Accelerating Differential Evolution Using an Adaptive Local Search

Nasimul Noman and Hitoshi Iba, *Member, IEEE*

***Abstract*—We propose a crossover-based adaptive local search (LS) operation for enhancing the performance of standard differential evolution (DE) algorithm. Incorporating LS heuristics is often very useful in designing an effective evolutionary algorithm for global optimization. However, determining a single LS length that can serve for a wide range of problems is a critical issue. We present a LS technique to solve this problem by adaptively adjusting the length of the search, using a hill-climbing heuristic. The emphasis of this paper is to demonstrate how this LS scheme can improve the performance of DE. Experimenting with a wide range of benchmark functions, we show that the proposed new version of DE, with the adaptive LS, performs better, or at least comparably, to classic DE algorithm. Performance comparisons with other LS heuristics and with some other well-known evolutionary algorithms from literature are also presented.**

***Index Terms*—Differential evolution (DE), global optimization, local search (LS), memetic algorithm (MA).**

I. Introduction

OVER THE PAST few years, the field of global optimization has been very active, producing different kinds of deterministic and stochastic algorithms for optimization in the continuous domain. Among the stochastic approaches, evolutionary computation (EC) offers a number of exclusive advantages: robust and reliable performance, global search capability, little or no information requirement, etc. [1]. These characteristics of EC, as well as other supplementary benefits such as ease of implementation, parallelism, no requirement for a differentiable or continuous objective function, etc., make it an attractive choice. Consequently, there have been many studies related to real-parameter optimization using EC, resulting in many variants such as *evolutionary strategies* (ES) [2], *real coded genetic algorithms* (RCGAs) [3], [4], *differential evolution* (DE) [5], *particle swarm optimization* (PSO) [6], etc.

Several studies have shown that incorporating some form of domain knowledge can greatly improve the search capability of evolutionary algorithms (EAs) [7]–[11]. Many problem dependent heuristics, such as approximation algorithm, local search (LS) techniques, specialized recombination operators, etc., have been tried in many different ways to accomplish this task. In particular, the hybridization of EAs with local searches has proven to be very promising [12], [13]. Cultural algorithms are another class of computational approaches that are related to EAs and make use of domain knowledge and LS activity [14], [15].

EAs embedded with a neighborhood search procedure are commonly known as Memetic algorithms (MAs) [9], [16]. MAs are population-based heuristic search approaches, that apply a separate LS process to refine individuals, i.e., to improve their fitness [8]. The rationale behind MAs is to provide an effective and efficient global optimization method by compensating for deficiency of EA in local exploitation and inadequacy of LS in global exploration. According to Lozano *et al.*, real coded MAs (RCMAs) have evolved mainly in two classes depending on the type of LS employed [17].

1) *Local improvement process (LIP) oriented LS (LLS)*: The first category refines the solutions of each generation by applying efficient LIPs, like gradient descent or hill-climbers. LIPs can be applied to every member of the population or with some specific probability and with various replacement strategies.
2) *Crossover-based LS (XLS)*: This group employs crossover operators for local refinement. A crossover operator is a recombination operator that produces offspring around the parents. For this reason, it may be considered as a move operator in an LS strategy [17]. This is particularly attractive for real coding because there are some real-parameter crossover operators that can generate offspring adaptively (i.e., according to the distribution of parents) without any additional adaptive parameter [18].

Adaptation of parameters and operators has become a very promising research field in MAs. Ong and Keane proposed meta-Lamarckian learning in MAs that adaptively chooses among multiple memes during a MA search [19]. They proposed two adaptive strategies, MA-S1 and MA-S2, in their work and empirical studies showed their superiority over other traditional MAs. An excellent taxonomy and comparative study on adaptive choice of memes in MAs is presented in [20]. In order to balance between local and genetic search, Bambha *et al.* proposed simulated heating that systematically integrates parameterized LS (both statically and dynamically) into EAs [21]. In the context of combinatorial problems, Krasnagor and Smith showed that self-adaptive hybridization between GA and LS/diversification process gives rise to a better global search metaheuristics [22]. Because of the superior performance of adaptive MAs, in this paper, we investigate a new XLS with adaptive capability for an EA, namely, DE.

