Fundamenta Informaticae 95 (2009) 1–26

DOI 10.3233/FI-2009-181

IOS Press

An Improved Harmony Search Algorithm with Differential Mutation Operator

Prithwish Chakraborty, Gourab Ghosh Roy, Swagatam Das, Dhaval Jain

Department of Electronics and Telecommunication Engineering

Jadavpur University, Kolkata, India

Ajith Abraham

Machine Intelligence Research Labs (MIR Labs)

Scientific Network for Innovation and Research Excellence

P.O. Box 2259 Auburn, Washington 98071-2259, USA

ajith.abraham@ieee.org

Abstract. Harmony Search (HS) is a recently developed stochastic algorithm which imitates the music improvisation process. In this process, the musicians improvise their instrument pitches searching for the perfect state of harmony. Practical experiences, however, suggest that the algorithm suffers from the problems of slow and/or premature convergence over multimodal and rough fitness landscapes. This paper presents an attempt to improve the search performance of HS by hybridizing it with Differential Evolution (DE) algorithm. The performance of the resulting hybrid algorithm has been compared with classical HS, the global best HS, and a very popular variant of DE over a test-suite of six well known benchmark functions and one interesting practical optimization problem. The comparison is based on the following performance indices - (i) accuracy of final result, (ii) computational speed, and (iii) frequency of hitting the optima.

Keywords: Global optimization, Meta-heuristics, Harmony Search, Differential Evolution, Explorative power, Population variance

Address for correspondence: Machine Intelligence Research Labs (MIR Labs), Scientific Network for Innovation and Research Excellence, P.O. Box 2259 Auburn, Washington 98071-2259, USA.

1. Introduction

The computational drawbacks of existing derivative-based numerical methods have forced the researchers all over the world to rely on meta-heuristic algorithms founded on simulations to solve engineering optimization problems. A common factor shared by the meta-heuristics is that they combine rules and randomness to imitate some natural phenomena. Last few decades have seen an incredible growth in the field of nature-inspired meta-heuristics. Two families of algorithms that primarily constitute this field today are the Evolutionary Algorithms (EAs) [1, 2, 3] and the Swarm Intelligence (SI) algorithms [4, 5, 6].

In recent past, the computational cost having been reduced almost dramatically, researchers all over the world are coming up with new EAs on a regular basis to meet the demands of the complex, real-world optimization problems. Following this tradition, in 2001, Geem et al. proposed Harmony Search (HS) [7, 8, 9], a derivative-free, meta-heuristic algorithm, mimicking the improvisation process of music players. Since its inception, HS has been successfully applied to a wide variety of practical optimization problems like pipe-network design [10], structural optimization [11], vehicle routing problem [12], combined heat and power economic dispatch problem [13], and scheduling of multiple dam system [14]. HS may be viewed as a simple real-coded GA, since it incorporates many important features of GA like mutation, recombination, and selection.

The performance of classical HS over numerical benchmark functions like those used in [15] suffers from stagnation and/or false convergence. Mahdavi et al. proposed an improved HS algorithm (IHS) [15] that employs a novel method generating new solution vectors with enhanced accuracy and convergence speed. Recently, Omran and Mahdavi tried to improve the performance of HS by incorporating some techniques from swarm intelligence [5]. The new variant called by them as GHS (Global best Harmony Search) [16] reportedly outperformed three other HS variant over the benchmark problems. Fesanghary et al. [17] tried to improve the local search behavior of HS by hybridizing it with Sequential Quadratic Program (SQP) [18]. In the pitch adjustment phase of classical HS each vector is probabilistically perturbed in random step size of fixed maximum amplitude. This step is quite similar to the mutation process employed for perturbing the search agents in Evolutionary Strategy (ES) [19].

This article proposes a new strategy for changing, for perturbing each vector with a differential mutation operator borrowed from the realm of Differential Evolution (DE) [20, 21, 22]. The new mutation strategy is inspired by Zaharie's seminal work [23], where the author theoretically showed that the differential mutation scheme has greater explorative power than the ES type mutation schemes. The new algorithm, called DHS (Differential Harmony Search), has been extensively compared with classical HS, GHS, and IHS.

The rest of the paper is organized in the following way; Section 2 briefly outlines the classical HS, Section 3 introduces the hybrid GHS algorithm in sufficient details, the experimental and numerical results have been presented and discussed in Section 4 and finally the paper is concluded in Section 6.

2. The Harmony Search Metaheuristic Algorithm

Most of the existing meta-heuristic algorithms imitate natural, scientific phenomena, e.g. physical annealing of metal plates in simulated annealing, evolution in evolutionary algorithms and human memory in tabu search. Following similar trend, a new meta-heuristic algorithm can be conceptualized from the

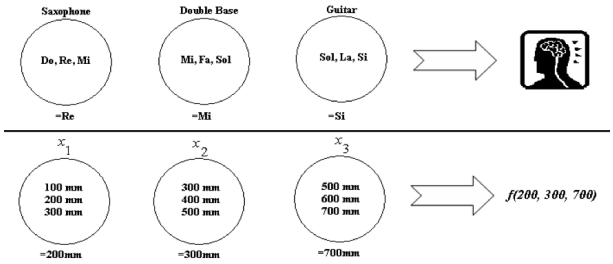


Figure 1. Analogy between music improvisation and engineering optimization (figure adopted from [8])

music improvisation process where musicians improvise their instruments' pitches searching for a perfect state of harmony. Although the estimation of a harmony is aesthetic and subjective, on the other hand, there are several theorists who have provided the standard of harmony estimation: Greek philosopher and mathematician Pythagoras (582 – 497BC) worked out the frequency ratios (or string length ratios with equal tension) and found that they had a particular mathematical relationship, after researching what notes sounded pleasant together. The octave was found to be a 1:2 ratio and what we today call a fifth to be a 2:3 ratio; French composer Leonin (1135 – 1201) is the first known significant composer of polyphonic "organum" which involved a simple doubling of the chant at an interval of a fifth or fourth above or below; and French composer Jean-Philippe Rameau (1683 – 1764) established the classical harmony theories in the book "*Treatise on Harmony*", which still form the basis of the modern study of tonal harmony [24].

In engineering optimization, the estimation of a solution is carried out by putting values of decision variables to objective function or fitness function and evaluating the function value with respect to several aspects such as cost, efficiency, and/or error. Just like music improvisation seeks a best state (fantastic harmony) determined by an aesthetic standard, optimization process seeks a best state (global optimum) determined by objective function evaluation; the pitch of each musical instrument determines the aesthetic quality, just as the objective function value is determined by the set of values assigned to each decision variable; aesthetic sound quality can be improved practice after practice, objective function value can be improved iteration by iteration. The HS meta-heuristic algorithm was derived based on natural musical performance processes that occur when a musician searches for a perfect state of harmony, such as during jazz improvisation.

The analogy between music improvisation and engineering optimization is illustrated in Figure 1. Each music player (saxophonist, double bassist, and guitarist) can correspond to each decision variable (x_1, x_2, x_3) and the range of each music instrument (saxophone = $\{$Do, Re, Mi$\}$; double bass = $\{$Mi, Fa, Sol$\}$; and guitar = $\{$Sol, La, Si$\}$) corresponds to the range of each variable value ($x_1 = \{100, 200, 300\}$; $x_2 = \{300, 400, 500\}$; and $x_3 = \{500, 600, 700\}$). If the saxophonist toots the note Re, the double bassist plucks Mi, and the guitarist plucks Si, their notes together make a new harmony (Re, Mi, Si). If this new harmony is better than existing harmony, the new one is kept. Likewise, the new solution vector (200mm, 300mm, 700mm) is kept if it is better than existing harmony in terms of objective function value. The harmony quality is improved by practice after practice.

Similarly in engineering optimization, each decision variable initially chooses any value within the possible range, together making one solution vector. If all the values of decision variables make a good solution, that experience is stored in each variable's memory and the possibility to make a good solution

is also increased next time. When a musician improvises one pitch, he (or she) has to follow any one of three rules:

1. playing any one pitch from his (or her) memory,
2. playing an adjacent pitch of one pitch from his (or her) memory, and
3. playing totally random pitch from the possible range of pitches.

Similarly, when each decision variable chooses one value in the HS algorithm, it follows any one of three rules:

1. choosing any one value from the HS memory (defined as memory considerations),
2. choosing an adjacent value of one value from the HS memory (defined as pitch adjustments) and
3. choosing totally random value from the possible range of values (defined as randomization).

According to the above concept, the HS meta-heuristic algorithm consists of the following five steps [7, 8]:

Step 1 : Initialization of the optimization problem and algorithm parameters.

Step 2 : Harmony memory initialization.

Step 3 : New Harmony improvisation.

Step 4 : Harmony memory update.

