Dynamic Regional Harmony Search with Opposition and Local Learning

A. K. Qin
INRIA Grenoble Rhone-Alpes
655 avenue de l'Europe, Montbonnot
38334 Saint Ismier Cedex, France
kai.qin@inrialpes.fr

Florence Forbes
INRIA Grenoble Rhone-Alpes
655 avenue de l'Europe, Montbonnot
38334 Saint Ismier Cedex, France
florence.forbes@inrialpes.fr

ABSTRACT
Harmony search (HS), as an emerging metaheuristic algorithm mimicking the musician's improvisation behavior, has demonstrated strong efficacy in solving various numerical and real-world optimization problems. To deal with the deficiencies in the original HS such as premature convergence and stagnation, a dynamic regional harmony search (DRHS) algorithm with opposition and local learning is proposed. DRHS utilizes opposition-based initialization, and performs independent harmony searches with respect to multiple groups created by periodically and randomly regrouping the harmony memory. Besides the traditional harmony improvisation operators, an opposition-based harmony creation scheme is used in DRHS to update each group memory. Any prematurely converged group will be restarted with its size being doubled to enhance exploration. Local search is periodically applied to exploit promising regions around top-ranked candidate solutions. The performance of DRHS has been evaluated and compared to the original HS using 12 numerical test problems taken from the CEC2005 benchmark. DRHS consistently outperforms HR on all test problems at both 10D and 30D.

Categories and Subject Descriptors
I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search – *heuristic methods*; G.1.6 [**Numerical Analysis**]: Optimization – *global optimization, unconstrained optimization*

General Terms
Algorithms, Performance, Experimentation

Keywords
Dynamical regrouping, local search, metaheuristic, opposition-based learning, regional harmony search, restart

1. INTROCUTION
Metaheuristic [1] is a generic computational technique that aims at efficiently solving various optimization problems arising in diverse scientific and engineering fields. Recent years have seen remarkable advances in metaheuristic algorithms inspired by different kinds of natural and behavioral phenomena, such as genetic algorithm [2], evolution strategy [3], artificial immune system [4], particle swarm optimization [5], ant colony optimization [6], and so on. These algorithms have demonstrated significant efficacy in numerous real-world applications.

Harmony search (HS) [7]-[15], as an emerging metaheuristic algorithm, mimics the musicians' improvisation behavior. In HS, a candidate solution of an optimization problem corresponds to a musical harmony composed of notes played by a group of musicians. Each decision variable in a candidate solution is analogous to a musician with its value range analogized by the pitch range within which the corresponding musician plays the note. The quality of candidate solutions corresponds to the euphoniousness of musical harmonies. By simulating how a group of musicians keep enriching their experiences to collaboratively seek for the most euphonious harmony in the improvisation procedure, HS searches for global optima using harmony improvisation operators to iteratively evolve the harmony memory (HM) that consists of promising candidate solutions.

HS has been successfully applied in a wide range of applications [8]-[10], although it suffers from some deficiencies. HS excessively relies on the harmony memory (HM) to exploit the solution space. The random selection operator can merely provide limited exploration beyond the HM. Therefore, the good performance of HS relies on a careful HM initialization that should extensively cover the solution space. On the other hand, a new harmony is always generated using the entire HM, which may degrade the efficacy of HS in solving multimodal problems. This is because too many harmonies scattering away from global optima may hamper the HM to evolve towards global optima. Moreover, the HM is prone to prematurely converging at undesirable local optima due to the greedy replacement based HM updating scheme. Furthermore, a limited HM capacity may result in stagnation during the searching unless the random selection operator takes considerable efforts to resume the evolution.

To address the above issues, we propose a dynamic regional harmony search (DRHS) algorithm incorporating opposition-based learning [16] and local search [17], [18]. Major characteristics of DRHS are highlighted below:

- Opposition-based learning is used to produce a HM that can better cover the entire solution space.
- The HM is randomly split into multiple groups. Each group performs HS independently. The HM is periodically regrouped. During the searching, any prematurely converged group will be restarted with its size being doubled. This dynamic regional search scheme can force each group to independently exploit different sub-regions of the solution space while attempting to prevent both stagnation and premature convergence.
- Each group first generates a new harmony using the original harmony improvisation operators. Then, an opposite harmony is created by applying the opposition-based learning to that new harmony with respect to the corresponding group. Among these two newly generated harmonies, the one with better

quality is used to update the memory of the corresponding group. This opposition-based harmony creation as well as the group based memory updating can enhance exploration within the group while attempting to prevent both stagnation and premature convergence.

