Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 750819, 12 pages
http://dx.doi.org/10.1155/2013/750819

*Research Article*

# Harmony Search Based Parameter Ensemble Adaptation for Differential Evolution

**Rammohan Mallipeddi**

*School of Electronics Engineering, Kyungpook National University, Taegu 702 701, Republic of Korea*

Correspondence should be addressed to Rammohan Mallipeddi; mallipeddi.ram@gmail.com

Received 28 June 2013; Accepted 8 July 2013

Academic Editor: Zong Woo Geem



In differential evolution (DE) algorithm, depending on the characteristics of the problem at hand and the available computational resources, different strategies combined with a different set of parameters may be effective. In addition, a single, well-tuned combination of strategies and parameters may not guarantee optimal performance because different strategies combined with different parameter settings can be appropriate during different stages of the evolution. Therefore, various adaptive/self-adaptive techniques have been proposed to adapt the DE strategies and parameters during the course of evolution. In this paper, we propose a new parameter adaptation technique for DE based on ensemble approach and harmony search algorithm (HS). In the proposed method, an ensemble of parameters is randomly sampled which form the initial harmony memory. The parameter ensemble evolves during the course of the optimization process by HS algorithm. Each parameter combination in the harmony memory is evaluated by testing them on the DE population. The performance of the proposed adaptation method is evaluated using two recently proposed strategies (DE/current-to-*p*best/bin and DE/current-to-gr_best/bin) as basic DE frameworks. Numerical results demonstrate the effectiveness of the proposed adaptation technique compared to the state-of-the-art DE based algorithms on a set of challenging test problems (CEC 2005).

## 1. Introduction

During the last decade, evolutionary algorithms (EAs) inspired by Darwinian theory of evolution are becoming increasingly popular because of their ability to handle nonlinear and complex optimization problems. Unlike, the conventional numerical optimization methods, EAs are population-based metaheuristic algorithms and require the objective function values, while properties such as differentiability and continuity are not necessary. However, EAs performance depends on the encoding schemes, evolutionary operators, and parameter settings such as population size, mutation scale factor, and crossover rate. In addition, an appropriate parameter selection is a problem dependent and requires a time-consuming trial-and-error parameter tuning process. The trail-and-error based parameter selection is ineffective if the optimization is required in an automated environment or if the user has no experience in the fine art of the control parameter tuning. To overcome this, different parameter adaptation schemes have been presented [1–6]. Among the different parameter adaptation techniques adaptive and self-adaptive techniques are popular due to their ability to adjust the parameter during the course of the evolution with minimal or no intervention from the user. In other words, in adaptive and self-adaptive techniques, the parameter adaptation is done based on the feedback from the search process. Self-adaptive techniques are based on the assumption that the most appropriate parameter values produce better offspring which are more likely to survive and propagate the better parameter values [7]. Therefore, in self-adaptive methods the parameters are directly encoded into the individuals and are evolved together with the encoded solutions.

Differential evolution (DE) [8] is a fast and simple technique that has been successfully applied in diverse fields [9–12]. Like most EAs, the performance of DE [13] is sensitive to population size ($NP$), mutation and crossover strategies, and their associated control parameters such as scale factor ($F$) and crossover rate ($CR$). In other words, the best combination

of strategies and their control parameter settings can be different for different optimization problems. In addition, for a given optimization the best combination of strategies and parameter values differ based on the available computational resources and required accuracy. Therefore, to successfully solve a specific optimization problem, it is generally necessary to perform a time-consuming trial-and-error search for the most appropriate strategies and their associated parameter values. However, in DE during the evolution process, the population traverses through different regions in the search space, within which different strategies [14] with different parameter settings may be more effective than a well-tuned, single combination of strategies and parameters. In DE literature, different partial adaptation schemes have been proposed [7, 14–17] to overcome the time-consuming trial-and-error procedure.

In [18], the authors proposed an adaptive DE algorithm referred to as JADE. In JADE, the authors implemented a new mutation strategy "DE/current-to-*p*best/1" and the control parameters ($F$ and $CR$) are self-adapted. "DE/current-to-*p*best/1" is a generalized version of "DE/current-to-best/1." JADE uses the conventional binomial crossover strategy. In JADE, the self-adaptation of the control parameters avoids the requirement of prior knowledge about parameter settings and works well without user interaction. Motivated by JADE, the authors in [19] proposed another adaptive DE algorithm referred to as MDE-*p*BX (Modified DE with *p*-best crossover). MDE-*p*BX uses a new mutation strategy "DE/current-to-gr_best/1" which is a modified version of "DE/current-to-best/1." Unlike JADE, MDE-*p*BX uses a more exploitative "*p*-best binomial crossover" strategy. In [20], the authors proposed an ensemble approach for parameter adaptation of DE, where each parameter has a pool of values competing to produce future offspring based on their success in the past generations.

Harmony search (HS) is also population-based metaheuristic optimization algorithm which mimics the music improvisation process. Recently, HS is gaining significance as an efficient optimization algorithm and is used in variety of applications. In HS, the generation of a new vector or solution is based on the consideration of all the existing vectors, rather than considering only two vectors as in DE (parent and mutant vector) [21]. This characteristic of HS makes it more explorative compared to the DE algorithm.

During the past decade, hybridization of EAs has gained significance, due to ability to complement each other's strengths and overcome the drawbacks of the individual algorithms. To enhance the exploitation ability in HS [22], memory consideration is generally employed where new individuals are generated based on the historical search experience. In addition, HS employs random selection approach to explore and sample new solutions from the search space. In HS, the random selection aids the exploration ability but not as efficient as the DE mutation strategy and results in slow convergence characteristics. However, a new solution formed using a set of few randomly selected individuals may limit the exploration ability in DE when the population diversity is low [23]. In [23], the authors propose a hybrid algorithm referred to as differential harmony search (DHS) by fusing the HS and DE mechanisms. The hybridized DHS algorithm could reasonably balance the exploration and exploitation abilities.

In this paper, we propose a DE parameter adaptation technique based on HS algorithm. In the proposed adaptation method, a group of DE control parameter combinations are randomly initialized. The randomly initialized DE parameter combinations form the initial harmony memory (HM) of the HS algorithm. Each combination of the parameters present in the HM is evaluated by testing on the DE population during the evolution. Based on the effectiveness of the DE parameter combinations present in HM, the HS algorithm evolves the parameter combinations. At any given point of time during the evolution of the DE population, the HM contains an ensemble of DE parameters that suits the evolution process of the DE population.

The rest of the paper is organized as follows. Section 2 provides a brief literature review on different adaptive DE and HS algorithms. Section 3 presents the proposed algorithm where the DE parameters are adapted using a HS algorithm. Section 4 presents the experimental results while Section 5 presents the conclusions with some future directions.

## 2. Literature Review

*2.1. Classical Differential Evolution.* Differential evolution (DE) is a simple real parameter optimization algorithm that belongs to the class of evolutionary algorithms (EAs) and involves the continuous application of operators such as mutation, crossover, and selection. DE starts with $NP$, $D$-dimensional parameter vectors, referred to as population, where each individual is a candidate solution to the problem at hand as shown in

$$\mathbf{X}_{i,G} = \left\{x^1_{i,G}, \ldots, x^D_{i,G}\right\}, \quad i = 1, \ldots, NP. \tag{1}$$

The initial population is uniformly sampled from the search space constrained by the prescribed minimum and maximum parameter bounds $\mathbf{X}_{\min} = \{x^1_{\min}, \ldots, x^D_{\min}\}$ and $\mathbf{X}_{\max} = \{x^1_{\max}, \ldots, x^D_{\max}\}$.