Step 5 : Repetition of Steps 3 and 4 until the termination criterion is satisfied.

Note that in what follows we shall denote vectors in bold.

Step 1. Initialization of the optimization problem and algorithm parameters:

In the first step, the optimization problem is specified as follows:

$$\text{Minimize (or Maximize) } f(\vec{x}) \tag{1}$$

subjected to $x_i \in X_i$, $i = 1, 2, \ldots, N$.

Where $f(\cdot)$is a scalar objective function to be optimized; $\vec{x}$ is a solution vector composed of decision variables x_i; X_i is the set of possible range of values for each decision variable x_i (continuous decision variable), that is ${}_Lx_i \leq X_i \leq {}_Ux_i$, where ${}_Lx_i$ and ${}_Ux_i$ are the lower and upper bounds for each decision variable respectively, and N is the number of decision variables. In addition, the control parameters of HS are also specified in this step. These parameters are the Harmony Memory Size (HMS) i.e. the number of solution vectors (population members) in the harmony memory (in each generation); Harmony Memory Considering Rate ($HMCR$); Pitch Adjusting Rate (PAR); and the Number of Improvisations (NI) or stopping criterion.

Step 2. Harmony memory initialization:

In the 2^{nd} step each component of each vector in the parental population (Harmony Memory), which is of size HMS, is initialized with a uniformly distributed random number between the upper and lower bounds $[{}_Lx_i, {}_Ux_i]$, where $1 \leq i \leq N$. This is done for the i-th component of the j-th solution vector using the following equation:

$$x_i^j = {}_Lx_i + rand(0,1) \cdot ({}_Ux_i - {}_Lx_i) \tag{2}$$

where $j = 1, 2, 3, \ldots, HMS$ and $rand(0,1)$ is a uniformly distributed random number between 0 and 1 and it is instantiated a new for each component of each vector.

Step 3. New Harmony improvisation:

In this step, a new harmony vector $\vec{x}' = (x'_1, x'_2, x'_3, x'_4, \ldots, x'_N)$ is generated based on three rules: (1) memory consideration, (2) pitch adjustment, and (3) random selection. Generating a new harmony is called 'improvisation'. In the memory consideration, the value of the first decision variable x'_1 for the new vector is chosen from any of the values already existing in the current HM i.e. from the set$\{x_1^1, \ldots, x_1^{HMS}\}$, with a probability $HMCR$. Values of the other decision variables $x'_2, x'_3, x'_4, \ldots, x'_N$ are also chosen in the same manner. The $HMCR$, which varies between 0 and 1, is the rate of choosing one value from the previous values stored in the HM, while $(1 - HMCR)$is the rate of randomly selecting a fresh value from the possible range of values.

$$x'_i \leftarrow \begin{cases} x_i \in \{x_i^1, x_i^2, x_i^3, \ldots, x_i^{HMS}\} & \text{with probability } HMCR \\ x_i \in X_i & \text{with probability } (1 - HMCR). \end{cases} \tag{3}$$

For example, an $HMCR = 0.80$ indicates that the HS algorithm will choose the decision variable value from historically stored values in the HM with an 80% probability or from the entire possible range with a 20% probability. Every component obtained by the memory consideration is further examined to determine whether it should be pitch-adjusted. This operation uses the parameter PAR (which is the rate of pitch adjustment) as follows:

Pitch Adjusting Decision for

$$x'_i = \begin{cases} x'_i \pm rand(0,1) \cdot bw & \text{with probability } PAR, \\ x'_i & \text{with probability } (1 - PAR), \end{cases} \tag{4}$$

where bw is an arbitrary distance bandwidth (a scalar number) and $rand()$ is a uniformly distributed random number between 0 and 1. Evidently step 3 is responsible for generating new potential variation in the algorithm and is comparable to mutation in standard EAs. Thus either the decision variable is perturbed with a random number between 0 and bw or otherwise it is left unaltered with a probability PAR or else it is left unchanged with probability $(1 - PAR)$.

Step 4. Harmony memory update:

If the new harmony vector $\vec{x}' = (x'_1, x'_2, x'_3, x'_4, \ldots, x'_N)$ is better than the worst harmony in the HM, judged in terms of the objective function value, the new harmony is included in the HM and the existing

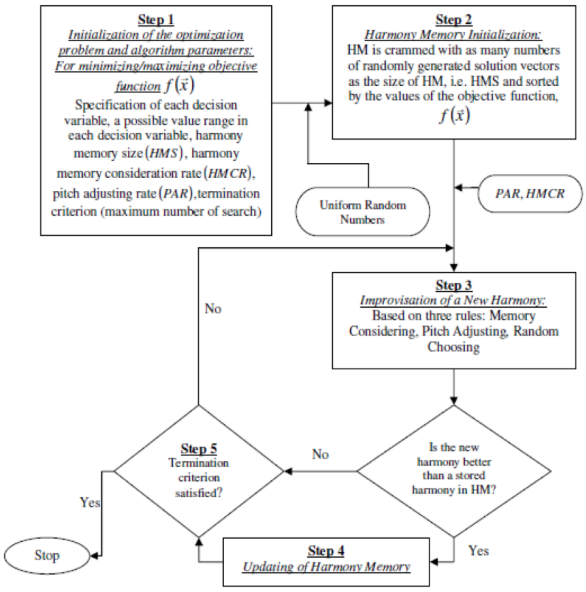


Figure 2. Flow-chart of the classical HS algorithm

worst harmony is excluded from the HM. This is actually the selection step of the algorithm where the objective function value is evaluated to determine if the new variation should be included in the population (Harmony Memory).

Step 5. Check stopping criterion:

If the stopping criterion (maximum number of improvisations) is satisfied, computation is terminated. Otherwise, steps 3 and 4 are repeated. The whole optimization process is represented with a flow-chart in Figure 2.

3. The Differential Harmony Search (DHS) Algorithm

Experiments with the classical HS meta-heuristics over the standard numerical benchmarks suggest that the algorithm does suffer from the problem of premature and/or false convergence, slow convergence especially over multimodal fitness landscape. Also the performance of the classical HS algorithm deteriorates with the growth of search space dimensionality. To circumvent this problems, in the present work we replace the pitch adjustment operation in classical HS with a mutation strategy borrowed from the realm of the DE algorithms. After experimenting with the members of the classical DE family of Storn and Price, we find that the mutation strategy employed by DE/rand/1 [22] produces best result when hybridized with classical HS. Following the perturbation process of DE/rand/1/bin, we mutate the target vector with the difference of two randomly selected population members. The process has been presented in equation (5).

$$\vec{x}'_i = \vec{x}'_i + F\left(\vec{x}_{r_1} - \vec{x}_{r_2}\right), \tag{5}$$

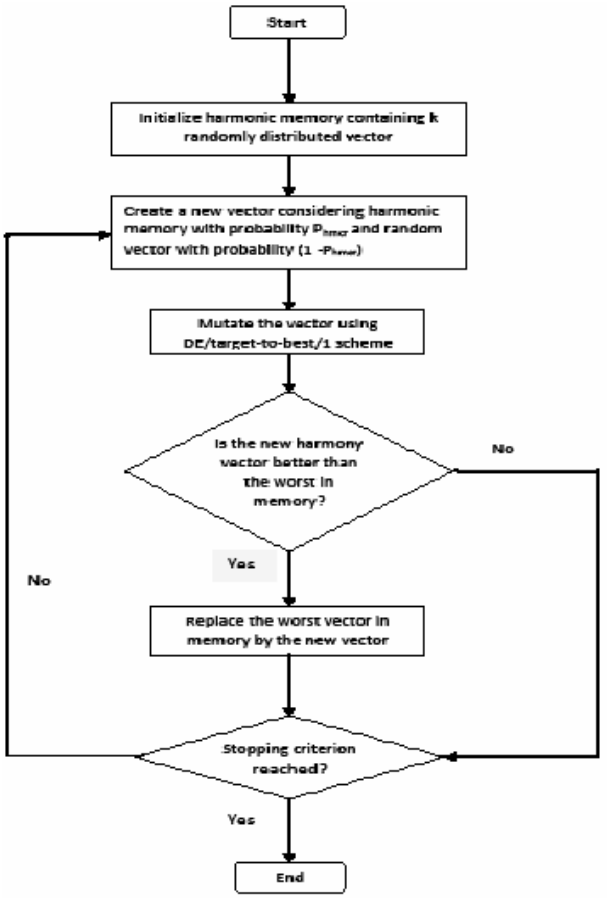


Figure 3. Flow-chart of DHS

where F is a scalar number called the scale factor. In our case the scale factor F is chosen to be a uniformly distributed random number between 0 and 1 as that gives better performance with the HS algorithm. From now onwards this modified HS equipped with a DE type mutation will be referred to as Differential Harmony Search (DHS). In what follows we undertake a simple mathematical analysis which shows that the DHS, under certain conditions, posses an increasing population variance (with generation) as compared to its classical counterpart. This ensures that the explorative power of DHS is on average greater than that of classical HS, which in turn results into better accuracy of the DHS algorithm. A flow-chart of the algorithm has been provided in Figure 3.