- Local search is periodically applied on some top-ranked group-best harmonies to exploit promising regions around them.

The superiority of DRHS over HS is demonstrated using 12 numerical test problems (five unimodal and seven multimodal problems with shifted global optima and/or rotated searching landscapes) taken from the CEC2005 benchmark [19] at 10D and 30D.

The remaining paper is organized as follows. Section 2 reviews the original HS algorithm. DRHS is detailed in Section 3 followed by experiments in Section 4. Section 5 concludes the paper with some future work.

2. HARMONY SEARCH (HS)

HS has received considerable attention since its invention [7], and already developed into an independent research branch of metaheuristic. It evolves a HM as shown in (1), composed of *HMS* (i.e., harmony memory size) candidate solutions with *D* decision variables $\mathbf{x}_i = [x_i(1), \ldots, x_i(D)]$, $i \in \{1, \ldots, HMS\}$, towards global optima using three harmony improvisation operators, i.e., $\mathbf{O_1}$: HM consideration operator, $\mathbf{O_2}$: random selection operator and $\mathbf{O_3}$: pitch adjustment operator, as well as the greedy replacement based HM updating scheme. The objective function $f(\cdot)$ in (1) measures the solution quality. This work only considers single-objective optimization problems where $f(\cdot)$ is a scalar function indicating better quality when its value is smaller (larger) in the case of minimization (maximization).

$$HM = \begin{bmatrix} x_1(1) & x_1(2) & \cdots & x_1(D) & | & f(\mathbf{x}_1) \\ x_2(1) & x_2(2) & \cdots & x_2(D) & | & f(\mathbf{x}_2) \\ \vdots & \vdots & \cdots & \vdots & | & \vdots \\ x_{HMS}(1) & x_{HMS}(2) & \cdots & x_{HMS}(D) & | & f(\mathbf{x}_{HMS}) \end{bmatrix} \quad (1)$$

Firstly, a HM of fixed size *HMS* is randomly initialized within the solution space. Then, a new harmony $\mathbf{x}_{\text{new}}$ is created by applying either $\mathbf{O_1}$ or $\mathbf{O_2}$ with probabilities *HMCR* and 1-*HMCR* respectively to determine the value of each decision variable $x_{\text{new}}(d)$, $d \in \{1, \ldots, D\}$, and subsequently applying $\mathbf{O_3}$ with probability *PAR* to refine the values of those decision variables produced by $\mathbf{O_1}$. *HMCR* and *PAR* denotes the HM consideration rate and the pitch adjustment rate (i.e., the operator execution probability), respectively. The parameter *BW* associated with $\mathbf{O_3}$ represents the bandwidth, which determines the maximum value range for the refining (i.e., the mutation step size). The newly generated harmony will replace the worst harmony in the current HM if it has better quality in comparison. This harmony creation and replacement process is repeated until certain termination criterion is met (e.g., the maximum number of function evaluations *maxFEvals* is reached).

The following describes the pseudo-code of the original HS algorithm for solving minimization problems where $U(0,1)$ denotes a random number uniformly distributed between 0 and 1. $x_{\text{U}}(d)$ and $x_{\text{L}}(d)$ represent the upper and lower bounds of the solution space with respect to the d^{th} decision variable.

Step 1 Set HS parameters: *HMS*, *HMCR*, *PAR* and *BW*

Step 2 Initialize HM randomly

For (i = 1 to *HMS*)

  For (d = 1 to D)

    $x_i(d) = x_{\text{L}}(d) + U(0,1) \times (x_{\text{U}}(d) - x_{\text{L}}(d))$

  End

  Evaluate $f(\mathbf{x}_i)$

End

Step 3 Create a new harmony $\mathbf{x}_{\text{new}} = [x_{\text{new}}(1), \ldots, x_{\text{new}}(D)]$ using three harmony improvisation operators $\mathbf{O_1}$, $\mathbf{O_2}$ and $\mathbf{O_3}$

For (d = 1 to D)

  If (U(0, 1) <= *HMCR*)

    $x_{\text{new}}(d) = x_r(d)$, r is random from {1,…,*HMS*} // $\mathbf{O_1}$

    If (U(0, 1) <= *PAR*)