After initialization, corresponding to each target vector $\mathbf{X}_{i,G}$ in the population at the generation $G$, a mutant vector $\mathbf{V}_{i,G} = \{v^1_{i,G}, v^2_{i,G}, \ldots, v^D_{i,G}\}$ can be generated via a mutation strategy. The most commonly used mutation strategy in DE is given by:

"DE/rand/1" [24]:

$$\mathbf{V}_{i,G} = \mathbf{X}_{r^i_1,G} + F \cdot \left(\mathbf{X}_{r^i_2,G} - \mathbf{X}_{r^i_3,G}\right). \tag{2}$$

The indices $r^i_1, r^i_2, r^i_3$ are randomly generated mutually exclusive integers in the range $[1, NP]$. The indices are randomly generated once for each mutant vector and are also different from the index $i$. The scale factor $F$ is a positive control parameter for scaling the difference vector.

After the mutation, crossover operation is applied to each pair of the target vector $\mathbf{X}_{i,G}$ and its corresponding mutant vector $\mathbf{V}_{i,G}$ to generate a trial vector: $\mathbf{U}_{i,G}$ =

$\{u_{i,G}^1, u_{i,G}^2, \ldots, u_{i,G}^D\}$. In the basic version, DE employs the binomial (uniform) crossover defined as follows [25]:

$$u_{i,G}^j = \begin{cases} v_{i,G}^j & \text{if } \left(\text{rand}_j\ [0,1] \le CR\right) \\ & \text{or } \quad \left(j = j_{\text{rand}}\right) \quad j = 1, 2, \ldots, D \\ x_{i,G}^j & \text{otherwise.} \end{cases} \tag{3}$$

In (3), the crossover rate $CR$ is a user-specified constant within the range [0,1], which controls the fraction of parameter values copied from the mutant vector. $j_{\text{rand}}$ is a randomly chosen integer in the range $[1, D]$. In DE, there exists another type of crossover operator called exponential crossover which is functionally equivalent to the circular two-point crossover operator [14].

After crossover, the generated trial vectors are evaluated using the objective function and a selection operation is performed as shown in

$$\mathbf{X}_{i,G+1} = \begin{cases} \mathbf{U}_{i,G}, & \text{if } f\left(\mathbf{U}_{i,G}\right) \le f\left(\mathbf{X}_{i,G}\right), \\ \mathbf{X}_{i,G}, & \text{otherwise.} \end{cases} \tag{4}$$

In (4), $f(\mathbf{U}_{i,G})$ and $f(\mathbf{X}_{i,G})$ correspond to the objective values of the trial and target vectors.

As mentioned above, the mutation, crossover, and selection steps are repeated generation after generation until a termination criterion (reaching the maximum number of function evaluations set) is satisfied. The algorithmic description of the DE is summarized in Algorithm 1.

*2.2. Parameter Adaptation in Differential Evolution.* Although, DE has attracted much attention recently as a global optimizer over continuous spaces [25], the performance of the conventional DE algorithm depends on the chosen mutation and crossover strategies and the associated control parameters. Depending on the complexity of the problem, the performance of DE becomes more sensitive to the strategies and the associated parameter values [26] and inappropriate choice may lead to premature convergence, stagnation, or wastage of computational resources [16, 26–29]. In other words, due to the complex interaction of control parameters with the DE's performance [7], choosing an appropriate mutation and crossover strategies and control parameters require some expertise. Since DE was proposed, various empirical guidelines were suggested for choosing the population size ($NP$) [24–26, 29, 30], mutation and crossover strategies [12, 24–26, 28, 31, 32], and their associated control parameter settings: scale factor ($F$) [24–26, 29, 30, 33, 34] and crossover rate ($CR$) [24–26, 28–30, 35, 36].

To some extent, the guidelines are useful for selecting the individual parameters of DE. However, the performance of DE is more sensitive to the combination of the mutation strategy and its associated parameters. For a mutation strategy, [7] a particular value of $CR$ makes the parameter $F$ sensitive while some other values of $CR$ make the same $F$ robust. Hence, the manual parameter tuning of DE is not easy and requires a good expertise. To overcome the burden of tuning the DE parameters by trial-and-error, various adaptive techniques have been proposed [14–16, 37–39]. The most popular adaptive DE variants are as follows [40].

(1) SaDE [14]: in SaDE, the trail vector generation strategies and the associated control parameter values are self-adapted based on their previous experiences of generating promising solutions.

(2) jDE [7]: the control parameters $F$ and $CR$ are encoded into the individuals and are adjusted based on the parameters $\tau_1$ and $\tau_2$. The initial values of $F$ and $CR$ of each population individual of DE were selected as 0.5 and 0.9, respectively. Then, based on a random number (rand) which is uniformly generated in the range of $[0, 1]$, the values of $F$ and $CR$ are reinitialized if rand $< \tau_1$ and rand $< \tau_2$, respectively. $F$ and $CR$ are reinitialized to a new value randomly generated in the ranges [0.1, 1.0] and [0, 1], respectively.

(3) JADE [18]: JADE employs a new mutation strategy "DE/current-to-$p$best" and updates the control parameters in an adaptive manner. "DE/current-to-$p$best" is a generalized version of "DE/current-to-best" and helps in diversifying the population and improves the convergence performance. In JADE, the parameter adaptation is done automatically and does not need any prior knowledge regarding relationship between the parameter settings and the characteristics of optimization problems. In JADE, the $F$ and $CR$ values corresponding to each population member are sampled from the mean values of $F$ and $CR$. The mean values of $F$ and $CR$ are updated by the individual $F$ and $CR$ values which are successful in generating better trail vectors compared to the target vectors.

(4) EPSDE [20]: while solving a specific problem, different mutation strategies with different parameter settings may be better during different stages of the evolution than a single mutation strategy with unique parameter settings as in the conventional DE. Motivated by these observations an ensemble of mutation strategies and parameter values for DE (EPSDE) was proposed in which a pool of mutation strategies, along with a pool of values corresponding to each associated parameter competes to produce successful offspring population. In EPSDE, the candidate pool of mutation strategies and parameters should be restrictive to avoid the unfavorable influences of less effective mutation strategies and parameters [14].

(5) MDE-$p$BX [19]: motivated by JADE, MDE-$p$BX employs a new mutation strategy "DE/current-to-gr_best/1" and the control parameters are self-adapted. According to the new mutation strategy, the algorithm uses the best individual of a group (whose size is $q$% of the population size) of randomly selected solutions from current generation to perturb the parent (target) vector. In addition, unlike JADE, MDE-$p$BX uses a modified binomial crossover operation referred to as "$p$-best crossover." According to the modified crossover operation, a biased parent selection scheme has been incorporated by letting

*STEP 1*: Randomly initialize a population of *NP*, *D*-dimensional parameter vectors. Set the generation number $G = 0$.
*STEP 2*: **WHILE** stopping criterion is not satisfied
**DO**
**Mutation**—Equation (2)
**Crossover**—Equation (3)
**Selection**—Equation (4)
*Increment the generation count* $G = G + 1$
*STEP 3*: **END WHILE**

Algorithm 1: Standard differential evolution algorithm.