DE is one of the most recent EAs for solving real-parameter optimization problems. Like other EAs, DE is a population-based, stochastic global optimizer capable of working reliably in nonlinear and multimodal environments [5]. Using a few parameters, DE exhibits an overall excellent performance for a

Manuscript received June 11, 2006; revised October 18, 2006.

N. Noman is with the IBA Laboratory, Graduate School of Frontier Sciences, University of Tokyo, Tokyo, Japan and also with the Department of Computer Science and Engineering, University of Dhaka, Dhaka, Bangladesh (e-mail: noman@iba.k.u-tokyo.ac.jp).

H. Iba is with the Department of Frontier Informatics, Graduate School of Frontier Sciences, University of Tokyo, Tokyo, Japan (e-mail: iba@iba.k.u-tokyo.ac.jp).

Digital Object Identifier 10.1109/TEVC.2007.895272

wide range of benchmark functions. Due to its simple but powerful search capability, it has got many real-world applications: pattern recognition, digital filter design, neural network training, etc. [23]. The advantages of DE, such as a simple and easy-to-understand concept, compact structure, ease of use, high convergence characteristics, and robustness, make it a high-class technique for real-valued parameter optimization.

Although DE was designed using the common concepts of EAs, such as multipoint searching, use of recombination and selection operators, it has some unique characteristics that make it different from many others in the family. The major differences are in the way offspring are generated and in the selection mechanism that DE applies to transit from one generation to the next. DE uses a one-to-one spawning and selection relationship between each individual and its offspring. Although these features are the strength of the algorithm, they can sometimes turn into weaknesses, especially if the global optimum should be located using a limited number of fitness evaluations. Because, by breeding an offspring for each individual, DE sometimes explores too many search points before locating the global optimum. In addition, though DE is particularly simple to work with, having only a few control parameters, choice of these parameters is often critical for the performance of DE [24]. Again, choosing the best among different learning strategies available for DE is often not easy for a particular problem [25]. Therefore, several researchers are now paying attention to the improvement of the classic DE algorithm using different heuristics [24]–[27].

For real-world applications, the fitness evaluation is usually the most expensive part of the search process; therefore, an EA should be able to locate the global optimum with the fewest possible number of fitness evaluations. Although DE belongs to the elite EA class in consideration of its convergence velocity, its overall performance does not meet the requirements for all classes of problems. In accordance with the earlier discussion, hybridization with a LS operation can accelerate DE by improving its neighborhood exploitation capability. We have already made a preliminary study on the use of LS operation for improving the performance of DE, particularly for high-dimensional optimization problems [28]. In this work, we present a more generalized and efficient LS process in the spirit of Lamarckian learning for accelerating classic DE.

The adaptive nature of the newly proposed LS scheme exploits the neighborhoods more effectively, and thus significantly improves the convergence characteristics of the original algorithm. The performance improvement is shown using a set of benchmark functions with different properties. The paper also presents a performance comparison with some well known MAs. The paper is organized as follows.

The next section of this paper contains a brief overview of DE. The third section presents some contemporary research on DE. In Section IV, the proposed new version of the DE algorithm, with adaptive LS, is presented in detail. Section V reports the experimental results comparing the proposed version of DE and the classic DE algorithm. Comparisons between the proposed adaptive LS strategy and other LS strategies, and between the newly proposed DE algorithm and other MAs are also presented in Section V. Section VI discusses the results focusing on the proposed DE characteristics. Finally, Section VII concludes this paper.

DE

1. Generate an Initial Population P^G
2. ***Evaluate*** P^G
3. For each individual I in P^G
4. ***Reproduce*** an offspring J from I
5. $P^{G+1} = P^{G+1} \cup$ ***Select*** (I, J)
6. Set G = G+1
7. Repeat Step 3 to 6 until termination criteria is met

Fig. 1. Generation alternation model of DE.