3.1. Performance analysis

The efficiency of most EAs depends on their extent of explorative and exploitative tendencies during the course of search. Exploitation means the ability of a search algorithm to use the information already collected and thus to orient the search more towards the goal while exploration is the process that allows introduction of new information into the population. Exploration helps the algorithm to quickly search the new regions of a large search-volume. Proper balance between these two characteristics results into

enhanced performance [25]. Generally EAs explore the search space by the (genetic) search operators, while exploitation is done via *selection* that promotes better individuals to the next generation.

Expected value of population variance form one generation to another generation is a good estimate of the explorative power of a stochastic search algorithm. In this paper we analyze the evolution of the population-variance of HS and its influence on the explorative power of the algorithm. We first find an analytical expression for the population-variance of DHS and then compare it with the expected population variance of classical HS to show that the formal algorithm possesses greater explorative power. Since in HS type algorithms, each dimension is perturbed independently, without the loss of generality, we carry forward our analysis for single-dimensional population members.

In DHS the perturbations are made independently for each component. As such, without loss of generality, we may conduct the analysis on one-dimensional elements. Let us consider a population of scalars $x = \{x_1, x_2, \ldots, x_m\}$ with elements $x_1 \in R$. The variance of this population is given by:

$$Var(x) = \frac{1}{m}\sum_{i=1}^{m}(x_i - \bar{x})^2 = \overline{x^2} - \bar{x}^2 \tag{6}$$

where, $\bar{x}$ is population mean and $\overline{x^2}$ is quadratic population mean.

If the elements of the population are affected by some random elements, $Var(x)$ will be a random variable and $E[Var(x)]$ will be the measure of the explorative power. Variance of DHS is calculated below with the following assumptions.

Theorem 3.1. Let $x = \{x_1, x_2, \ldots, x_m\}$ be the current population, y be an intermediate vector obtained after random selection and harmony memory consideration and z the vector obtained after pitch adjusting y. Let $w = \{w_1, w_2, \ldots, w_m\}$ be the final population after selection. If

$$P_{HMCR} = \text{Harmonic Memory Consideration Probability}$$

and, we consider the allowable range for the new values of x is $\{x_{\min}, x_{\max}\}$ where $x_{\max} = a$, $x_{\max} = -a$ and the required random numbers are continuously uniformly distributed between 0 and 1, then

$$E[V(w)] = \frac{2}{3}V(x) + \frac{a^2}{3}\left(\frac{m-1}{m}\right)(1 - P_{HMCR}) + \\ + \frac{\overline{x^2}}{m}\left(P_{HMCR} + \frac{m-1}{m}\right) - \frac{\bar{x}^2 P_{HMCR}}{m^2}\{P_{HMCR} + 2(m-1)\}$$

Proof:
Here $x = \{x_1, x_2, \ldots, x_m\}$ is the current population. So the population mean $\bar{x}$ is given by

$$\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{7}$$

And, quadratic population mean $\overline{x^2}$ as

$$\overline{x^2} = \frac{1}{m}\sum_{i=1}^{m} x_i^2 \tag{8}$$

y is an intermediate vector obtained after random selection and harmony memory consideration. It is obtained as:

$$y = \begin{cases} x_l & \text{with probability } P_{HMCR} \\ x_r & \text{with probability } (1 - P_{HMCR}), \end{cases} \tag{9}$$

where l and k are two elements of $\{1, 2, \ldots, m\}$ and x_r is a new random number in the allowable range $\{x_{\min}, x_{\max}\}$ or $\{-a, a\}$ and the index k is a random variable with values in $\{1, 2, \ldots, m\}$, with the probability $P_k = P(i = k) = \frac{1}{m}$.

Therefore,

$$\begin{aligned} E(x_l) &= \sum_l x_l P_l = \frac{1}{m} \sum_l x_l = \bar{x} \\ E(x_l^2) &= \sum_l x_l^2 P_l = \frac{1}{m} \sum_l x_l^2 = \overline{x^2} \end{aligned} \tag{10}$$

Similarly, those for x_r may be given as in Eq. (11) with detailed proof being provided in Appendix B.

$$\begin{aligned} E(x_r) &= 0 \\ E(x_r^2) &= \frac{a^2}{3} \end{aligned} \tag{11}$$

Using Eqs. (10) and (11), the following relations can be inferred

$$E(y) = E(x_l)P_{HMCR} + E(x_r)(1 - P_{HMCR}) = P_{HMCR}\bar{x} \tag{12}$$

$$E(y_2) = E(x_l^2)P_{HMCR} + E(x_r^2)(1 - P_{HMCR}) = \overline{x^2}P_{HMCR} + \frac{a^2}{3}(1 - P_{HMCR}) \tag{13}$$

Now z is the vector obtained after mutating y. This has the following structure:

$$z = y + F(x_{\beta_1} - x_{\beta_2}),$$

where F is a uniformly distributed random number varying between 0 and 1. As such following Appendix A, we can say that

$$\begin{aligned} \overline{F} &= E(F) = \frac{1}{2} \\ \overline{F^2} &= E(F^2) = \frac{1}{3} \end{aligned} \tag{14}$$

Also, x_{β_1} and x_{β_2} are two randomly chosen members from x such that $\beta_1 \neq \beta_2$.
Thus following equations hold

$$\begin{aligned} E(x_{\beta_1}) &= E(x_{\beta_2}) = \bar{x} \\ E[x_{\beta_1}^2] &= E[x_{\beta_2}^2] = \overline{x^2} \\ E[x_{\beta_1} x_{\beta_2}] &= \frac{1}{m(m-1)} \left\{ (m\bar{x})^2 - m\overline{x^2} \right\} \end{aligned} \tag{15}$$

Here, the first two equations of (15) are self-evident from (10) while, the last one is derived in Appendix C.

Therefore, using (15) we can write

$$\begin{aligned} E(z) &= E(y) + E(F)E(x_{\beta_1} - x_{\beta_2}) \\ &= E(y) + \overline{F}\left[E(x_{\beta_1}) - E(x_{\beta_2})\right] \\ &= \bar{x}P_{HMCR} \end{aligned}$$

and,

$$\begin{aligned} E(z^2) &= E(y + F(x_{\beta_1} - x_{\beta_2})^2) \\ &= E\left(y^2 + F^2(x_{\beta_1}^2 + x_{\beta_2}^2 - 2x_{\beta_1}x_{\beta_2}) + 2Fy(x_{\beta_1} - x_{\beta_2})\right) \\ &= E[y^2] + \overline{F^2}(E[x_{\beta_1}^2] + E[x_{\beta_2}^2] - 2E[x_{\beta_1}x_{\beta_2}]) + 2\overline{F}E[y](E[x_{\beta_1}] - E[x_{\beta_2}]) \\ &= E[y^2] + 2\overline{F^2}\frac{m}{m-1}(\overline{x^2} - \bar{x^2}). \end{aligned} \tag{16}$$

Now, $w = \{w_1, w_2, \ldots, w_m\}$ is the final population obtained by replacing y with a random member of the original population. Each element w_k of the final population may be represented by the following structure:

$$w_k = \begin{cases} z & \text{with probability } p = \frac{1}{m} \\ x_k & \text{with probability } p = (1 - \frac{1}{m}) \end{cases}$$

Therefore,

$$\begin{aligned} E(w_k) &= E(z)p + E(x_k)(1-p) \\ &= \frac{1}{m}\bar{x}P_{HMCR} + \frac{m-1}{m}\bar{x} \\ E(w_k^2) &= E(z^2)p + E(x_k^2)(1-p). \end{aligned} \tag{17}$$

Now if $\bar{w}$ represent the population mean of the final target population then,

$$\begin{aligned} \bar{w} &= \frac{1}{m}\sum w_k \\ E(\overline{w}^2) &= E\left(\frac{1}{m}\sum w_k\right)^2 \\ &= \frac{1}{m^2}E\left(\sum w_k^2 + \sum\sum_{l\neq j} w_l w_j\right) \\ &= \frac{1}{m^2}\left\{\sum E(w_k^2) + \sum\sum_{l\neq j} E(w_l w_j)\right\} \\ &= \frac{1}{m^2}\left\{mE(w_k^2) + m(m-1)E^2(w_k)\right\} \end{aligned}$$