*STEP 1*: Initialize the HM with HMS randomly generated solutions. Set generation count $G = 0$.
*STEP 2*: **WHILE** stopping criterion is not satisfied
/* *Generate a new solution** /
**FOR** each decision variable **DO**
**IF** $rand_1$ < HMCR
Pick the value from one of the solutions in HM
**IF** $rand_2$ < PAR
Perturb the value picked /* *New solution generated** /
**END IF**
**END IF**
**END FOR**
**IF** new solution better than the worst solution in HM (in terms of fitness)
Replace the worst solution in HM with new solution
**END IF**
*Increment the generation count* $G = G + 1$
*STEP 3*: **END WHILE**

Algorithm 2: Standard harmony search algorithm.

each mutant undergo the usual binomial crossover with one of the $p$ top-ranked individuals from the current population and not with the target vector with the same index as used in all variants of DE.

*2.3. Harmony Search Algorithm.* Unlike most EAs, which simulate natural selection and biological evolution, HS is a population-based metaheuristic optimization algorithm which mimics the music improvisation process where musicians improvise their instruments' pitch by searching for a perfect state of harmony. Some of the characteristics of HS that distinguish it from other metaheuristics such as DE are as follows [21]: (1) considering all the existing solution vectors while generating a new vector, rather than considering only two vectors as in DE (target vector and trail vector); and (2) independent consideration for each decision variable in a solution vector. An overview of the standard HS algorithm is presented in Algorithm 2.

In HS the improvisation operators, memory consideration, pitch adjustment, and random consideration play a major role in achieving the desired balance between the exploitation and exploration during the optimization process [41]. Essentially, both pitch adjustment and random consideration are the key components of achieving the desired diversification in HS. In random consideration, the new vector's components are generated at random and have the same level of efficiency as in other algorithms that handle randomization. The random consideration of HS allows the exploration of new regions that may not have been visited in the search space. In HS, pitch adjustment enhances diversification by tuning the components of a new vector's within a given bandwidth by adding/subtracting a small random amount to an existing component stored in HM. Further to that, pitch adjustment operator can also be considered as a mechanism to support the intensification of HS through controlling the probability of PAR. The intensification in the HS algorithm is represented by the third HS operator, memory consideration. A high harmony acceptance rate means that good solutions from the history/memory are more likely to be selected or inherited. This is equivalent to a certain degree of elitism. Obviously, if the acceptance rate is too low, solutions will converge more slowly.

Recently, HS algorithm garnered a lot of attention from the research community and is successfully applied in solving many optimization problems in engineering and computer science. Consequently, the interest in HS has led to the improvement and development of its performance in line with the requirements of problems that are solved. The improvements proposed by different researchers related to HS can be categorized as follows [42]: (1) HS improvement

*STEP 1*: Initialize a population of *NP*, *D*-dimensional individuals as the population of DE
*STEP 2*: Initialize HM with HMS randomly selected individuals.
*STEP 3*: **WHILE** stopping criterion is not satisfied
**DO**
**Generate a new vector based on HS**
**FOR** 1: HMS + 1 /* Evaluate each parameter combination of the HM */
**Mutation**
**Crossover**
**Selection**
**Evaluate the objective value of each HM vector**
***Increment the generation count*** $G = G + 1$
**END FOR**
**Update HM**
*STEP 4*: **END WHILE**

Algorithm 3: Harmony search based parameter ensemble adaptation for DE (HSPEADE).

by appropriate parameters setting; and (2) improvement of HS by hybridizing with other metaheuristic algorithms.

## 3. Harmony Search Based Parameter Ensemble Adaptation for DE (HSPEADE)

As highlighted in the previous section, depending on the nature of problem (unimodal or multimodal) and available computation resources, different optimization problems require different mutation and crossover strategies combined with different parameter values to obtain optimal performance. In addition, to solve a specific problem, different mutation and crossover strategies with different parameter settings may be better during different stages of the evolution than a single set of strategies with unique parameter settings as in the conventional DE. Motivated by these observations, the authors in [20] proposed an ensemble approach (EPSDE) in which a pool of mutation and crossover strategies, along with a pool of values corresponding to each associated parameter competes to produce successful offspring population.

In EPSDE, each member in the DE population is randomly assigned a mutation and crossover strategies and the associated parameter values taken from the respective pools. The population members (target vectors) produce offspring (trial vectors) using the assigned strategies and parameter values. If the generated trial vector is able to enter the next generation of the evolution, then combination of the strategies and the parameter values that produced trail vector are stored. If trial vector fails to enter the next generation, then the strategies and parameter values associated with that target vector are randomly reinitialized from the respective pools or from the successful combinations stored with equal probability.

To have an optimal performance based on the ensemble approach, the candidate pool of strategies and parameters should be restrictive to avoid the unfavorable influences of less effective strategies and parameters [14]. In other words, the strategies and the parameters present in the respective pools should have diverse characteristics, so that they can exhibit distinct performance characteristics during different stages of the evolution, when dealing with a particular problem.

In EPSDE, since the strategy and parameter pools are restrictive, most of the individuals in the pools may become obsolete during the course of the evolution of DE population. Therefore, it would be apt if the strategy and the parameter pools can evolve with the DE population. Based on this motivation, we propose an HS based parameter ensemble adaptation for DE (HSPEADE). The overall view of the proposed HSPEADE is presented in Algorithm 3.

As shown in Algorithm 3, after the initialization of DE population, the HM of the HS algorithm is initialized with HMS number of randomly generated vectors. The members of the HM are the parameter combinations ($F$ and $CR$ values) corresponding to the mutation and crossover strategies used. Using the members in the HM, a new parameter combination vector is generated using the HS algorithm described in Algorithm 2. Each of the HMS + 1 parameter combinations is evaluated by testing them on the DE population during the evolution. After evaluating all the members of the HM and the newly generated parameter combination, the HM is updated as in HS algorithm. The generation of new parameter combination and the updating of the HM are performed throughout the evolution process of the DE algorithm.

To obtain optimal performance based on the ensemble approach, it is obvious that the parameter combinations in HM should be diverse during initial generations of the DE population evolution and should converge to the optimal combination towards the end of the evolution. During the course of the experimentation, we observed that HS is more suitable to evolve the parameter ensemble due to its characteristics such as the following: (1) HS generates a single vector every generation and replaces the worst performing vector; (2) it can randomly generate new solution vectors thus enabling diversity if needed and (3) it considers all the solution vectors in the memory to generate a new solution.

In HSPEADE, to facilitate the diversity in parameter ensemble in the initial stages and to allow the HS to converge

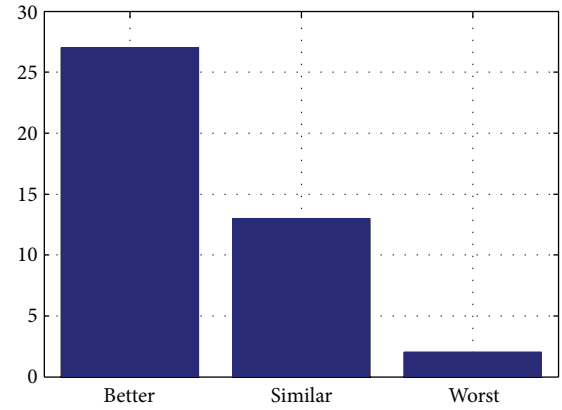


Figure 1: Performance comparison of JADE and HSPEADE1.