II. Differential Evolution

Like other EAs, DE is a population-based stochastic optimizer that starts to explore the search space by sampling at multiple, randomly chosen initial points [23], [29]. Thereafter, the algorithm guides the population towards the vicinity of the global optimum through repeated cycles of reproduction and selection. The generation alternation model used in "classic DE" for refining candidate solutions in successive generations is shown in Fig. 1.

The different components of the DE algorithm are summarized as follows.

Parent Choice: As shown in the DE model, each individual in the current generation is allowed to breed through mating with other randomly selected individuals from the population. Specifically, for each individual $x_i^G, i = 1, \ldots, P$, where G denotes the current generation, three other random individuals x_j^G, x_k^G and x_l^G are selected from the population such that j, k and $l \in \{1, \ldots, P\}$ and $i \neq j \neq k \neq l$. This way, a parent pool of four individuals is formed to breed an offspring.

Reproduction: After choosing the parents, DE applies a *differential mutation* operation to generate a mutated individual v_i^G, according to the following equation:

$$v_i^G = x_j^G + F\left(x_k^G - x_l^G\right) \tag{1}$$

where F, commonly known as *scaling factor* or *amplification factor*, is a positive real number, typically less than 1.0 that controls the rate at which the population evolves. To complement the differential mutation search strategy, DE then uses a crossover operation, often referred to as *discrete recombination*, in which the mutated individual v_i^G is mated with x_i^G and generates the *offspring* or *trial individual* u_i^G. The genes of u_i^G are inherited from x_i^G and v_i^G, determined by a parameter called *crossover probability* ($C_r \in [0, 1]$), as follows:

$$u_{i,t}^G = \begin{cases} v_{i,t}^G, & \text{if } r(t) \leq C_r \text{ or } t = rn(i) \\ x_{i,t}^G, & \text{if } r(t) > C_r \text{ and } t \neq rn(i) \end{cases} \tag{2}$$

where $t(= 1, \ldots, N)$ denotes the tth element of individual vectors. $r(t) \in [0, 1]$ is the tth evaluation of a uniform random number generator and $rn(i) \in \{1, \ldots, N\}$ is a randomly chosen index which ensures that u_i^G gets at least one element from v_i^G. From the above description, another difference between DE and GA becomes clear; that is in DE mutation is applied before crossover, which is the opposite of GA. Moreover, in GA, mutation is applied occasionally to maintain

diversity in the population, whereas in DE, mutation is a regular operation applied to generate each offspring.

Selection: DE applies selection pressure only when picking survivors. A *knockout* competition is played between each individual x_i^G and its offspring u_i^G and the winner is selected deterministically based on objective function values and promoted to the next generation.

Many variants of the classic DE have been proposed, which use different learning strategies and/or recombination operations in the reproduction stage [5], [23]. In order to distinguish among its variants, the notation $DE/a/b/c$ is used, where "a" specifies the vector to be mutated (which can be random or the best vector); "b" is the number of difference vectors used; and "c" denotes the crossover scheme, *binomial* or *exponential*. The binomial crossover scheme is represented in (2) and in case of exponential crossover, the *crossover probability* C_r regulates how many consecutive genes of the mutated individual v_i^G, on average, are copied to the *trial individual* u_i^G. Using this notation, the DE strategy described above can be denoted as *DE/rand/1/bin*. Other well-known variants are *DE/best/1/bin*, *DE/rand/2/bin*, and *DE/best/2/bin* which can be implemented by simply replacing (1) by (3)–(5), respectively. Again, each of the above algorithms can be configured to use the exponential crossover

$$v_i^G = x_{\text{best}}^G + F\left(x_j^G - x_k^G\right) \tag{3}$$

$$v_i^G = x_j^G + F\left(x_k^G - x_l^G\right) + F\left(x_m^G - x_n^G\right) \tag{4}$$

$$v_i^G = x_{\text{best}}^G + F\left(x_j^G - x_k^G\right) + F\left(x_l^G - x_m^G\right) \tag{5}$$

where x_{best}^G represents the best individual in the current generation, m and $n \in \{1, \ldots, N\}$, and $i \neq j \neq k \neq l \neq m \neq n$. A recent study that empirically compares some of the variants of DE is presented in [30].