Also, if $\overline{w^2} = \frac{1}{m}\sum w_k^2$ then,

$$E(\overline{w^2}) = \frac{1}{m}E\left(\sum w_k^2\right) = E(w_k^2)$$

Variance of final population may be given as

$$V(w) = \overline{w^2} - \bar{w}^2$$

So expected population variance may be given as

$$\begin{aligned} E\left(V(w)\right) &= E[\overline{w^2} - \bar{w}^2] \\ &= E(w_k^2) - \frac{1}{m}E(w_k^2) + \frac{m-1}{m}E^2(w_k) \\ &= \frac{m-1}{m}\left[E(\overline{w_k^2}) - E^2(w_k)\right] \end{aligned}$$

Putting the values from Eq. (17) and simplifying the results we get,

$$\begin{aligned} E(V(w)) =& \frac{2}{3}V(x) + \frac{a^2}{3}\left(\frac{m-1}{m}\right)(1 - P_{HMCR}) + \frac{\overline{x^2}}{m}\left(P_{HMCR} + \frac{m-1}{m}\right) - \\ &- \frac{\bar{x}^2 P_{HMCR}}{m^2}\Big\{P_{HMCR} + 2(m-1)\Big\} \end{aligned} \tag{18}$$

□

Lemma 3.1. Considering typical values of BW $(= 0.25)$ and P_{PAR} $(= 0.63)$ and assuming $P_{HMCR} = 1$ and $m = 8$, inter-generation variance of DHS algorithm is greater than HS.

Proof:
For HS the inter-generation variance was given by [15] as

$$E(Var(z)) = \frac{(m-1)}{m} \cdot \left[\begin{array}{l} HMCR \cdot var(x) + HMCR \cdot (1 - HMCR) \cdot \bar{x}^2 + \\ + HMCR \cdot (1 - HMCR) \cdot PAR \cdot bw \cdot \bar{x} + \\ + HMCR \cdot PAR \cdot bw^2 \cdot \left(\frac{1}{3} - \frac{HMCR \cdot PAR}{4}\right) + \\ + \frac{a^2}{3} \cdot (1 - HMCR) \end{array}\right] \tag{19}$$

Putting typical values of bw $(= 0.25)$ and P_{PAR} $(= 0.63)$ and assuming $P_{HMCR} = 1$ and $m = 8$

$$E[Var(z)]_{HS} = 0.85Var(x) + 0.0069 \tag{20}$$

And, for DHS putting $P_{HMCR} = 1$ and $m = 8$ we get,

$$E[Var(z)]_{DHS} = 0.901.Var(x) \tag{21}$$

$$\therefore E[Var(z)]_{DHS} - E[Var(z)]_{HS} > 0 \tag{22}$$

□

The analysis undertaken above ignores the selection process. As such the actual value of variance of the function is slightly lesser than the computed one. Nevertheless, it serves to give a value for variance and proves superiority of DHS than the HS algorithm I terms of the explorative search behavior.

Table 1. Benchmark Functions Used

Function	Mathematical Representation
Sphere Model	$f_1(\vec{x}) = \sum_{i=1}^{n} x_i^2$
Rosenbrock	$f_2(\vec{x}) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$
Rastrigin	$f_3(\vec{x}) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$
Griewank	$f_4(\vec{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$
Ackley	$f_5(\vec{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i\right) + 20 + e$
Shekel's Foxholes	$f_6(\vec{x}) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right]^{-1}$ where $(a_{ij}) = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \dots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \dots & 32 & 32 & 32 \end{pmatrix}$

4. The Experimental Setup

4.1. Benchmark Functions Used

The performance of the DHS algorithm has been evaluated on a test suite of seven well-known benchmarks (Table 1) [26]. In Table 1, n represents the number of dimensions (we used $n = 10, 20, 30$). The first two test functions are unimodal, having only one minimum. The others are multimodal, with a considerable number of local minima in the region of interest. All benchmark functions have the global minimum at the origin or very near to the origin [26]. Table 2 summarizes the initialization and search ranges used for these functions. An asymmetrical initialization procedure has been used here following the work reported in [27]. Finally, in order to validate the versatality of the proposed algorithm, simulation resutls have been provided for few shifted functions, constrained functions as well as two well known complicated functions in Tables 6, **??**, and **??**.

4.2. Competitor Algorithms and Parametric Set up

Simulations were carried out to obtain a comparative performance analysis of the proposed method with respect to: (a) classical HS [8], (b) IHS [15], (c) GHS [16], and one most popular DE variant called DE/rand/1/bin [22]. For all the HS-variants we used the same population size for harmonic memory and HMS was chosen to be equal to 10. IHS generally introduces a number of parameters, which should be tuned for different problems. Here we disallow any kind of hand tuning of any of the parameters over any benchmark problem. Hence, after performing a series of hand-tuning experiments, for IHS we use

Table 2. Search Ranges of Benchmark Functions

f	Global Optimum	Range of search	Range of Initialization
f_1	$f_1(\vec{0}) = 0$	$(-100, 100)^n$	$(50, 100)^n$
f_2	$f_2(\vec{1}) = 0$	$(-100, 100)^n$	$(15, 30)^n$
f_3	$f_3(\vec{0}) = 0$	$(-10, 10)^n$	$(2.56, 5.12)^n$
f_4	$f_4(\vec{0}) = 0$	$(-600, 600)^n$	$(300, 600)^n$
f_5	$f_5(\vec{0}) = 0$	$(-32, 32)^n$	$(15, 32)^n$
f_6	$f_6(-31.95, -31.95) = 0.998$	$(-65.536, 65.536)^2$	$(0, 65.536)^2$

$HMCR = 0.95$, $PAR_{\min} = 0.35$, $PAR_{\max} = 0.99$, $bw_{\min} = 1.00e-06$, and $bw_{\max} = \frac{1}{20(x_{\max} - x_{\min})}$ over all the unconstrained problems as our preliminary experiments suggest that IHS with this parametric setup provides uniformly good results over all the benchmarks presented here. Similarly for GHS, we choose $PAR_{\min} = 0.01$, $PAR_{\max} = 0.99$, and $HMCR = 0.9$. For the classical HS algorithm we selected $HMCR = 0.9$, $PAR = 0.3$ and $bw = 0.01$ following [15].

In case of DE/rand/1/bin algorithm, Np or population size is chosen as 10 times the search-space dimensionality of the problem. We also took $Cr = 0.9$ and $F = 0.8$ [22].

4.3. Simulation Strategy

To make the comparison fair, the populations for all the competitor algorithms (for all problems tested) were initialized using the same random seeds. We choose the number of fitness evaluations (FEs) as a measure of computation time instead of 'generations' or 'iterations' since different algorithms usually perform different amounts of tasks in their inner loops. To judge the accuracy of different algorithms, we first let each of them run for a very long time over every benchmark function, until the number of FEs exceeds a given upper limit (which was fixed depending on the complexity of the problem). Fifty independent runs of each of the five contestant algorithms were carried out and the average and the standard deviation of the best-of-run values were recorded. In order to compare the speeds of different algorithms, we select a threshold value of the objective function for each benchmark problem. We run each algorithm on a function and stop as soon as the best objective function value determined by the algorithm falls below the predefined threshold. Then we note the number of FEs the algorithm takes. A lower number of FEs corresponds to a faster algorithm. Note that each run of an algorithm is continued until a maximum number of FEs is exceeded. The maximum number of FEs for each function is same as the number of FEs for the previous experiment.

5. Experimental Results

5.1. Results for Numerical Benchmarks

Table 3 compares the algorithms on the quality of the optimum solution. The mean and the standard deviation (within parentheses) of the best-of-run values for 30 independent runs of each of the four algorithms are presented. Each algorithm was run up to a predetermined number of FEs (depending

Table 3. Average and the standard deviation of the best-of-run solution for 50 runs tested on seven benchmark functions