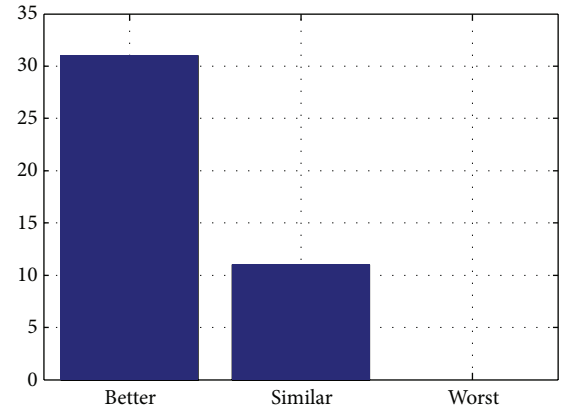


Figure 2: Performance comparison of EPDE1 and HSPEADE1.

to the optimal combination, we made some modifications in the HS algorithm. In HS algorithm shown in Algorithm 2, the parameters HMCR and PAR are deterministically changed. HMCR is linearly increased from 0 to 1 while PAR is decreased linearly from 1 to 0 with the increase in the generation count.

## 4. Experimental Setup and Results

In this section, we evaluate the performance of the proposed parameter adaptation technique for DE. The details regarding the test problems, experimental environment, and algorithms used for comparison are given below.

*4.1. Problem Set.* The performance of the proposed method is evaluated using a selected set of standard test functions from the special session on real-parameter optimization of the IEEE Congress on Evolutionary Computation (CEC 2005) [43]. In this work, we use the first 14 functions of CEC 2005 out of which functions 1–5 are unimodal, functions 6–12 are multimodal, and functions 13-14 are hybrid composition functions. The details about the problems such as parameter ranges, location of the optimal solution, and the optimal objective values can be found in [43]. In the present work, to

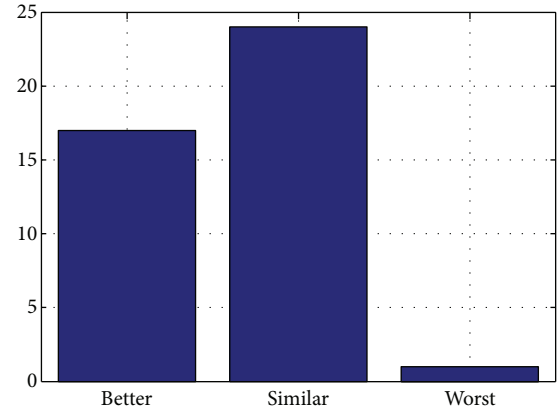


Figure 3: Performance comparison of MDE and HSPEADE2.

evaluate the scalability of the algorithms 30-, 50-, and 100-dimensional versions of the test problems are considered. The number of function evaluation used for 30-, 50-, and 100-dimensional problems are 100000, 500000, and 1000000, respectively. On each of the test problems, every algorithm under consideration is run 30 times.

*4.2. Setup for Algorithmic Comparison.* The proposed HSPEADE being a general idea can be applied with any frame work. In this work, the experiments are designed as follows.

(1) We consider a single crossover strategy which is binomial crossover. We selected binomial crossover because the two recent adaptive DE algorithms (JADE [18] and MDE-$p$BX [19]) which show significant performance on the problem set considered employ binomial crossover. It is to be noted that MDE-$p$BX uses a modified "$p$-best binomial crossover" operator. However, in this work we consider the classical binomial crossover only.

(2) We consider two mutation strategies "DE/current-to-$p$best" and "DE/current-to-gr_best"

The algorithmic comparison is divided into two sets as follows. SET 1 uses the "DE/current-to-$p$best" strategy while SET 2 uses the "DE/current-to-gr_best" strategy. EPSDE algorithm mentioned above is referred to as EPDE below because in the current work the strategies are fixed. MDE-$p$BX algorithm is referred to as MDE below because in the present work we use a simple binomial crossover instead of the greedy "$p$-best crossover."

*SET 1*:

JADE: "DE/current-to-$p$best" strategy, binomial crossover strategy, $c = 0.1$, $p = 0.05$, $F$ and $CR$ are adapted [18].

EPDE1: "DE/current-to-$p$best" strategy, binomial crossover, $c = 0.1$, $p = 0.05$, $F = \{0.5, 0.7, 0.9\}$, $CR = \{0.1, 0.5, 0.9\}$.

Table 1: Performance of JADE, EPDE1, and HSEPDE1 on 30D problems.

| | JADE | | EPDE1 | | HSPEADE1 | |
|---|---|---|---|---|---|---|
| | Mean | Std. | Mean | Std. | Mean | Std. |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | *8.59E − 28* | *4.19E − 28* | 7.89E − 10 | 7.89E − 10 | 5.83E − 14 | 1.84E − 13 |
| 3 | **1.31E + 04** | **5.75E + 04** | 2.47E + 05 | 9.21E + 04 | 3.09E + 04 | 1.27E + 04 |
| 4 | 2.45E − 02 | 8.40E − 02 | *8.16E − 05* | *1.70E − 04* | **6.71E − 06** | **6.39E − 06** |
| 5 | 7.53E + 02 | 3.68E + 02 | *1.29E − 02* | *2.81E − 02* | **3.19E − 04** | **3.46E − 04** |
| 6 | 1.03E + 01 | 2.72E + 01 | *1.26E + 00* | *1.88E + 00* | **5.69E − 15** | **5.20E − 15** |
| 7 | 1.56E − 02 | 1.51E − 02 | 1.53E − 02 | 1.19E − 02 | **0** | **0** |
| 8 | 2.08E + 01 | 2.46E − 01 | 2.09E + 01 | 2.97E − 02 | 2.09E + 01 | 2.83E − 02 |
| 9 | **0** | **0** | 8.61E + 00 | 2.04E + 00 | **0** | **0** |
| 10 | *2.73E + 01* | *5.70E + 00* | 1.26E + 02 | 1.03E + 01 | 2.51E + 01 | 5.41E + 00 |
| 11 | *2.68E + 01* | *2.03E + 00* | 3.36E + 01 | 1.29E + 00 | 2.69E + 01 | 1.56E + 00 |
| 12 | *4.82E + 03* | *3.97E + 03* | 3.52E + 04 | 3.59E + 04 | 6.03E + 03 | 6.01E + 03 |
| 13 | *1.67E + 00* | *3.04E − 02* | 5.36E + 00 | 2.90E − 01 | **1.48E + 00** | **9.91E − 02** |
| 14 | *1.24E + 01* | *3.27E − 01* | 1.31E + 01 | 9.57E − 02 | **1.28E + 01** | **2.47E − 01** |

Table 2: Performance of JADE, EPDE1, and HSEPDE1 on 50D problems.