III. Related Research on DE

Being fascinated by the prospect and potential of DE, many researchers are now working on its improvement, which resulted in many variants of the algorithm. A brief overview of these contemporary research efforts is presented in this section.

Fan and Lampinen [26] proposed a new version of DE which uses an additional mutation operation called trigonometric mutation operation (TMO). This modified DE algorithm is named trigonometric mutation DE (TDE) algorithm. In fact, TDE uses a probabilistic mutation scheme in which the new TMO and the original differential mutation operation are employed stochastically. Introducing an additional control parameter M_t for stochastic mutation, they showed that the TDE algorithm can outperform the classic DE algorithm for some benchmarks and real-world problems [26].

Sun *et al.* [31] proposed DE/EDA, a hybrid of DE and estimation of distribution algorithm (EDA), in which new promising solutions are created by DE/EDA offspring generation scheme. DE/EDE makes use of local information obtained by DE mutation and of global information extracted from a population of solutions by EDA modeling. The presented experimental results demonstrated that DE/EDA outperforms DE and EDA in terms of solution quality within a given number of objective function evaluations. Besides, some other hybrids of DE with PSO have also been proposed [32], [33]. Noman and Iba have proposed a DE variant where they applied EA-like generational model for accelerating the search capability of the algorithm [34].

Recently, some studies on parameter selection for DE [24], [35] found that the performance of DE is sensitive to its control parameters. Therefore, there has been an increasing interest in building new DE algorithms with adaptive control parameters. Zaharie [36] proposed to transform F into a Gaussian random variable. Liu and Lampinen proposed a fuzzy adaptive differential evolution (FADE) which uses fuzzy logic controllers to adapt the mutation and crossover control parameters [24]. The presented experimental results suggest that FADE performs better than traditional DE with all fixed parameters. Brest *et al.* [27] proposed another version of DE that employs self-adaptive parameter control in a way similar to ES. Their proposed algorithm encodes the F and C_r parameters into the chromosome and uses a self-adaptive control mechanism to change them. Their proposed algorithm outperformed the standard DE and FADE algorithm. Das *et al.* [37] have proposed two variants of DE, DERSF, and DETVSF, that use varying scale factors. They showed that those variants outperform the classic DE algorithm.

Qin and Suganthan [25] have taken the self-adaptability of DE one step further by choosing the learning strategy, as well as the parameter settings, adaptively, according to the learning experience. Their proposed self-adaptive DE (SaDE) does not use any particular learning strategy, nor any specific setting for the control parameters F and C_r. SaDE uses its previous learning experience to adaptively select the learning strategy and parameter values, which are often problem dependent.

In our early work [28], we have proposed *fittest individual refinement* (FIR), a crossover-based LS method for DE for high-dimensional optimization problems. The FIR scheme accelerates DE by applying a fixed-length crossover-based search in the neighborhood of the best solution in each generation. Using two different implementations (DEfirDE and DEfirSPX), we showed that the proposed FIR scheme increases the convergence velocity of DE for high-dimensional optimization of well-known benchmark functions.

IV. Differential Evolution With Adaptive XLS

In order to design an effective and efficient MA for global optimization, we need to take advantage of both the exploration abilities of EA and the exploitation abilities of LS by combining them in a well-balanced manner [38]. For successful incorporation of a crossover-based LS (XLS) in an EA, several issues must be resolved, such as the length of the XLS, the selection of individuals which undergo the XLS, the choice of the other parents which participate in the crossover operation, whether deterministic or stochastic application of XLS should be used, etc. Depending on the way the search length is selected, different XLS can be classified into three categories.

Fixed length XLS generates a predetermined number of offspring to search the neighborhood of the parent individuals. This type of search has been used in [17], [28], and [39].

