td>0.21</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>UF5</td>
<td>0.35</td>
<td>0.02</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>UF6</td>
<td>0.14</td>
<td>0.05</td>
<td>0.0</td>
<td>0.27</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>UF7</td>
<td>0.06</td>
<td>0.0</td>
<td>0.58</td>
<td>0.05</td>
<td>0.96</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>UF8</td>
<td>0.9</td>
<td>1.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>UF9</td>
<td>0.0</td>
<td>0.2</td>
<td>0.0</td>
<td>1.0</td>
<td>0.01</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>UF10</td>
<td>0.0</td>
<td>0.66</td>
<td>0.01</td>
<td>1.0</td>
<td>0.88</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table IV  
**Kruskal-Wallis P-values for the R2 Indicator**

<table>
<thead>
<tr>
<th></th>
<th>NSHS x NSHS</th>
<th>NSHS x NSGHS</th>
<th>NSHS x NSSGHS</th>
<th>NSHS x NSSGHS</th>
<th>NSGHS x NSHS</th>
<th>NSGHS x NSGHS</th>
<th>NSGHS x NSSGHS</th>
<th>NSGHS x NSSGHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>UF1</td>
<td>0.88</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>UF2</td>
<td>0.78</td>
<td>0.38</td>
<td>1.0</td>
<td>0.14</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>UF3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>UF4</td>
<td>0.92</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>UF5</td>
<td>0.31</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>UF6</td>
<td>0.05</td>
<td>0.02</td>
<td>0.0</td>
<td>0.32</td>
<td>0.02</td>
<td>0.06</td>
<td>0.06</td>
<td>0.06</td>
</tr>
<tr>
<td>UF7</td>
<td>0.04</td>
<td>0.01</td>
<td>0.89</td>
<td>0.33</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>UF8</td>
<td>0.2</td>
<td>1.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>UF9</td>
<td>0.21</td>
<td>0.01</td>
<td>0.0</td>
<td>0.06</td>
<td>0.0</td>
<td>0.04</td>
<td>0.04</td>
<td>0.04</td>
</tr>
<tr>
<td>UF10</td>
<td>0.0</td>
<td>0.33</td>
<td>0.0</td>
<td>0.99</td>
<td>0.04</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table V  
**MS_Friedman Mean Ranks for HS Variants**

<table>
<thead>
<tr>
<th></th>
<th>Hypervolume</th>
<th>Unary-ε</th>
<th>R2</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSHS</td>
<td>51.3 (1.5)</td>
<td>56.0 (1.5)</td>
<td>68.4 (3.5)</td>
</tr>
<tr>
<td>NSIHS</td>
<td>31.4 (1.5)</td>
<td>60.3 (2.5)</td>
<td>57.2 (2.3)</td>
</tr>
<tr>
<td>NSGHS</td>
<td>52.2 (1.5)</td>
<td>59.0 (2.5)</td>
<td>64.8 (3.0)</td>
</tr>
</tbody>
</table>

Based on the mean ranks obtained with MS_Friedman test, the best algorithm for all indicators was NSHS. The hypervolume indicator showed the NSIHS algorithm as the second best, followed by NSGHS and NSSGHS. On R2 indicator, the NSIHS, NSSGHS and NSGHS were incomparable. While on Unary-ε indicator NSSGHS was the worst algorithm.

C. Comparison against NSGA-II

Tests were also conducted to infer the relative performance of NSHS with respect to NSGA-II. The Mann-Whitney test was applied to the quality indicator results of both algorithms. It allowed to compare both algorithms statistically, for each problem instance. The p-values for the NSHS algorithm are presented in Table VI, again lower p-values means better performance of NSHS.

From Table VI can be observed that, NSHS algorithm beats NSGA-II on most of the instances in all indicators. In order to compare both algorithms in all of the problems, the MS_Friedman was executed. The p-values were again near zero and the critical difference was 2.795. The mean ranks for the hypervolume indicator was: 28.87 for NSHS and 32.13 for NSGA-II according to unary-ε indicator, 23.57 for NSHS and 37.43 for NSGA-II and for R2 indicator 27.83 for NSHS and 33.16 for NSGA-II. Therefore, NSHS obtained a better overall performance, accordingly to all quality indicators, in CEC 2009 unconstrained instances.

V. Conclusion

This work proposed the hybridization of four HS versions (HS, IHS, GHS and SGHS) with the NSGA-II framework. The four algorithms were then tested with ten instances of benchmark functions used in CEC 2009 [8]. Three quality indicators (Hypervolume, Unary-ε and R2) were used to evaluate the Pareto-front approximations outputted in each run. Statistical tests showed that NSHS, the original HS algorithm using non-dominated sorting, was the best among all proposed multi-objective versions. Our best approach, the NSHS algorithm, was then favorably compared with the original NSGA-II, attesting the success of the proposed hybridization.

The results obtained with HS operators in multi-objective problems motivate the analysis of different aspects, such as: effects of other HS variants and parameter values in problems with different characteristics, computational effort and comparisons against other MOEAs. Also, one next step to be considered would be the adaptation of HS operators on other state-of-art frameworks, like SPAM and HypE [27].
ACKNOWLEDGMENT

The authors would like to thank UNICENTRO and Fundação Araucária for the financial support.

REFERENCES