| | JADE | | EPDE1 | | HSPEADE1 | |
|---|---|---|---|---|---|---|
| | Mean | Std. | Mean | Std. | Mean | Std. |
| 1 | 6.347E − 15 | 2.32E − 15 | *0* | *0* | **0** | **0** |
| 2 | 5.63E − 04 | 7.82E − 06 | *2.47E − 05* | *2.16E − 05* | **3.49E − 08** | **6.85E − 08** |
| 3 | **8.75E + 04** | **2.96E + 04** | 7.84E + 05 | 2.33E + 05 | 1.34E + 05 | 4.20E + 04 |
| 4 | 1.66E + 03 | 5.13E − 01 | 2.16E + 02 | 3.54E + 02 | **6.16E + 00** | **3.77E + 00** |
| 5 | 3.16E + 03 | 5.29E + 02 | 2.20E + 03 | 6.49E + 02 | **1.19E + 03** | **4.51E + 02** |
| 6 | *1.54E + 01* | *1.06E + 01* | 3.26E + 01 | 2.89E + 01 | **6.97E − 09** | **2.16E − 08** |
| 7 | 9.83E − 03 | 1.38E − 02 | 8.85E − 03 | 1.18E + 02 | **0** | **0** |
| 8 | 2.11E + 01 | 3.25E − 02 | 2.11E + 01 | 3.34E − 02 | 2.11E + 01 | 5.22E − 02 |
| 9 | **0** | **0** | 5.35E + 01 | 6.45E + 00 | **0** | **0** |
| 10 | *1.94E + 02* | *2.06E + 01* | 2.74E + 02 | 1.20E + 01 | **6.67E + 01** | **2.16E + 01** |
| 11 | *6.21E + 01* | *1.74E + 00* | 6.47E + 01 | 1.49E + 00 | **5.33E + 01** | **2.26E + 00** |
| 12 | 1.77E + 05 | 7.11E + 04 | **1.71E + 04** | **1.28E + 04** | **1.73E + 04** | **9.14E + 03** |
| 13 | 2.31E + 01 | 4.78E − 01 | *1.33E + 01* | *6.41E − 01* | **2.66E + 00** | **7.21E − 02** |
| 14 | 2.28E + 01 | 2.55E − 01 | 2.29E + 01 | 1.79E − 01 | **2.23E + 01** | **4.09E − 01** |

HSPEADE1: "DE/current-to-*p*best" strategy, binomial crossover, $F$, $CR$ and $p$ are encoded into the HS algorithm for adaptation. $F$ ranges from 0.2 to 1.2, $CR$ ranges from 0 to 1, and $p$ ranges from 0.05 to 2.50.

*SET 2*:

MDE: "DE/current-to-gr_best" strategy, binomial crossover strategy, $q = 0.15$, $F$ and $CR$ are adapted [19].

EPDE2: "DE/current-to-gr_best" strategy, binomial crossover strategy, $q = 0.15$, $F = \{0.5, 0.7, 0.9\}$, $CR = \{0.1, 0.5, 0.9\}$.

HEPEADE2: "DE/current-to-gr_best" strategy, binomial crossover, $F$, $CR$, and $q$ are encoded in to the HS algorithm for adaptation. $F$ ranges from 0.2 to 1.2, $CR$ ranges from 0 to 1, and $p$ ranges from 0.05 to 2.50.

*4.3. Statistical Tests.* To compare the performance of different algorithms, we employ two types of statistical tests, namely, $t$-test and Wilcoxon rank sum test. The $t$-test being a parametric method can be used to compare the performance of two algorithms on a single problem. When the performances of two algorithms are to be compared on multiple problems $t$-test is not valid as the normality assumption fails [44]. Therefore, to compare the performance of two algorithms over a set of different problems, we can use a nonparametric test such as the Wilcoxon rank sum test [44].

*4.4. Experimental Results.* The experimental results (mean and standard deviations) corresponding to algorithms JADE, EPDE1, and HSEPDE1 (SET 1) on 30-, 50-, and 100-dimensional problems are presented in Tables 1, 2, and 3, respectively. The experimental results (mean and standard deviations) corresponding to algorithms MDE, EPDE2, and

Table 3: Performance of JADE, EPDE1, and HSEPDE1 on 100D problems.

| | JADE | | EPDE1 | | HSPEADE1 | |
|---|---|---|---|---|---|---|
| | Mean | Std. | Mean | Std. | Mean | Std. |
| 1 | 6.48$E-10$ | 4.02$E-09$ | *1.57E − 28* | *2.49E − 28* | **5.05E − 30** | **1.59E − 29** |
| 2 | 3.49$E+01$ | 6.47$E+01$ | *5.61E + 00* | *1.77E + 01* | **5.99E − 01** | **3.82E − 01** |
| 3 | 3.01$E+06$ | 6.21$E+04$ | *1.53E + 06* | *601E + 05* | **1.02E + 06** | **3.39E + 05** |
| 4 | 5.03$E+04$ | 6.98$E-03$ | 2.03$E+04$ | 8.35$E+03$ | **1.03E + 04** | **2.49E + 03** |
| 5 | 7.53$E+05$ | 3.95$E+03$ | *5.89E + 03* | *1.33E + 03* | **3.39E + 03** | **7.82E + 02** |
| 6 | 6.35$E+03$ | 7.37$E+01$ | *1.85E + 02* | *4.13E + 01* | **5.49E − 01** | **1.66E + 00** |
| 7 | 8.13$E-03$ | 5.69$E-03$ | 4.19$E-03$ | 6.77$E-03$ | **1.23E − 03** | **3.89E − 03** |
| 8 | 2.19$E+01$ | 9.47$E+01$ | 2.13$E+01$ | 1.49$E-02$ | 2.13$E+01$ | 2.61$E-02$ |
| 9 | *3.55E − 12* | *1.12E − 12* | 2.64$E+02$ | 1.19$E+01$ | 1.06$E-15$ | 1.91$E-15$ |
| 10 | *5.94E + 02* | *8.88E + 01* | 7.13$E+02$ | 1.53$E+01$ | **2.39E + 02** | **8.14E + 01** |
| 11 | *1.28E + 02* | *3.57E + 00* | 1.51$E+02$ | 3.16$E+00$ | **1.27E + 02** | **4.82E + 00** |
| 12 | 6.77$E+05$ | 2.11$E+04$ | *1.02E + 05* | *6.39E + 04* | **9.39E + 04** | **2.96E + 04** |
| 13 | *6.96E + 00* | *7.09E − 01* | 4.15$E+01$ | 1.65$E+00$ | **6.17E + 00** | **6.72E − 01** |
| 14 | *4.58E + 01* | *3.98E − 01* | 4.74$E+01$ | 1.85$E-01$ | 4.67$E+01$ | 3.35$E-01$ |

Table 4: Performance of MDE, EPDE2, and HSEPDE2 on 30D problems.

| | MDE | | EPDE2 | | HSPEADE2 | |
|---|---|---|---|---|---|---|
| | Mean | Std. | Mean | Std. | Mean | Std. |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 2.19$E-12$ | 5.17$E-12$ | 6.29$E-10$ | 6.59$E-10$ | **2.33E − 18** | **5.67E − 18** |
| 3 | **2.45E + 04** | **1.31E + 04** | 3.30$E+05$ | 1.52$E+05$ | 2.83$E+04$ | 1.69$E+04$ |
| 4 | 5.96$E-05$ | 1.33$E-04$ | *7.30E − 06* | *9.85E − 06* | **3.88E − 06** | **6.79E − 06** |
| 5 | 6.48$E+02$ | 3.81$E+02$ | *1.04E − 01* | *2.72E − 01* | **1.93E − 04** | **1.56E − 04** |
| 6 | 2.49$E+01$ | 2.67$E+01$ | *8.02E − 01* | *1.68E + 00* | **2.02E − 15** | **2.17E − 15** |
| 7 | 1.79$E-02$ | 1.29$E-02$ | *9.59E − 03* | *9.58E − 03* | **0** | **0** |
| 8 | 2.08$E+01$ | 2.75$E-01$ | 2.09$E+01$ | 5.49$E-02$ | 2.09$E+01$ | 5.56$E-02$ |
| 9 | **0** | **0** | 8.62$E+00$ | 1.96$E+00$ | **0** | **0** |
| 10 | *2.72E + 01* | *5.27E + 00* | 1.19$E+02$ | 1.01$E+01$ | **2.31E + 01** | **7.51E + 00** |
| 11 | *2.72E + 01* | *1.68E + 00* | 3.25$E+01$ | 1.84$E+00$ | **2.55E + 01** | **3.50E + 00** |
| 12 | **2.81E + 03** | **2.01E + 03** | 3.97$E+04$ | 2.42$E+04$ | 4.19$E+03$ | 4.61$E+03$ |
| 13 | **1.44E + 00** | **1.02E − 01** | 5.31$E+00$ | 2.79$E-01$ | 1.45$E+00$ | 1.15$E-01$ |
| 14 | **1.22E + 01** | **5.39E − 01** | 1.31$E+01$ | 2.47$E-01$ | 1.28$E+01$ | 1.52$E-01$ |

HSEPDE2 (SET 2) on 30-, 50-, and 100-dimensional problems are presented in Tables 4, 5, and 6, respectively. In Tables 1–6, the mean and standard deviation (std.) values are reported for every algorithm on each test problem.