Fun	D	Max^n FE	Mean Best Value (Standard Deviation)				
			DHS	IHS	Classical HS	GHS	DE/rand/1/bin
f_1	10	50,000	**5.884900e-007** (**6.646400e-006**)	1.639500e-002 (6.708000e-003)	2.171200e-002 (4.732000e-003)	6.318541e+000 (3.035200e+000)	7.733437e-004 (6.7679e-007)
	20	1×10^5	**6.429200e-006** (**8.725800e-006**)	2.893210e-001 (4.892000e-002)	1.426900e-001 (2.220100e-002)	9.618317e+000 (3.994490e+000)	6.845594e-005 (4.98644e-004)
	30	2×10^5	**9.389000e-005** (**6.520000e-004**)	9.990920e-001((1.238010e-001)	2.542532e-001 (3.055009e-002)	1.769400e+000 (8.803482e-001)	8.478346e-004 (4.664e-004)
f_2	10	50,000	**1.568433e-002** (**2.357514e-003**)	9.753570e+000 (1.082548e+000)	1.031847e+001 (4.172880e-001)	2.254647e+003 (2.179374e+003)	1.622342e-001 (4.2573e-005)
	20	1×10^5	**2.864111e+000** (**1.497365e+000**)	5.959299e+001 (2.589792e+001)	4.663445e+001 (2.596948e+001)	4.468664e+003 (3.825431e+003)	8.86379e-001 (3.237e-005)
	30	2×10^5	**1.045937e+001** (**3.721272e+001**)	6.15397e+001 (1.100124e+001)	5.553776e+001 (1.161574e+001)	6.355397e+002 (3.543231e+002)	5.23198e+001 (3.91179e-002)
f_3	10	50,000	**4.927550e-003** (**7.426300e-001**)	2.173265e+001 (6.210750e+000)	2.306190e+001 (6.695788e+000)	5.189726e+000 (1.308674e+000)	5.246318e-002 (3.94711e-002)
	20	1×10^5	**6.262530e-001** (**5.226480e-001**)	8.543985e+001 (1.381052e+001)	6.930525e+001 (1.343840e+001)	8.764900e+000 (3.102358e+000)	1.870534e-001 (6.28432e-002)
	30	2×10^5	**2.309900e-002** (**1.693000e-002**)	1.853878e+002 (1.964701e+001)	1.166632e+002 (1.370221e+001)	3.512368e+000 (1.055009e+000)	4.838537e+000 (3.872431e+001)
f_4	10	50,000	**2.701000e-002** (**9.005000e-003**)	9.035700e-002 (1.298720e-001)	6.340700e-002 (7.265900e-002)	2.266825e-001 (1.400353e-001)	2.327581e-003 (4.6573e-003)
	20	1×10^5	**4.959500e-004** (**5.200000e-004**)	7.612600e-002 (9.901100e-002)	6.228300e-002 (1.311900e-002)	2.117310e-001 (1.109799e-001)	6.87334e-002 (5.32529e-002)
	30	2×10^5	**1.316900e-002** (**3.560000e-004**)	7.334800e-001 (8.200000e-003)	7.301747e-001 (1.297544e-002)	9.785366e-001 (1.295663e-001)	2.19345e-001 (1.186345e-001)
f_5	10	50,000	**8.815280e-006** (**7.634358e-006**)	1.083798e+001 (8.340050e-001)	1.036288e+001 (6.978520e-001)	8.643679e+000 (4.795448e-001)	4.593572e-005 (9.44871e-005)
	20	1×10^5	**3.647528e-005** (**6.936542e-004**)	1.259577e+001 (7.375590e-001)	1.163075e+001 (7.591900e-001)	8.297712e+000 (2.890263e-001)	1.736824e-003 (4.09364e-003)
	30	2×10^5	**4.453663e-004** (**3.657234e-003**)	1.337918e+001 (5.676430e-001)	1.142695e+001 (5.157850e-001)	7.598800e+000 (5.788477e-002)	6.45453e-002 (3.432434e-002)
f_6	2	1×10^4	**9.9800390e-001** (**1.1562e-008**)	9.9860553e-001 (4.26578e-003)	1.080039e+000 (1.138973e-002)	1.280884e+000 (1.89843e-004)	9.9890146e-001 (1.57087e-003)

upon the complexity of the problem). Missing values of standard deviation indicate a zero standard deviation. The best solution in each case has been shown in bold.

Table 4 shows results of unpaired t-tests between DHS and the best of the three competing algorithms in each case (standard error of difference of the two means, 95% confidence interval of this difference, the t value, and the two-tailed P value). For all cases in Table 4, sample size = 60 and number of degrees of freedom = 58. It is interesting to see from Tables 3 and 4 that the proposed DHS algorithm meets or beats the nearest competitor in a statistically significant way.

Table 5 reports the number of runs (out of 50) that managed to find the optimum solution (within the given tolerance) without exceeding the maximum no. of FEs, the mean number of FEs, and standard deviations (within parenthesis) required by the algorithms to converge within the prescribed threshold value. Entries marked as 0 indicate that no runs of the corresponding algorithm converged below the threshold objective function value. Missing values of standard deviation in these tables also indicate a zero standard deviation.

Tables 5 shows that, not only does DHS yield the most accurate results for nearly all the benchmark problems, but it does so consuming the least amount of computational time. In addition, the number of runs that converge below a pre-specified cut-off value is also greatest for EHS over most of the benchmark problems covered here. This indicates the higher robustness (i.e. the ability to produce similar results over repeated runs on a single problem) of the algorithm as compared to its other four competitors.

Table 4. Results of unpaired t-tests on the data of Table 3

Fn, Dim	*Std. Err*	t	95% Conf. Intvl	Two-tailed P	Significance
f_1, 20	0.016	4.7691	-0.11130337 to -0.04549263	< 0.0001	**Extremely significant**
f_1, 30	0.006	43.008	-0.25620711 to -0.23341489	< 0.0001	**Extremely significant**
f_3, 10	0.275	17.067	-5.237055 to -4.137545	< 0.0001	**Extremely significant**
f_3, 20	0.575	14.201	-9.311759 to -7.011041	< 0.0001	**Extremely significant**
f_3, 30	0.567	6.1563	-4.6233206 to -2.3544994	< 0.0001	**Extremely significant**
f_4, 10	0.013	2.7723	-0.06381928 to -0.01030072	0.0075	**Very Statistically significant**
f_5, 10	0.090	13.198	-1.361613 to -1.002987	< 0.0001	**Extremely significant**
f_4, 20	0.055	15.059	-0.9323926 to -0.7136074	< 0.0001	**Extremely significant**

Usually in the community of stochastic search algorithms, robust search is weighted over the highest possible convergence rate [28]. In Figure 4 we have graphically presented the rate of convergence of all the methods for all the functions (in 30 dimensions). These results show that the proposed method leads to significant improvements in most cases.

The functions $f_7 - f_{11}$, listed below, are examples of GA-hard constrained optimization problems, extensively studied in [29, 30, 31, 32, 33] and in this paper we have transformed them into an uncontrained one by adding a penalty term in the following way:

$$\psi_i(\vec{x}) = f_i(\vec{x}) + A_i \cdot \sum_{j=1}^{C_i} \max\{0, g_j(\vec{x})\} \tag{23}$$

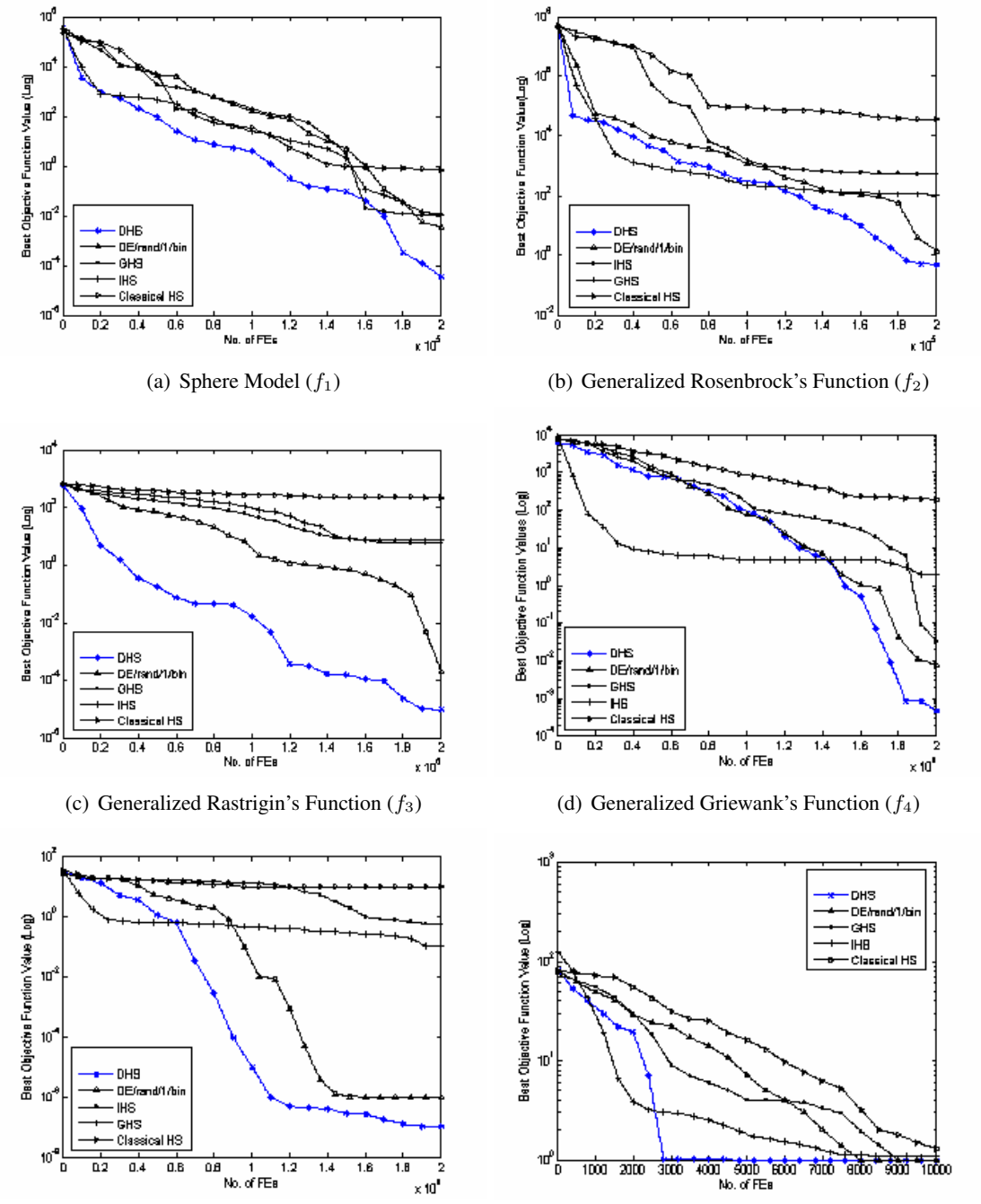
where $i = 7, \ldots, 11$, C_i is the number of constraints with the i-th benchnark, and A_i is the static penalty coefficient for the i-th contrained benchmark problem. Functions f_8 and f_{11} include an equality constraint, which has been converted into inequality constraint by using $|h(\vec{x}) - \delta| \leq 0$ using the degree of violation $\delta = 10^{-4}$. Values of the static penalty coefficients are as follows [34]: $A_7 = 0.5$, $A_8 = 10^5$, $A_9 = 10^4$, $A_{10} = 500$, and $A_{11} = 10$. The following constrained optimization functions were considered in this paper:

Minimize

$$f_7(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$$

Table 5. Number of runs (out of 50) converging to the cut off fitness value for six benchmark functions

Fun	D	Max^n FE	No. of runs converging to the cut off				
			DHS	IHS	Classical HS	GHS	DE/rand/1/bin
f_1	10	1.00e-005	**50,26253.20 (445.34)**	50,32044.22 (610.298)	50,13423.68 (341.827)	42,41544.34 (8532.261)	50,33932.64 (1450.492)
	20	1.00e-005	**31,68931.5 (5712.65)**	42,18298.21 (1340.34)	44,7364.32 (2235.83)	22,85367.86 (4540.12)	34,46296.46 (674.25)
	30	1.00e-005	**29,172228.72 (6473.45)**	45,84712.34 (8952.34)	41,36523.46 (7326.74)	16,74782.68 (6638.93)	21,23876.62 (7821.63)
f_2	10	1.00e-003	**24,454563.25 (7653.44)**	10,137628.60 (10183.23)	0	11,167322.43 (0.4291)	13,72874.34 (67232.91)
	20	1.00e-003	**13,40943.68 (5831.84)**	0	1,48585	0	0
	30	1.00e-003	**8,83723.25**	0	0	0	0
f_3	10	1.00e-005	**12,113817.52 (12723.837)**	0 0	0 0	0 0	0 0
	20	1.00e-005	**3,179834.33 (21353.82)**	0	0	0	0
	30	1.00e-005	50,34983.68 (4481.73)	32,39782.57 (7434.12)	0	0	**50,27474.24 (5327.08)**
f_4	10	1.00e-005	50,83924.16 (4028.47)	25,88731.04 (2139.57)	0	0	**50,73483.50 (11142.76)**
	20	1.00e-005	**43,129834.31 (10835.44)**	0	27,122833.73 (9738.62)	0	0
	30	1.00e-005	**12,154982.76 (32432.87)**	0	19,167909.57 (5672.89)	0	0
f_5	10	1.00e-005	**32,38769.46 (3692.58)**	21,37583.67 (7432.82)	26,41029.75 (3732.68)	0	0
	20	1.00e-005	**25,76093.80 (4827.57)**	17,75834.83 (6907.89)	15,85734.46 (5003.68)	0	0
	30	1.00e-005	**50,27243.44 (447.03)**	32,29583.49 (1241.57)	47,37263.92 (1832.45)	7,48374.34 (227.48)	15,35563.46 (1519.46)



(a) Sphere Model (f_1)

(b) Generalized Rosenbrock's Function (f_2)

(c) Generalized Rastrigin's Function (f_3)

(d) Generalized Griewank's Function (f_4)

(e) Generalized Ackley's Function (f_5)

(f) Shekel's Foxhole Function (f_6)

Figure 4. Progress to the optimum solution for numerical benchmarks

Subject to:

$$\begin{aligned}
g_1(x) &= 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0 \\
g_2(x) &= 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0 \\
g_3(x) &= 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0 \\
g_4(x) &= -8x_1 + x_{10} \leq 0 \\
g_5(x) &= -8x_2 + x_{11} \leq 0 \\
g_6(x) &= -8x_3 + x_{12} \leq 0 \\
g_7(x) &= -2x_4 - x_5 + x_{10} \leq 0 \\
g_8(x) &= -2x_6 - x_7 + x_{11} \leq 0 \\
g_9(x) &= -2x_8 - x_9 + x_{12} \leq 0
\end{aligned}$$

where the bounds are $0 \leq x_i \leq 1$ $(i = 1, \ldots, 9)$, $0 \leq x_i \leq 100$ $(i = 10, 11, 12)$ and $0 \leq x_{13} \leq 1$. The global optimum is at $x^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$ where $f(x^*) = -15$. Constraints g_1, g_2, g_3, g_4, g_5 and g_6 are active.

Maximize

$$f_8(x) = \left(\sqrt{n}\right)^n \prod_{i=1}^{n} x_i$$

Subject to

$$h(x) = \sum_{i=1}^{n} x_i^2 - 1 = 0,$$

where $n = 10$ and $0 \leq x_i \leq 1,\ (i = 1, \ldots, n)$. The global optimum is at

$$x^* = \frac{1}{\sqrt{n}},$$

$i = (1, \ldots, n)$ where $f(x^*) = 1$

Minimize

$$f_9(x) = 5.3578547x_2^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$$

Subject to:

$$\begin{aligned}
g_1(x) &= 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0 \\
g_2(x) &= -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0 \\
g_3(x) &= 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 0 \\
g_4(x) &= -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0 \\
g_5(x) &= 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0010985x_3x_4 - 25 \leq 0 \\
g_6(x) &= -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0010985x_3x_4 + 20 \leq 0
\end{aligned}$$

where the bounds are $78 \leq x_1 \leq 102$, $33 \leq x_2 \leq 45$ and $27 \leq x_i \leq 45$, $(i = 3, 4, 5)$. The global optimum is at $x^* = (78, 33, 29.995256025682, 45, 36.775812905788)$ where $f(x^*) = -30665.539$. Constraints g_1 and g_6 are active.

Minimize

$$f_{10}(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$$

Subject to:

$$\begin{aligned} g_1(x) &= -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0 \\ g_2(x) &= -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0 \\ g_3(x) &= -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0 \\ g_4(x) &= 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0 \end{aligned}$$

where $-10 \leq x_i \leq 10$, $(i = 1, \ldots, 7)$.

The global optimum is at

$$x^* = (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227)$$

where $f(x^*) = 680.6300573$. Constraints g_1 and g_4 are active.

Minimize

$$f(x) = x_1^2 + (x_2 - 1)^2$$

Subject to:

$$h(x) = x_2 - x_1^2 = 0$$

where $-1 \leq x_1 \leq 1$ and $-1 \leq x_2 \leq 1$. The global optimum is at $x^* = (\pm\frac{1}{\sqrt{2}}, \frac{1}{2})$, where $f(x^*) = 0.75$.

Besides comparing DHS with the two state-of-the-art HS variants we also take into account here another evolutionary algorithm – RY [29] that is especially devised for optimizing constrained objective functions. Numerical results for RY have been taken from [29]. Since the authors in [29] used a termination criterion of 1750 generations (corresponding to 350 000 FEs) for the RY algorithm over the five benchmarks we test here, in order to make the comparison fair, we compare both the qualities of their final solutions and the computational cost at the same. Accordingly, the termination criterion of the three HS variants is that the quality of the best solution cannot further be improved in the successive 50 generations for each function. We reported the results for 30 independent runs of DHS, GHS, and IHS with different random seeds in table 6. In each run the three HS variants start from the same initial population. The results for the RY algorithm are also for 30 independent trials for each function.