      If (U(0, 1) <= 0.5) // $\mathbf{O_3}$

        $x_{\text{new}}(d) = x_{\text{new}}(d) + U(0,1) \times BW$

      Else

        $x_{\text{new}}(d) = x_{\text{new}}(d) - U(0,1) \times BW$

      End

End

  Else

    $x_{\text{new}}(d) = x_{\text{L}}(d) + U(0,1) \times (x_{\text{U}}(d) - x_{\text{L}}(d))$ // $\mathbf{O_2}$

  End

End

Evaluate $f(\mathbf{x}_{\text{new}})$

Step 4 Update the HM with $\mathbf{x}_{\text{new}}$ using the greedy replacement

$$\text{worst} = \arg\max_i (f(\mathbf{x}_{\text{i}}))$$

$$\mathbf{x}_{\text{worst}} = \mathbf{x}_{\text{new}} \text{, if } f(\mathbf{x}_{\text{new}}) < f(\mathbf{x}_{\text{worst}})$$

Step 5 If any termination criterion is met, return the best harmony found so far, otherwise go to **Step 3**.

3. DYNAMIC REGIONAL HARMONY SEARCH WITH OPPOSITION AND LOCAL LEARNING (DRHS)

Many HS variants have been developed in recent years. For example, the improved HS [12] dynamically adjusts the values of

PAR and *BW* at different searching stages according to certain rules. The global HS [13] creates new harmonies using the global-best harmony in the HM. The self-adaptive HS [14] utilizes the harmony distribution information to perform the pitch adjustment and thus eliminates the parameter *BW*. HS has also been widely hybridized with other metaheuristic algorithms such as particle swarm optimization [11] and differential evolution [9] to collaboratively boost the optimization performance.

This section describes a dynamic regional harmony search (DRHS) algorithm, which is proposed to address the deficiencies in the original HS as mentioned in Section 1.

Those deficiencies are recapitulated below, following by the strategies used in DRHS to address them:

- The good performance of HS relies on a careful HM initialization that should extensively cover the solution space.

DRHS strategies: DRHS initializes one half of the HM randomly within the solution space with another half obtained using opposition-based learning [16] with respect to the solution space. The opposition-based initialization scheme has been successfully incorporated into various metaheuristic algorithms [15], [16], which can make the initial candidate solutions to better cover the entire solution space.

- The new harmony is always generated using the entire HM, which may degrade the efficacy of HS in solving multimodal problems where many harmonies may scatter away from global optima.

DRHS strategies: DRHS splits the HM into multiple groups and forces each group independently exploit different sub-regions of the solution space. This can make promising sub-regions of the solution space to be efficiently exploited by certain groups. To prevent premature convergence, the HM is periodically and randomly regrouped. Moreover, an opposition-based restarting is invoked to reactive any converged group. Meanwhile, the size of any restarted group is doubled to enhance its exploration ability.

- The HM is prone to prematurely converging at undesirable local optima due to the greedy replacement based HM updating scheme.

DRHS strategies: For each group, besides a new harmony generated by the original harmony improvisation operators, DRHS also creates an opposite harmony by applying the opposition-based learning to that new harmony with respect to the corresponding group. Among these two newly generated harmonies, the one with better quality is used to update the group memory. This opposition-based harmony creation as well as the group based memory updating can reduce the risk of premature convergence.

- The limited HM capacity may lead to stagnation unless the random selection operator takes considerable efforts to resume the evolution

DRHS strategies: The above periodical HM regrouping, group restarting with doubled size and opposition-based harmony creation schemes can reduce the risk of stagnation.

Moreover, DRHS periodically applies local search (the period is set to 50 generations in our work) on some top-ranked group-best harmonies respectively to exploit promising regions around them. The Broyden–Fletcher–Goldfarb–Shanno (BFGS) [18] method is used with computation budget set to 200 function evaluations. To prevent premature convergence, the Baldwinian local learning rule [17] is used to only update the best solution found so far while not modifying those group-best harmonies from which local search starts.

Furthermore, DRHS reserves the final few numbers of function evaluations ($0.02 \times maxFEvals$ is used in our work) for BFGS to fully exploit the region around the best harmony in the HM. If BFGS prematurely converges, the global-best HS method [13] will be involved subsequently until the *maxFEvals* is reached. To better exploit around the best solution found so far, the Lamarckian local learning rule [17] is used to update the best harmony in the HM.