Dynamic length XLS varies the length of the LS gradually with the progress of the search, e.g., by applying longer XLS in the beginning, and gradually applying shorter length XLS towards the end of the search [40].

AHCXLS(I, n_p)

1. P[1] = I
2. Repeat i=2 to n_p times
3. P[i] = select random individuals from the population
4. End Repeat
5. C = ***Crossover*** (P)
6. If C is better than P[1]
   P[1] = C
7. Else
   Return (P[1])
8. Go to step 5

(a)

DEahcSPX

1. Generate an Initial Population P^G
2. ***Evaluate*** P^G
3. B = ***BestIndex***(P^G)
4. P^G.[B] = ***AHCXLS*** (P^G.[B], n_p)
5. For each individual I in P^G
6. ***Reproduce*** an offspring J from I
7. $P^{G+1} = P^{G+1} \cup$ ***Select*** (I, J)
8. Set G = G+1
9. Repeat Step 3 to 8 until termination criteria is met

(b)

Fig. 2. Proposed DEahcSPX algorithm and the adaptive LS scheme AHCXLS. I is the individual on which the AHCXLS is applied and n_p is the total number of individuals that take part in the crossover operation. *BestIndex* return the index of the best individual of the current generation. Other symbols represent standard notations.

SPX

1. Choose n_p parents x_i^G, $i=1,\ldots,n_p$ according to the generational model used and calculate their center of mass O

$$O = \frac{1}{n_p}\sum_{i=1}^{n_p} x_i^G$$

2. Generate random numbers r_i

$$r_i = u^{\frac{1}{i+1}}, (i = 1, \cdots, n_p - 1)$$

where u is a uniform random number $\in [0,1]$

3. Calculate y_i and C_i

$$y_i = O + \varepsilon(x_i^G - O), \quad (i = 1, \cdots, n_p)$$

$$C_i = \begin{cases} 0, & (i = 1) \\ r_{i-1}(y_{i-1} - y_i + C_{i-1}), & (i = 2, \cdots, n_p) \end{cases}$$

where $\varepsilon = 1.0$ is the expansion rate, a control parameter in SPX.

4. Generate an offspring C

$$C = y_{n_p} + C_{n_p}$$

Fig. 3. The simplex crossover (SPX) operation.

Adaptive length XLS determines the direction and length of the search by taking some sort of feedback from the search [40].

In fixed length XLS, it is integral to identify a proper length for the LS since an XLS that is too short may be unsuccessful at exploring the neighborhood of the solution and therefore unsuccessful at improving the search quality. On the other hand, too long an XLS may backfire by consuming additional fitness evaluations unnecessarily. However, finding a single length for XLS that gives optimized results for each problem in each dimension is almost impossible [17]. Similarly, determining a robust adjustment rate is not easy for dynamic length XLS. Therefore, we propose a Lamarckian LS that adaptively determines the length of the search by taking feedback from the search. We call this LS strategy adaptive hill-climbing XLS (AHCXLS) because it uses a simple hill-climbing algorithm to determine the search length adaptively. The pseudocode of AHCXLS is shown in Fig. 2(a).

Another issue in designing XLS is selecting the individuals that will undergo the LS process. XLS can be applied on every individual or on some deterministically/stochastically selected individuals. In principle, the XLS should be applied only to individuals that will productively take the search towards the global optimum. This is particularly important because application of XLS on an ordinary individual may unnecessarily waste function evaluations and turn out to be expensive. Unfortunately, there is no straightforward method of selecting the most promising individuals for XLS. In EC, the solutions with better fitness values are generally preferred for reproduction, as they are more likely to be in the proximity of a basin of attraction. Therefore, we deterministically select the best individual of the population for exploring its neighborhood using the XLS, and thereby we expect to end with a nearby better solution. The other individuals that participate in the crossover operation of XLS are chosen randomly to keep the implementation simple and to promote population diversity. Finally, we have to choose a suitable crossover operator for using in the XLS scheme. Tsutsui *et al.* have proposed simplex crossover (SPX) for real-coded GAs [4]. The SPX operator uses n_p parental vectors for recombination, as shown in Fig. 3.