Table 6 indicates that DHS appears to be computationally more efficient than the two contestant HS-variants as well as the RY algorithm at least over the test-suite used here for constrained optimization problems.

5.2. Application to the Spread Spectrum Radar Poly-phase Code Design Problem

A famous problem of optimal design arises in the field of spread spectrum radar poly-phase codes [35]. Such a problem is very well-suited for the application of global optimization algorithms like DE. The problem can be formally stated as:

$$\text{Global min } f(\vec{x}) = \max\{\varphi_1(\vec{x}), \ldots, \varphi_{2m}(\vec{x})\}, \tag{24}$$

Table 6. Simulation results for constrained functions $f_7 - f_{11}$(best out of 30 runs).

Func	Value type	DHS	IHS	GHS	RY Algorithm
f_7	Mean	**-15.000**	-14.9924	-14.9961	**-15.000**
	Std. Dev	**0.00**	0.00167	0.00235	**0.00**
	Mean No. of FEs (Std. Dev)	**35635.64 (1753.46)**	176390.82 (20193.57)	152 738.56 (11378.36)	350
	% of Trials giving Feasible Solutions	**100%**	70%	90%	**100%**
f_8	Mean	**-30665.539**	-30664.831	-30662.242	**-30665.539**
	Std. Dev	$\mathbf{2.732 \times 10^{-10}}$	4.758×10^{-4}	1.468×10^{-3}	2×10^{-5}
	Mean No. of FEs (Std. Dev)	**65783.72 (3726.68)**	104729.84 (6579.46)	85082.07 (7547.84)	350
	% of Trials giving Feasible Solutions	**100%**	80%	90%	**100%**
f_9	Mean	**5126.555182**	5128.942	5130.382	5128.881
	Std. Dev	**0.6526**	I.12	III.89	3.V
	Mean No. of FEs (Std. Dev)	**67932.56 (4526.7)**	217841.49 (10928.52)	117264.68 (9825.37)	350
	% of Trials giving Feasible Solutions	**100%**	90%	87%	**100%**
f_{10}	Mean	680.658897	721.472	694.738	**680.656**
	Std. Dev	0.00133	10.732	2.652	0.036
	Mean No. of FEs (Std. Dev)	**70936.48 (4838.77)**	192831.29 (26712.43)	90293.76 (7362.48)	350
	% of Trials giving Feasible Solutions	**100%**	76%	84%	**100%**
f_{11}	Mean	**0.75000**	0.750472	0.767613	**0.75000**
	Std. Dev	$\mathbf{2.638 \times 10^{-12}}$	0.0084	0.067	8×10^{-5}
	Mean No. of FEs (Std. Dev)	**28983.78 (3008.46)**	57583.30 (1272.84)	48723.42 (2957.03)	350
	% of Trials giving Feasible Solutions	**100%**	84%	79%	**100%**

where $\vec{x} = \{(x_1, ..., x_n) \in \Re^n | 0 \leq x_j \leq 2\pi,\ j = 1, \ldots, n\}$ and $m = 2n - 1$, with

$$\begin{aligned}
\varphi_{2i-1}(\vec{x}) &= \sum_{j=i}^{n} \cos\left(\sum_{k=|2i-j-1|-1}^{j} x_k\right)\ i = 1, 2, \ldots, n \\
\varphi_{2i}(\vec{X}) &= 0.5 + \sum_{j=i+1}^{n} \cos\left(\sum_{k=|2i-j-1|-1}^{j} x_k\right),\ i = 1, 2, \ldots, n-1 \\
\varphi_{m+i}(\vec{X}) &= -\varphi_i(\vec{X}),\ i = 1, 2, \ldots, m
\end{aligned} \tag{25}$$

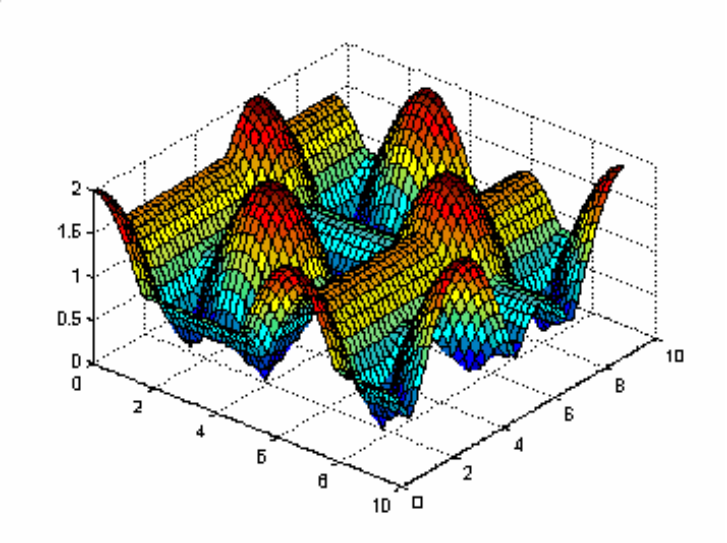


Figure 5. $f(\vec{X})$ of equation (22) for $D = 2$.

Table 7. Average and standard deviation (in parentheses) of the best-of-run solutions for 50 runs over the spread spectrum radar poly-phase code design problem (number of dimensions $n = 19$ and $n = 20$). For all cases each algorithm was run up to 5×10^6 FEs.

N	Mean Best-of-run solution (standard deviation)					
	DE/rand/1/bin	IHS	HS	GHS	DHS	Statistical Significance (by unpaired t-test)
19	7.4949e-01 (8.93e-03)	7.5834e-01 (9.56e-04)	7.5932e-01 (3.88e-05)	7.6094e+01 (4.72e-03)	**7.4734e-01 (5.84e-04)**	**Very Significant**
20	8.5746e-01 (4.83e-03)	8.3982e-01 (3.98e-03)	8.3453e-01 (6.53e-04)	8.4283e-01 (3.44e-02)	**8.0635e-01 (2.7343e-03)**	**Extremely Significant**

According to [35] the above problem has no polynomial time solution. The objective function for $n = 2$ is shown in Figure 5.

In Table 7, we show the mean and the standard deviation (within parentheses) of the best-of-run values for 30 independent runs of each of the six algorithms over the two most difficult instances of the radar poly-phase code design problem (for dimensions $n = 19$ and $n = 20$). Figure 6 graphically presents the rate of convergence of the contestant algorithms for the 20 - dimensional radar code design problem.

Table 7 and Figure 6 clearly indicates that DHS can achieve more accurate and statistically better results as compared to all the contestants consuming less amount of computational time for this difficult real life optimization problem.

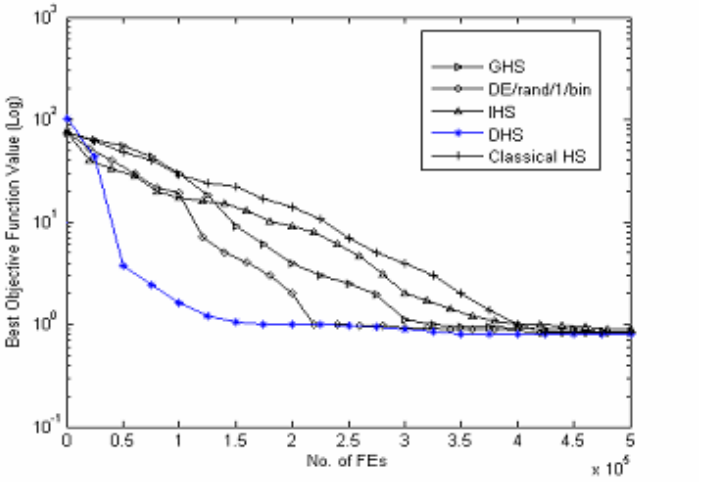


Figure 6. Progress to the optimum solution for spread spectrum radar poly-phase code problem (n = 20).

6. Conclusions

This paper has presented an improved variant of the classical Harmony Search metaheuristic by blending with it a difference vector-based mutation operator borrowed from the DE family of algorithms. A theoretical analysis indicates that the population-variance of HS can be enhanced by the incorporation of the differential mutation operator. This intensifies the explorative power of HS and the algorithm, modified in this way, is able to beat the state-of-the-art variants of HS over popular unconstrained and constrained benchmark functions in a statistically meaningful way. The Differential HS (DHS) was found to outperform the classical HS and its two variants namely GHS and HIS in terms of final accuracy, convergence speed and robustness. Checking the effect of variation of the scale factor F of the differential mutation operator may be a worthy issue for future investigations. Other modifications to the algorithm leading to better trade-offs between explorative and exploitative tendencies should also be investigated both empirically and theoretically. Future works will also focus on studying the applications of DHS on engineering optimization problems.