The following describes the pseudo-code of the DRHS algorithm for solving minimization problems where $U(0,1)$ denotes a random number uniformly distributed between 0 and 1. $x_{\mathrm{U}}(d)$ and $x_{\mathrm{L}}(d)$ represent the upper and lower bounds of the solution space with respect to the d^{th} decision variable. Parameters different from the original HS include: the number of groups (*#GP*), the initial group sizes (GPS_j, $j = 1,\ldots,\#GP$) and the regrouping period (*refreshGap*). The *HMS* equals the summation of all group sizes.

Step 1 Set DRHS parameters: *#GP*, GPS_j ($j = 1,\ldots,\#GP$), *HMCR*, *PAR*, *BW* and *refreshGap*

$$HMS = \sum_j GPS_j$$

Step 2 Initialize HM using opposition-based learning (assuming that *HMS* is an even number):

For (i = 1 to $HMS/2$)

For (d = 1 to D)

$$x_i(d) = x_{\mathrm{L}}(d) + U(0,1) \times (x_{\mathrm{U}}(d) - x_{\mathrm{L}}(d))$$

$$x_{i+HMS/2}(d) = x_{\mathrm{U}}(d) + x_{\mathrm{L}}(d) - x_i(d)$$

End

Evaluate $f(\mathbf{x}_i)$ and $f(\mathbf{x}_{i+HMS/2})$

End

Step 3 Apply DRHS

Step 3.1 Set the generation counter: *iGen* = 0, and randomly split the HM into *#GP* groups:

$$\mathrm{GP}_j = \{\mathbf{x}_k^j \mid k = 1,\cdots,GPS_j\},\ j \in \{1,\ldots,\#GP\}$$

Step 3.2 Apply HS to each group with restarting if converged

For (j = 1 to *#GP*)

Step 3.2.1 Restart the converged group with doubled size

$$\mathrm{var}(\mathrm{GP}_j) = \max_d(\mathrm{var}_k(x_k^j(d)))$$

If $\mathrm{var}(\mathrm{GP}_j) < 10^{-12}$

For (k = 1 to GPS_j)

For (d = 1 to D)

$$x_k^j(d) = x_{\mathrm{L}}(d) + U(0,1) \times (x_{\mathrm{U}}(d) - x_{\mathrm{L}}(d))$$

$x^j_{k+GPS_j}(d) = x_U(d) + x_L(d) - x^j_k(d)$

End

Evaluate $f(\mathbf{x}^j_k)$ and $f(\mathbf{x}^j_{k+GPS_j})$

End

$GPS_j = 2 \times GPS_j$

$HMS = \sum_j GPS_j$

End

Step 3.2.2 Create a new harmony $\mathbf{x}^j_{\text{new}}$ in GP_j

For (d = 1 to D)

If (U(0,1) <= $HMCR$)

$x^j_{\text{new}}(d) = x^j_r(d)$, r is random from $\{1, \ldots, GPS_j\}$ // **O_1**

If (U(0,1) <= PAR)

If (U(0, 1) <= 0.5) // **O_3**

$x^j_{\text{new}}(d) = x^j_{\text{new}}(d) + U(0,1) \times BW$

Else

$x^j_{\text{new}}(d) = x^j_{\text{new}}(d) - U(0,1) \times BW$

End

End

Else

$x^j_{\text{new}}(d) = x_L(d) + \text{U}(0,1) \times (x_U(d) - x_L(d))$ // **O_2**

End

End

Evaluate $f(\mathbf{x}^j_{\text{new}})$

Step 3.2.3 Create an opposite harmony $\mathbf{x}^j_{\text{opt}}$ of $\mathbf{x}^j_{\text{new}}$ in GP_j

For (d = 1 to D)

$x^j_{\text{opt}}(d) = \max_k(x^j_k(d)) + \min_k(x^j_k(d)) - x^j_{\text{new}}(d)$

End

Evaluate $f(\mathbf{x}^j_{\text{opt}})$

Step 3.2.4 Update GP_j using the better one of $\mathbf{x}^j_{\text{opt}}$ and $\mathbf{x}^j_{\text{new}}$

$\text{worst}^j = \arg\max_k(f(\mathbf{x}^j_k))$ // find group-worst harmony

If $f(\mathbf{x}^j_{\text{opt}}) < f(\mathbf{x}^j_{\text{new}})$

$\mathbf{x}^j_{\text{worst}^j} = \mathbf{x}^j_{\text{opt}}$ if $f(\mathbf{x}^j_{\text{opt}}) < f(\mathbf{x}^j_{\text{worst}^j})$