The $t$-test and Wilcoxon rank sum test results comparing the performance of algorithms in SET 1 and SET 2 are presented in Tables 7 and 8, respectively. In Tables 7 and 8, the $t$-test results comparing two algorithms on each problem are presented and the last row presents the Wilcoxon rank sum test results. For each of the two tests, +1, 0, −1 in A versus B comparison indicates B is better than A, B is equal to A, and B is worse than A, respectively. For example, in JADE versus EPDE1 comparison in Table 7 (30D) EPSDE1 is better, equal and worst on test problems 4, 7, and 9, respectively.

### 4.5. Analysis of Experimental Results

*SET 1.* In Tables 1, 2, and 3, for each test problem, the best performing algorithms among the JADE and EPDE1 are highlighted using italic while the overall best among JADE, EPDE1, and HSPEADE1 are highlighted using bold.

From the experimental results, it can be observed that JADE performs better than EPDE1 on 30-dimensional versions of the problems. However, as the dimensionality of the test problems increases, the performance of the EPDE1 becomes better compared to JADE algorithm. The improved performance of EPDE1 can be attributed to the ensemble approach as different combinations of strategies and parameters can be effective during different stages of the evolution process [20].

Table 5: Performance of MDE, EPDE2, and HSEPDE2 on 50D problems.

| | MDE | | EPDE2 | | HSPEADE2 | |
|---|---|---|---|---|---|---|
| | Mean | Std. | Mean | Std. | Mean | Std. |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | *4.19E − 07* | *4.17E − 07* | 2.30*E* − 05 | 1.72*E* − 05 | **1.08E − 08** | **3.36E − 08** |
| 3 | 2.77*E* + 05 | 7.29*E* + 04 | **1.19E + 05** | **4.68E + 04** | 6.29*E* + 05 | 1.81*E* + 05 |
| 4 | 2.03*E* + 02 | 2.19*E* + 02 | *1.30E + 02* | *1.31E + 02* | **3.56E + 00** | **2.87E + 00** |
| 5 | 4.27*E* + 03 | 5.51*E* + 02 | *2.23E + 03* | *5.09E + 02* | **8.73E + 02** | **6.51E + 02** |
| 6 | 9.78*E* + 01 | 3.82*E* + 01 | *7.77E + 00* | *6.97E + 00* | **3.99E − 01** | **1.26E + 00** |
| 7 | *2.22E − 03* | *4.71E − 03* | 1.10*E* − 02 | 1.65*E* − 02 | 3.45*E* − 03 | 5.59*E* − 03 |
| 8 | 2.11*E* + 01 | 2.98*E* − 02 | 2.12*E* + 01 | 2.77*E* − 02 | 2.11*E* + 01 | 4.06*E* − 02 |
| 9 | *5.33E − 16* | *1.19E − 15* | 5.41*E* + 01 | 3.67*E* + 00 | **0** | **0** |
| 10 | **6.34E + 01** | **6.99E + 00** | 2.78*E* + 02 | 1.14*E* + 01 | 7.54*E* + 01 | 1.77*E* + 01 |
| 11 | *5.37E + 01* | *3.07E + 00* | 6.44*E* + 01 | 2.01*E* + 00 | **5.15E + 01** | **6.29E + 00** |
| 12 | 2.14*E* + 04 | 1.92*E* + 04 | *1.49E + 04* | *1.23E + 04* | **1.46E + 04** | **1.37E + 04** |
| 13 | *2.87E + 00* | *3.64E − 01* | 1.29*E* + 01 | 8.07*E* − 01 | **2.84E + 00** | **2.79E − 01** |
| 14 | *2.18E + 01* | *3.04E − 01* | 2.28*E* + 01 | 3.29*E* − 01 | 2.23*E* + 01 | 2.79*E* − 01 |

Table 6: Performance of MDE, EPDE2, and HSEPDE2 on 100D problems.

| | MDE | | EPDE2 | | HSPEADE2 | |
|---|---|---|---|---|---|---|
| | Mean | Std. | Mean | Std. | Mean | Std. |
| 1 | *5.05E − 30* | *1.59E − 29* | 2.68*E* − 28 | 2.13*E* − 28 | **6.31E − 30** | **1.60E − 29** |
| 2 | *3.20E − 01* | *4.12E − 01* | 1.07*E* + 00 | 6.92*E* − 01 | **1.19E − 02** | **1.30E − 02** |
| 3 | 3.16*E* + 06 | 4.85*E* + 05 | *1.31E + 06* | *3.51E + 05* | **1.22E + 06** | **3.91E + 05** |
| 4 | *1.78E + 04* | *7.20E + 03* | 1.93*E* + 04 | 7.89*E* + 03 | **8.88E + 03** | **2.70E + 03** |
| 5 | 1.38*E* + 04 | 1.50*E* + 03 | *6.02E + 03* | *1.03E + 03* | **2.89E + 03** | **6.91E + 02** |
| 6 | 2.04*E* + 02 | 4.41*E* + 01 | *1.39E + 02* | *3.69E + 01* | **1.17E + 00** | **2.14E + 00** |
| 7 | **8.04E − 04** | **2.34E − 03** | 6.65*E* − 03 | 6.57*E* − 03 | 5.17*E* − 03 | 7.01*E* − 03 |
| 8 | 2.13*E* + 01 | 2.21*E* − 02 | 2.13*E* + 01 | 1.99*E* − 02 | 2.13*E* + 01 | 2.19*E* − 02 |
| 9 | *1.94E − 14* | *7.83E − 15* | 2.61*E* + 02 | 1.64*E* + 01 | **0** | **0** |
| 10 | *2.69E + 02* | *4.02E + 01* | 6.99*E* + 02 | 2.80*E* + 01 | **2.17E + 02** | **5.19E + 01** |
| 11 | *1.33E + 02* | *3.85E + 00* | 1.49*E* + 02 | 2.97*E* + 00 | **1.25E + 02** | **4.06E + 00** |
| 12 | 8.35*E* + 04 | 4.31*E* + 04 | 8.47*E* + 04 | 3.92*E* + 04 | **6.63E + 04** | **5.09E + 04** |
| 13 | *8.99E + 00* | *1.42E + 00* | 4.20*E* + 01 | 1.24*E* + 00 | **6.32E + 00** | **1.31E + 00** |
| 14 | **4.55E + 01** | **5.08E + 01** | 4.74*E* + 01 | 1.78*E* − 01 | 4.63*E* + 01 | 3.53*E* − 01 |

From Tables 1–3, it can be observed that HSPEADE1 outperforms JADE and EPDE1 in most of the test problems. On 30-dimensional problems HSPEADE1 is better than or equal to JADE in 11 cases while JADE is better than HSPEADE1 in 3 cases. As the dimensionality of the test problems increases, the performance of HSPEADE1 gets better than JADE.