A. Estimation of moments of a uniformly distributed random number

Let, R be a particular instantiation of a continuously uniformly distributed random number, lying between 0 and 1. Probability distribution of such a random number may be given as shown in the figure below:

Here $\psi(z) = \text{Continuous Uniform Probability Distribution Function}$. In this case,$p = 0$, $q = 1$ and $z = R$.

Therefore, R and R^2 may be estimated as shown below

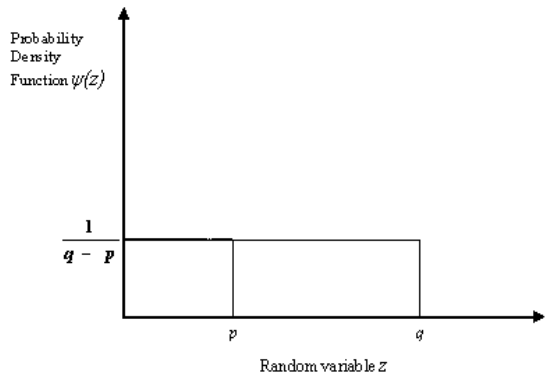


Figure 7. Graphical plot of continuous uniform probability distribution

$$\begin{aligned} E(R) &= \int_0^1 Rf(R)\mathrm{d}R = \int_0^1 R\mathrm{d}R = \left[\frac{R^2}{2}\right]_0^1 = \frac{1}{2} \\ E(R^2) &= \int_0^1 R^2 f(R)\mathrm{d}R = \int_0^1 R^2\mathrm{d}R = \frac{1}{3} \end{aligned}$$

B. Calculation of $E(x_r)$ and $E(x_r^2)$

Let, x_r be a member of the one-dimensional population, initialized randomly in the search space. Thus in order to find $E(x_r)$ and $E(x_r^2)$, x_r may be modeled as follows:

$$x_r = -a + 2Ra$$

where, R is a particular instantiation of a continuously uniformly distributed random number, lying between 0 and 1, while and represents the upper and lower bound of the allowable initialization range.

$$\therefore x_r^2 = a^2(4R^2 - 4R + 1)$$

Using statistical estimates of the uniformly distributed random number 'R' , as proved in Appendix A, we can estimate those of x_r as follows:

$$\begin{aligned} E(x_r) &= -a + 2aE(R) = -a + 2a \cdot \frac{1}{2} \\ E(x_r^2) &= a^2\left\{4E(R^2) - 4E(R) + 1\right\} = a^2\left\{4 \cdot \frac{1}{3} - 4 \cdot \frac{1/2}{+}1\right\} = \frac{a^2}{3} \end{aligned}$$

C. Calculation of $E[x_{\beta_1} x_{\beta_2}]$

$$\begin{aligned} E[x_{\beta_1} x_{\beta_2}] &= \sum_{\substack{\beta_1 \\ \beta_1 \neq \beta_2}} \sum_{\beta_2} x_{\beta_1} x_{\beta_2} P(x_{\beta_1}, x_{\beta_2}) \\ &= \frac{1}{m(m-1)} \sum_{\substack{\beta_1 \\ \beta_1 \neq \beta_2}} \sum_{\beta_2} x_{\beta_1} x_{\beta_2} \\ &= \frac{1}{m(m-1)} \left\{ \left(\sum x_{\beta_1} \right)^2 - \sum x_{\beta_1}^2 \right\} \\ &= \frac{1}{m(m-1)} \left\{ (m\bar{x})^2 - m\overline{x^2} \right\} \end{aligned}$$

References

[1] T. Bäck, D. Fogel, Z. Michalewicz, *Handbook of Evolutionary Computation*, Oxford Univ. Press, 1997.

[2] A.E. Eiben and J.E. Smith, *Introduction to Evolutionary Computing*, Springer, 2003.

[3] D. Ashlock, *Evolutionary Computation for Modeling and Optimization*, Springer, 2006.

[4] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial System,* Oxford University Press, New York, 1999.

[5] J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm Intelligence*, Morgan Kaufmann, San Francisco, CA, 2001.

[6] A. P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*, John Wiley & Sons, 2006.

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

[8] K.S. Lee and Z.W. Geem, "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice", *Computer Methods in Applied Mechanics and Engineering*, *Eng*, 194, 3902–3933, 2004.

[9] Z. W. Geem (Ed.), *Music-Inspired Harmony Search Algorithm: Theory and Applications*, Studies in Computational Intelligence, Springer, 2009.

[10] Z.W. Geem, J.H. Kim, and G.V. Loganathan, "Harmony search optimization: application to pipe network design", *Int. J. Model. Simul,* 22,(2), 125–133, 2002.

[11] K. S. Lee and Z.W. Geem, "A new structural optimization method based on the harmony search algorithm", *Computers and Structures*, 82, pp. 781–798, Elsevier, 2004.

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

[13] A. Vasebi, M. Fesanghary, and S. M. T. Bathaeea, "Combined heat and power economic dispatch by harmony search algorithm", *International Journal of Electrical Power and Energy Systems*, Vol. 29, No. 10, 713-719, Elsevier, 2007.

[14] Z. W. Geem, "Optimal scheduling of multiple dam system using harmony search algorithm", *Lecture Notes in Computer Science*, Vol. 4507, 316-323, Springer, 2007.

[15] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems", *Applied Mathematics and Computation*, Vol. 188, 1567–1579, Elsevier Science, 2007.

[16] M. G. H. Omran and M. Mahdavi, "Global-best harmony search", *Applied Mathematics and Computation*, Vol.198, 643–656, Elsevier Science, 2008.

[17] M. Fesanghary, M. Mahdavi, M. Minary-Jolandan, Y. Alizadeh, "Hybridizing sequential quadratic programming with HS algorithm for engineering optimization", *Computer Methods in Applied Mechanics and Engineering*, Volume 197, Issues 33- 40, 1, pp. 3080-3091, Elsevier, June 2008.

[18] P.T. Boggs and J.W. Tolle, "Sequential quadratic programming", *Acta Numerica*, 4, 1–52, 1995.

[19] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies: a comprehensive introduction", *Natural Computing*, 1(1):3-52, 2002.

[20] R. Storn and K. V. Price, "Differential evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces", Technical Report TR-95-012, ICSI, `http://http.icsi.berkeley.edu/~storn/litera.html`, 1995.

[21] ____, "Differential Evolution – a simple and efficient heuristic for global optimization over continuous spaces", *Journal of Global Optimization*, 11(4) 341–359, 1997.

[22] R. Storn, K. V. Price, and J. Lampinen,*Differential Evolution - A Practical Approach to Global Optimization*, Springer, Berlin, 2005.

[23] D. Zaharie, "Critical values for the control parameters of differential evolution algorithms", in R. Matousek, P. Osmera (eds.), Proc. of Mendel 2002, 8-th International Conference on Soft Computing, Brno,Czech Republic, June 2002, pp. 62-67.

[24] R. Parncutt, *Harmony: A Psychoacoustical Approach*, Springer Verlag, 1989.

[25] A.E. Eiben and C.A. Schippers, "On evolutionary exploration and exploitation", *Fundamenta Informaticae*, 35, 1-16, IOS Press, 1998.

[26] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, 3(2), 82-102, July 1999.

[27] P. J. Angeline, "Evolutionary optimization versus particle swarm optimization: Philosophy and the performance difference," Lecture Notes in Computer Science (vol. 1447), *Proc. of* 7^{th} *International Conference on. Evolutionary Programming – Evolutionary Programming* VII, pp. 84-89, 1998

[28] A. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol.3, no. 2, pp. 124-141, 1999.

[29] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 284–294, Sep. 2000.

[30] S. A. Kazarlis, S. E. Papadakis, J. B. Theochairs, and V. Petridis, "Microgenetic algorithms as generalized hill-climbing operators for GA optimization," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 3, pp. 204–217, Jun. 2001.

[31] S. Koziel and Z. Michalewicz, "Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 1, pp. 19–44, 1999.

[32] Z. Michalewicz and G. Nazhiyath, "Genocop III: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints," in *Proceedings of the 2nd IEEE Conference on Evolutionary Computation*. Piscataway, NJ: IEEE Press, vol. 2, pp. 647–651, 1995.

[33] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *Evolutionary Computation*, vol. 4, no. 1, pp. 1–32, 1996.

[34] L. Jiao, Y. Li, M. Gong, and X. Zhang, "Quantum-inspired immune clonal algorithm for global numerical optimization", *IEEE Transactions on System, Man, and Cybernetics*, Part B, Vol. 38, No.5, 1234-1253, 2008.

[35] N. Mladenović, J. Petrovic, V. Kovacevic-Vujicic, and M. Cangalovic, "Solving spread-spectrum radar polyphase code design problem by tabu search and variable neighborhood search," *European Journal of Operational Research*, 153, 389-399, 2003.