Else

$\mathbf{x}^j_{\text{worst}^j} = \mathbf{x}^j_{\text{new}}$ if $f(\mathbf{x}^j_{\text{new}}) < f(\mathbf{x}^j_{\text{worst}^j})$

End

End

Step 3.3 Increase *iGen* by 1

Step 3.4 Apply BFGS to a few top-ranked group-best harmonies

If (mod(*iGen*, 50) == 0)

Rank the best harmonies within each group

Apply BFGS respectively to the first 25% top-ranked group-best harmonies for at most 200 function evaluations

End

Step 3.5 Randomly regroup HM under a fixed period

If (mod(*iGen*, *refreshGap*) == 0)

$\text{HM} = \bigcup_j \text{GP}_j$

Randomly split the HM into #GP groups

End

Step 3.6 Go to **Step 3.2** if the expensed total number of function evaluations is smaller than $0.98 \times maxFEvals$

Step 4 Apply BFGS to the global-best harmony in HM

$\text{best} = \arg\min_i(f(\mathbf{x}_i))$ // Find global-best harmony

Apply BFGS to the global-best harmony $\mathbf{x}_{\text{best}}$ in the current HM for at most $0.02 \times maxFEvals$ function evaluations

If BFGS improves $f(\mathbf{x}_{\text{best}})$, replace $\mathbf{x}_{\text{best}}$ by the BFGS solution

Step 5 Apply the global-best HS to the current HM

If the expensed total number of function evaluations is smaller than *maxFEvals*, apply the global-best HS to the current HM until any termination criterion is met.

Note that whenever any termination criterion is met during the searching, DRHS immediately terminates and returns the best solution found so far.

4. EXPERIMENTS

The performances of DRHS and HS are evaluated and compared using 12 numerical unimodal and multimodal test problems with shifted global optima and/or rotated searching landscapes at 10D and 30D.

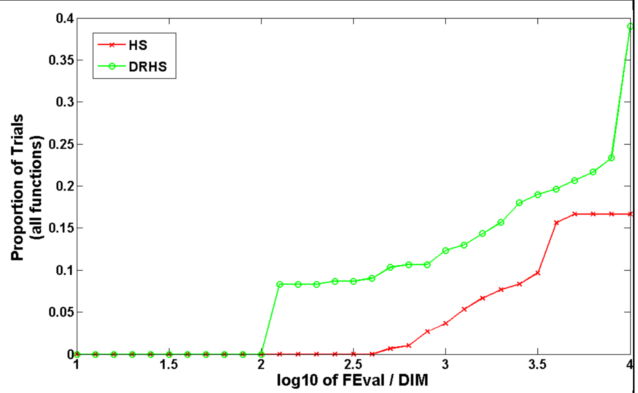

Figure 1. Empirical cumulative distribution function (ECDF) of the number of function evaluations (FEval) at success under the pre-specifeid accuracy level over 25 runs on all 12 test functions at 10D.

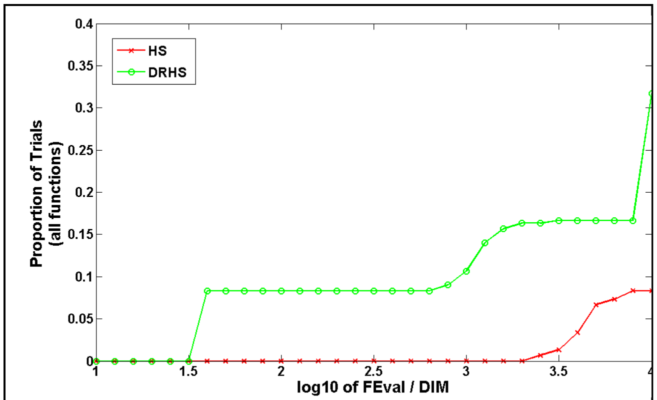

Figure 2. Empirical cumulative distribution function (ECDF) of the number of function evaluations (FEval) at success under the pre-specifeid accuracy level over 25 runs on all 12 test functions at 30D.