From the results, it is clear that the performance of HSPEADE1 is always better than or equal to EPDE1. This confirms our assumption that evolving parameter ensemble is better than fixed combination of parameter ensemble.

*SET 2*. In Tables 4, 5, and 6, for each test problem, the best performing algorithms among the MDE and EPDE2 are highlighted using italic while the overall best among MDE, EPDE2, and HSPEADE2 are highlighted using bold.

From the experimental results in Tables 4–6 a similar observation to the above can be made. In 30-, 50-, and 100-dimensional problems, the performance of MDE and EPDE2 is distributed. Unlike EPDE1 which dominates JADE as the dimensionality increases, EPDE2 is unable to dominate MDE. This may be due to the explorative ability of "DE/current-to-gr_best" strategy employed in MDE. However, as the dimensionality of the test problems increases the performance of HSPEADE2 becomes better compared to MDE.

From the Wilcoxon rank sum test results (bottom row of Tables 7 and 8), it can be observed that HSPEADE (HSPEADE1 and HSPEADE2) is always better than the algorithms under comparison. In both the experimental setups (SET 1 and SET 2), the statistical $t$-test results present in Tables 7 and 8 are summarized in Figures 1 to 4 to present a better view. For example in Figure 1, the bar plots indicate the number of test problems (30D, 50D, and 100D) on which HSPEADE1 is better, similar, and worst compared

Table 7: Statistical test results to compare the performance of JADE, EPDE1, and HSEPDE1.

| | JADE versus EPDE1 | | | JADE versus HSPEADE1 | | | EPDE1 versus HSPEADE1 | | |
|---|---|---|---|---|---|---|---|---|---|
| | 30D | 50D | 100D | 30D | 50D | 100D | 30D | 50D | 100D |
| 1 | 0 | +1 | +1 | 0 | +1 | +1 | 0 | 0 | 0 |
| 2 | −1 | +1 | +1 | 0 | +1 | +1 | +1 | +1 | +1 |
| 3 | −1 | −1 | +1 | −1 | −1 | +1 | +1 | +1 | +1 |
| 4 | +1 | 0 | 0 | +1 | +1 | +1 | 0 | +1 | +1 |
| 5 | +1 | 0 | +1 | +1 | +1 | +1 | 0 | +1 | +1 |
| 6 | +1 | −1 | +1 | +1 | +1 | +1 | +1 | +1 | +1 |
| 7 | 0 | 0 | 0 | +1 | +1 | +1 | +1 | +1 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | −1 | −1 | −1 | 0 | 0 | 0 | +1 | +1 | +1 |
| 10 | −1 | −1 | −1 | 0 | +1 | +1 | +1 | +1 | +1 |
| 11 | −1 | −1 | −1 | 0 | +1 | 0 | +1 | +1 | +1 |
| 12 | −1 | +1 | +1 | 0 | +1 | +1 | +1 | 0 | +1 |
| 13 | −1 | +1 | −1 | +1 | +1 | +1 | +1 | +1 | +1 |
| 14 | −1 | 0 | −1 | +1 | +1 | 0 | +1 | +1 | 0 |
| Wilcoxon test | −1 | −1 | +1 | +1 | +1 | +1 | +1 | +1 | +1 |

Table 8: Statistical test results to compare the performance of MDE, EPDE2, and HSEPDE2.

| | MDE versus EPDE2 | | | MDE versus HSPEADE2 | | | EPDE2 versus HSPEADE2 | | |
|---|---|---|---|---|---|---|---|---|---|
| | 30D | 50D | 100D | 30D | 50D | 100D | 30D | 50D | 100D |
| 1 | 0 | 0 | −1 | 0 | 0 | 0 | 0 | 0 | +1 |
| 2 | 0 | −1 | −1 | +1 | +1 | +1 | +1 | +1 | +1 |
| 3 | −1 | +1 | +1 | 0 | 0 | +1 | +1 | −1 | 0 |
| 4 | +1 | 0 | 0 | 0 | +1 | +1 | 0 | +1 | +1 |
| 5 | +1 | +1 | +1 | +1 | +1 | +1 | +1 | +1 | +1 |
| 6 | +1 | +1 | +1 | +1 | +1 | +1 | +1 | +1 | +1 |
| 7 | +1 | 0 | −1 | +1 | 0 | 0 | +1 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | −1 | −1 | −1 | 0 | 0 | +1 | +1 | +1 | +1 |
| 10 | −1 | −1 | −1 | 0 | 0 | +1 | +1 | +1 | +1 |
| 11 | −1 | −1 | −1 | 0 | 0 | +1 | +1 | +1 | +1 |
| 12 | −1 | 0 | 0 | 0 | 0 | 0 | +1 | 0 | 0 |
| 13 | −1 | −1 | −1 | 0 | 0 | +1 | +1 | +1 | +1 |
| 14 | −1 | −1 | −1 | 0 | 0 | −1 | +1 | +1 | +1 |
| Wilcoxon test | 0 | 0 | 0 | +1 | +1 | +1 | +1 | +1 | +1 |

to JADE. From Figures 1, 2, 3, and 4, it is clear that the performance of HSPEADE1 is better than JADE and EPDE1 while HSPEADE2 is better than MDE and EPDE2.

## 5. Conclusions and Future Work

To improve the performance of DE, different adaptation techniques have been proposed. In this paper, we propose a new parameter adaptation technique for DE based on ensemble approach and HS algorithm and is referred to as HSPEADE. In HSPEADE, an ensemble of parameters is randomly sampled and forms the initial harmony memory. The parameter ensemble evolves during the course of the optimization process by HS algorithm. Each parameter combination in the harmony memory is evaluated by testing them on the DE population. During the initial stages of the evolution the DE parameter combinations in the harmony memory of HS are diverse and facilitate exploration for the better parameter combination. However, during the end of the evolution process fine tuning of the parameter combinations occurs and facilitates exploitation.

The performance of HSPEADE is evaluated by using two recently proposed DE strategies (DE/current-to-*p*best/bin and DE/current-to-gr_best/bin) and the numerical results show that the proposed adaptation technique significant improvement compared to the state-of-the-art adaptive DE

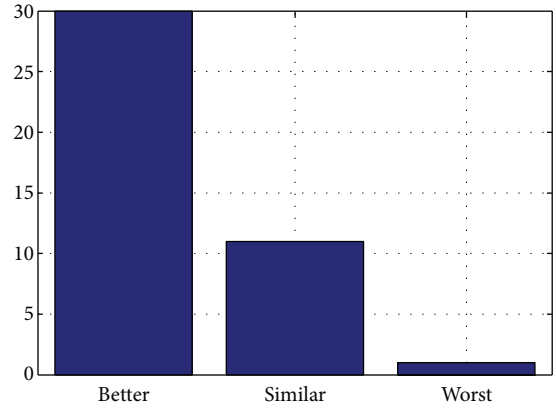


Figure 4: Performance comparison of EPDE2 and HSPEADE2.

algorithms. From the experimental results, it can be observed that the proposed adaptation technique can handle the scalability issues better compared to the other adaptive techniques.