4.1 Test Problems

The following 12 numerical test functions taken from the CEC2005 benchmark [19] are used in our work:

- **Five unimodal functions**

F_1: Shifted Sphere Function

F_2: Shifted Schwefel's Problem 1.2

F_3: Shifted Rotated High Conditioned Elliptic Function

F_4: Shifted Schwefel's Problem 1.2 with Noise in Fitness

F_5: Schwefel's Problem 2.6 with Global Optimum on Bounds

- **Seven multimodal functions**

F_6: Shifted Rosenbrock's Function

F_7: Shifted Rotated Ackley's Function with Global Optimal on Bounds

F_8: Shifted Rastrigin's Function

F_9: Shifted Rotated Rastrigin's Function

F_{10}: Schwefel's Problem 2.13

F_{11}: Expended Extended Griewank's plus Rosenbrock's Function (F8F2)

F_{12}: Shifted Rotated Expended Scaffer's F6

The function definition, global optima and their corresponding objective function values, solution space ranges of these 12 functions are all detailed in [19].

4.2 Experimental Setup

HS is configured according to empirical guidelines [7], [8], [14]: *HMS* = 50, *HMCR* = 0.98, *PAR* = 0.3, *BW* = 0.01. The parameters of DRHS are set as: *HMCR* = 0.98, *PAR* = 0.3, *BW* = 0.01, *#GP* = 10, $GPS_i = 5$ ($i = 1,\ldots,10$), *refreshGap* = 10. Common parameters of HS and DRHS are set same for a fair comparison.

For each test problem, each of DRHS and HS is executed 25 times starting from different random seeds while both DRHS and HS share the same random seed with respect to any individual run.

Two termination criteria are applied: (1) the maximum number of function evaluations (*maxFEvals*) is reached. Here, the *maxFEvals* is set to 10^4 times the problem dimension, which means 10^5 for 10D problems and 3×10^5 for 30D problems; (2) The difference of objective function values between the best solution found so far and the global optimal solution (i.e., error function value (**EFV**)) is smaller than 10^{-8}. In such a case, the EFV is negligible and set to zero.

The optimization performance is quantitatively measured by (1) the mean value and standard deviation of the best EFVs achieved when an algorithm terminates over 25 runs and (2) the success rate (**SR**) over 25 runs. An optimization algorithm is regarded as successfully solving the problem once it achieves an EFV smaller than the pre-specified accuracy level. According to the specification in [19], the accuracy level is set to 10^{-6} for F_1 to F_5 and 10^{-2} for F_6 to F_{12}.

Practical optimization tasks are often subjected to the strict requirement on the computation speed of the algorithm applied to solve them, which is proportional to the executed number of function evaluations. To inspect an optimization algorithm's efficacy with respect to various computation budgets (i.e., the maximally allowed number of function evaluations), the empirical cumulative distribution function (ECDF) [20] with respect to the number of function evaluations at success (i.e., the number of function evaluations when the EFV just goes below the pre-specified accuracy level) over 25 runs on all 12 test functions is illustrated.

4.3 Results

Tables 1 and 2 report, with respect to each of 12 test problems at 10D and 30D respectively, the performances of HS and DRHS in terms of the mean value and standard deviation of the best EFVs over 25 runs as well as the SR under the pre-specified accuracy level over 25 runs. For each function, bold fonts show the largest SR (if not zero) and the optimal best EFVs (i.e., with the smallest mean value) as well as those best EFVs indiscernible from the optimal based on the Wilcoxon's signed-rank test [21] at the significance level of 0.05. This nonparametric statistical hypothesis test assesses whether the medians of two sets of the best EFVs achieved by two algorithms over 25 runs are statistically significantly different. In comparison with HS, DRHS consistently demonstrates superiority on all test problems at both 10D and 30D.

Table 1. Performances of HS and DRHS in terms of the mean value (mean) and standard deviation (std) of the best error function values achieved when the algorithm terminates as well as the success rate under the pre-specified accuracy level over 25 runs with respect to each of 12 test functions at 10D. Bold fonts show the optimal value as well as those indiscernible from the optimal based upon Wilcoxon's signed-rank test at the significance level of 0.05.

Function ID	Performance Measures		HS	DRHS
F_1	Best EFV	mean	3.039E-09	**0.000E+00**
		std	5.737E-09	**0.000E+00**
	SR (10^{-6})		**1.00**	**1.00**
F_2	Best EFV	mean	1.701E+02	**7.419E-09**
		std	1.092E+02	**1.577E-08**
	SR (10^{-6})		0.00	**0.92**
F_3	Best EFV	mean	1.008E+06	**4.329E+00**
		std	7.527E+05	**1.178E+01**
	SR (10^{-6})		0.00	0.00
F_4	Best EFV	mean	9.370E+02	**9.453E+01**
		std	7.260E+02	**9.328E+01**
	SR (10^{-6})		0.00	0.00
F_5	Best EFV	mean	1.804E+03	**2.688E+02**
		std	1.185E+03	**2.450E+02**
	SR (10^{-6})		0.00	0.00
F_6	Best EFV	mean	1.384E+03	**4.784E-01**
		std	2.976E+03	**1.322E+00**
	SR (10^{-2})		0.00	**0.88**
F_7	Best EFV	mean	2.036E+01	**2.000E+01**
		std	6.917E-02	**8.164E-05**
	SR (10^{-2})		0.00	0.00
F_8	Best EFV	mean	8.933E-07	**0.000E+00**
		std	5.492E-07	**0.000E+00**
	SR (10^{-2})		**1.00**	**1.00**
F_9	Best EFV	mean	1.230E+01	**3.423E+00**
		std	6.492E+00	**9.561E-01**
	SR (10^{-2})		0.00	0.00
F_{10}	Best EFV	mean	1.774E+02	**3.145E+00**
		std	4.350E+02	**6.774E+00**
	SR (10^{-2})		0.00	**0.04**
F_{11}	Best EFV	mean	4.319E-01	**4.082E-01**
		std	1.330E-01	**1.338E-01**
	SR (10^{-2})		0.00	0.00
F_{12}	Best EFV	mean	2.941E+00	**2.412E+00**
		std	4.499E-01	**5.805E-01**
	SR (10^{-2})		0.00	0.00

Figures 1 and 2 illustrate, at 10D and 30D respectively, the ECDFs with respect to the number of function evaluations at success under the pre-specified accuracy level over 25 runs on all 12 test functions. They clearly reveal that DRHS always outperforms HS at both 10D and 30D after the first 1000 function evaluations.

Table 2. Performances of HS and DRHS in terms of the mean value (mean) and standard deviation (std) of the best error function values achieved when the algorithm terminates as well as the success rate under the pre-specified accuracy level over 25 runs with respect to each of 12 test functions at 30D. Bold fonts show the optimal value as well as those indiscernible from the optimal based upon Wilcoxon's signed-rank test at the significance level of 0.05.

Function ID	Performance Measures		HS	DRHS
F_1	Best EFV	mean	2.997E-05	**0.000E+00**
		std	4.643E-06	**0.000E+00**
	SR (10^{-6})		0.00	**1.00**
F_2	Best EFV	mean	1.409E+03	**5.626E-08**
		std	6.133E+02	**1.201E-07**
	SR (10^{-6})		0.00	**0.16**
F_3	Best EFV	mean	7.513E+06	**2.014E+03**
		std	3.227E+06	**1.708E+03**
	SR (10^{-6})		0.00	0.00
F_4	Best EFV	mean	9.981E+03	**1.277E+03**
		std	2.996E+03	**7.172E+02**
	SR (10^{-6})		0.00	0.00
F_5	Best EFV	mean	5.766E+03	**3.364E+03**
		std	9.979E+02	**6.620E+02**
	SR (10^{-6})		0.00	0.00
F_6	Best EFV	mean	6.180E+02	**4.377E+01**
		std	2.476E+03	**6.848E+01**
	SR (10^{-2})		0.00	0.00
F_7	Best EFV	mean	2.094E+01	**2.000E+01**
		std	5.513E-02	**1.130E-06**
	SR (10^{-2})		0.00	0.00
F_8	Best EFV	mean	5.581E-03	**0.000E+00**
		std	1.267E-03	**0.000E+00**
	SR (10^{-2})		**1.00**	**1.00**
F_9	Best EFV	mean	5.125E+01	**2.436E+01**
		std	3.932E+01	**5.131E+00**
	SR (10^{-2})		0.00	0.00
F_{10}	Best EFV	mean	3.626E+03	**1.208E+03**
		std	3.419E+03	**2.065E+03**
	SR (10^{-2})		0.00	0.00
F_{11}	Best EFV	mean	1.962E+00	**1.516E+00**
		std	2.796E-01	**2.929E-01**
	SR (10^{-2})		0.00	0.00
F_{12}	Best EFV	mean	1.300E+01	**1.255E+01**
		std	3.029E-01	**2.861E-01**
	SR (10^{-2})		0.00	0.00

5. CONCLUSIONS AND FUTURE WORK

We present a dynamic regional harmony search (DRHS) algorithm with opposition and local learning to address the deficiencies in the original HS such as premature convergence and stagnation. After an opposition-based initialization, DRHS periodically and randomly regroups the HM, and performs the harmony search independently within each group using the original harmony improvisation operators as well as an opposition-based harmony creation scheme. Any prematurely converged group is restarted with its size being doubled to enhance exploration. Local search is periodically applied to exploit promising regions around some top-ranked harmonies.