In the present work, we only consider the evolution of the parameter ensemble using the HS framework. As a future work, we would like to incorporate the ensemble of mutation and crossover strategies into the HS framework.

## Acknowledgment

This research was supported by Kyungpook National University Research Fund, 2012.

## References

[1] P. J. Angeline, M. Palaniswami, Y. Attikiouzel, R. J. Marks, D. Fogel, and T. Fukuda, "Adaptive and self-adaptive evolutionary computation," *Computational Intelligence*, pp. 152–161, 1995.

[2] Á. 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.

[3] J. Gomez, D. Dasgupta, and F. Gonazalez, "Using adaptive operators in genetic search," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'03)*, Chicago, Ill, USA, 2003.

[4] B. R. Julstrom, "What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm," in *Proceedings of the 6th International Conference on Genetic Algorithms*, Pittsburgh, Pa, USA, 1995.

[5] J. E. Smith and T. C. Fogarty, "Operator and parameter adaptation in genetic algorithms," *Soft Computing*, vol. 1, pp. 81–87, 1997.

[6] A. Tuson and P. Ross, "Adapting operator settings in genetic algorithms," *Evolutionary Computation*, vol. 6, no. 2, pp. 161–184, 1998.

[7] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[8] R. Storn and K. Price, "Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces," Tech. Rep. TR-95-012, ICSI, http://ftp.icsi.berkeley.edu/ftp/pub/techreports/1995/tr-95-012.pdf.

[9] R. Joshi and A. C. Sanderson, "Minimal representation multisensor fusion using differential evolution," *IEEE Transactions on Systems, Man, and Cybernetics Part A*, vol. 29, no. 1, pp. 63–76, 1999.

[10] T. Rogalsky, R. W. Derksen, and S. Kocabiyik, "Differential evolution in aerodynamic optimization," in *Proceedings of 46th Annual Conference of Canadian Aeronautics and Space Institute*, pp. 29–36, Montreal, Quebec, 1999.

[11] M. K. Venu, R. Mallipeddi, and P. N. Suganthan, "Fiber Bragg grating sensor array interrogation using differential evolution," *Optoelectronics and Advanced Materials, Rapid Communications*, vol. 2, no. 11, pp. 682–685, 2008.

[12] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 526–553, 2009.

[13] J. Liu and J. Lampinen, "On setting the control parameter of the differential evolution method," in *Proceedings of 8th International Conference on Soft Computing (MENDEL '02)*, pp. 11–18, 2002.

[14] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[15] M. G. H. Omran, A. Salman, and A. P. Engelbrecht, "Self-adaptive differential evolution," in *Proceedings of the Computational Intelligence and Security (LNCS '05)*, pp. 192–199, 2005.

[16] D. Zaharie, "Control of population diversity and adaptation in differential evolution algorithms," in *Proceedings of the 9th International Conference on Soft Computing*, Brno, Czech Republic, 2003.

[17] J. Tvrdík, "Adaptation in differential evolution: a numerical comparison," *Applied Soft Computing Journal*, vol. 9, no. 3, pp. 1149–1155, 2009.

[18] Z. Jingqiao and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, pp. 945–958, 2009.

[19] S. M. Islam, S. Das, S. Ghosh, S. Roy, and P. N. Suganthan, "An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 42, no. 2, pp. 482–500, 2012.

[20] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 1679–1696, 2011.

[21] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[22] M. G. H. Omran and M. Mahdavi, "Global-best harmony search," *Applied Mathematics and Computation*, vol. 198, no. 2, pp. 643–656, 2008.

[23] L.-P. Li and L. Wang, "Hybrid algorithms based on harmony search and differential evolution for global optimization," in *Proceedings of the 1st ACM/SIGEVO Summit on Genetic and Evolutionary Computation, (GEC '09)*, pp. 271–278, Shanghai, China, June 2009.

[24] R. Storn, "On the usage of differential evolution for function optimization," in *Proceedings of the Biennial Conference of the*

*North American Fuzzy Information Processing Society (NAFIPS '96)*, Berkeley, Calif, USA, June 1996.

[25] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[26] R. Gämperle, S. D. Müller, and P. Koumoutsakos, "A parameter study for differential evolution," in *Proceedings of the Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, Interlaken, Switzerland, 2002.

[27] S. Das, A. Konar, and U. K. Chakraborty, "Two improved differential evolution schemes for faster global search," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*, pp. 991–998, June 2005.

[28] J. Lampinen and I. Zelinka, "On stagnation of the differential evolution algorithm," in *Proceedings of 6th International Mendel Conference on Soft Computing (MENDEL '00)*, pp. 76–83, 2000.

[29] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution : A Practical Approach to Global Optimization*, Natural Computing Series, Springer, Berlin, Germany, 2005.

[30] J. Rönkkönen, S. Kukkonen, and K. V. Price, "Real-parameter optimization with differential evolution," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'05)*, pp. 506–513, Edinburgh, Scotland, September 2005.

[31] K. V. Price, "An introduction to differential evolution," in *New Ideas in Optimization*, D. Corne, M. Dorgio, and F. Glover, Eds., pp. 79–108, McGraw-Hill, London, UK, 1999.

[32] A. Iorio and X. Li, "Solving rotated multi-objective optimization problems using differential evolution," in *Proceedings of the Australian Conference on Artificial Intelligence*, Cairns, Australia, 2004.

[33] D. Zaharie, "Critical values for the control parameters of differential evolution," in *Proceedings of 18th International Conference on Soft Computing (MENDEL '02)*, pp. 62–67, Brno, Czech Republic, 2002.

[34] J. Rönkkönen and J. Lampinen, "On using normally distributed mutation step length for the differential evolution algorithm," in *Proceedings of 19th International MENDEL Confernce on Soft Computing (MENDEL '03)*, pp. 11–18, Brno, Czech Republic, 2003.

[35] D. Zaharie, "Influence of crossover on the behavior of differential evolution algorithms," *Applied Soft Computing Journal*, vol. 9, no. 3, pp. 1126–1138, 2009.

[36] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, "Modified differential evolution for constrained optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pp. 25–32, July 2006.

[37] H. A. Abbass, "The self-adaptive pareto differential evolution algorithm," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '02)*, vol. 1, pp. 831–836, 2002.

[38] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing*, vol. 9, no. 6, pp. 448–462, 2005.

[39] D. Zaharie and D. Petcu, "Adaptive Pareto differential evolution and its parallelization," in *Proceedings of 5th International Conference on Parallel Processing and Applied Mathematics*, pp. 261–268, Czestochowa, Poland, 2003.

[40] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[41] X.-S. Yang, "Harmony search as a metaheuristic algorithm," in *Music-Inspired Harmony Search Algorithm*, Z. Geem, Ed., vol. 191, pp. 1–14, Springer, Berlin, Germany, 2009.

[42] O. M. Alia and R. Mandava, "The variants of the harmony search algorithm: an overview," *Artificial Intelligence Review*, vol. 36, no. 1, pp. 49–68, 2011.

[43] P. N. Suganthan, N. N. Hansen, J. J. Liang et al., *Problem Definitions and Evaluation Criteria For the CEC, 2005 Special Session on Real-Parameter Optimization*, Nanyang Technological University; Singapore and KanGAL; IIT, Kanpur, India, 2005.

[44] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization," *Journal of Heuristics*, vol. 15, no. 6, pp. 617–644, 2009.