Ongoing and planned research work includes: comprehensive analysis of parameter sensitivity, study of self-adaptive parameter turning schemes, further investigation of the local searching behavior, and extensive performance evaluation on more numerical and real-world problems.

6. ACKNOWLEDGMENTS

Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, being developed under the INRIA ALADDIN development action with support from CNRS, RENATER and several Universities as well as other funding bodies (see https://www.grid5000.fr). This work was partially supported by NSFC under Grant No. 61005051 and SRFDP under Grant No. 20100092120027.

7. REFERENCES

[1] Glover, F. W., and Kochenberger, G. A. *Handbook of Metaheuristics, 1st edition*. Springer, 2003.

[2] Goldberg, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Kluwer Academic Publishers, 1989.

[3] Hansen, N., and Ostermeier, A. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9, 2, 2001, 159-195.

[4] De Castro, L. N., and Timmis, J. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer, 2002.

[5] Kennedy, J., and Eberhart, R. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, 1995, 1942–1948.

[6] Dorigo, M., and Stützle, T. *Ant Colony Optimization*. MIT Press, Cambridge, MA, 2004.

[7] Geem, Z. W., Kim, J. H., and Loganathan, G. V. A new heuristic optimization algorithm: harmony search. *Simulation*, 76, 2, 2001, 60-68.

[8] Geem, Z. W. *Music-Inspired Harmony Search Algorithm: Theory and Applications, 1st edition*. Springer, 2009.

[9] Liao, T. W. Two hybrid differential evolution algorithms for engineering design optimization. *Applied Soft Computing*, 10, 4, 2010, 1188-1199.

[10] Geem, Z. W., and Williams, J. C. Ecological optimization using harmony search. In *Proceedings of American Conference on Applied Mathematics*, Harvard University, Cambridge, MA, USA, March 24-26, 2008.

[11] Zhao, S. Z., Suganthan, P. N., Pan, Q. K., and Tasgetiren, M. F. Dynamic multi-swarm particle swarm optimizer with harmony search. *Expert Systems with Application*, 38, 4, 2011, 3735-3742.

[12] Mahdavi, M., Fesanghary, M., and Damangir, E. An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and Computation*, 188, 2, 2007, 1567-1579.

[13] Omran, M. G. H., and Mahdavi, M. Global-best harmony search. *Applied Mathematics and Computation*, 198, 2, 2008, 643-656.

[14] Wang, C.-M., and Huang, Y.-F. Self-adaptive harmony search algorithm for optimization. *Expert Systems with Applications*, 37, 4, 2010, 2826-2837.

[15] Zhao, P. J. A hybrid harmony search algorithm for numerical optimization. In *Proceedings of 2010 International Conference on Computational Aspects of Social Networks (CASoN'10)*, 2010, 255-258.

[16] Rahnamayan, S., Tizhoosh, H. R., and Salama, M. M. A. Opposition-based differential evolution. *IEEE Transactions on Evolutionary Computation*, 12, 1, 2008, 64-79.

[17] Ku K. W. C., and Mak, M. W. Exploring the effects of Lamarckian and Baldwinian learning in evolving recurrent neural networks. In *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (CEC'07)*, 1997, 617-621.

[18] Avriel, M. *Nonlinear Programming: Analysis and Methods*. Dover Publishing, 2003.

[19] Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y.-P., Auger A., and Tiwari, S. *Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization*. Technical Report, Nanyang Technological University, Singapore, May 2005 AND KanGAL Report #2005005, IIT Kanpur, India.

[20] Hansen, N., Auger, A., Finck, S., and Ros, R. *Real-parameter black-box optimization benchmarking 2010: Experimental setup*. Technical Report RR-7215, INRIA, 2010.

[21] Siegel, S. *Non-Parametric Statistics for the Behavioral Sciences*. New York: McGraw-Hill, 1956.