SPX has various advantages: it does not depend on a coordinate system, the mean vector of parents and offspring generated with SPX are the same and SPX can preserve the covariance matrix of the population with an appropriate parameter setting. These properties make SPX a suitable operator for neighborhood search. Besides, in our preliminary study [28], we found that SPX was a promising operation for local tuning, and therefore we use SPX as the fundamental crossover operation in this study for comparison purpose. More details about the SPX crossover can be found in [4]. The new version of DE with the AHCXLS and SPX operation is titled as DEahcSPX and is described in Fig. 2(b).

The primary difference between the newly proposed DEahcSPX algorithm and our previously proposed DEfirSPX algorithm is that we are no more required to look for a good search length for the XLS operation. The simple rule of hill-climbing adaptively determines the best length by taking feedback from the search. Hence, using the best length (according to the heuristics) for the LS adaptively, the new algorithm makes best use of the function evaluations and thereby identifies the optimum at a higher velocity compared to the earlier proposal. Furthermore, the earlier DEfirSPX is only suitable for high-dimensional optimization problems because of its fixed-length XLS strategy that consumes a fixed number of function evaluations in each call. Such fixed number of function evaluations for local tuning can be considered as negligible compared with the total number of function evaluations allowed for solving higher dimensional problems. On the other hand, because of the adaptive XLS-length adjustment capability of AHCXLS, the newly proposed DEahcSPX algorithm is applicable to optimization problems of any dimension. Finally, because of the simple hill-climbing mechanism, the new adaptive LS does not add any additional complexity or any additional parameter to the original algorithm.

V. Experiments

We have carried out different experiments to assess the performance of DEahcSPX using the test suite described in Appendix I. The test suite consists of 20 unconstrained single-objective benchmark functions with different character-

TABLE I
BEST **ERROR** VALUES AT N = 30, AFTER 300 000 FES

	DE	DEahcSPX		DE	DEahcSPX
F_{sph}	5.73E-17 ± 2.03E-16	**1.75E-31 ± 4.99E-31**	F_1	3.87E-14 ± 2.71E-14	**0.00E+00 ± 0.00E+00**
F_{ros}	5.20E+01 ± 8.56E+01	**4.52E+00 ± 1.55E+01**	F_2	8.50E-02 ± 7.94E-02	**6.52E-05 ± 4.84E-05**
F_{ack}	1.37E-09 ± 1.32E-09	**2.66E-15 ± 0.00E+00**	F_3	3.63E+06 ± 2.06E+06	**1.29E+06 ± 9.22E+05**
F_{grw}	2.66E-03 ± 5.73E-03	**2.07E-03 ± 5.89E-03**	F_4	5.54E+01 ± 6.37E+01	**4.62E+00 ± 8.78E+00**
F_{ras}	2.55E+01 ± 8.14E+00	2.14E+01 ± 1.23E+01	F_5	1.08E+03 ± 5.31E+02	**9.00E+02 ± 4.79E+02**
F_{sch}	4.90E+02 ± 2.34E+02	4.70E+02 ± 2.96E+02	F_6	6.67E+01 ± 1.51E+02	**3.84E+00 ± 3.75E+00**
F_{sal}	2.52E-01 ± 4.78E-02	1.80E-01 ± 4.08E-02	F_7	7.59E-03 ± 8.96E-03	**7.39E-03 ± 6.32E-03**
F_{wht}	3.10E+02 ± 1.07E+02	3.06E+02 ± 1.10E+02	F_8	2.09E+01 ± 1.33E-01	2.09E+01 ± 1.12E-01
F_{pn1}	4.56E-02 ± 1.31E-01	**2.07E-02 ± 8.46E-02**	F_9	2.43E+01 ± 6.23E+00	2.04E+01 ± 8.19E+00
F_{pn2}	1.44E-01 ± 7.19E-01	**1.71E-31 ± 5.35E-31**	F_{10}	7.33E+01 ± 6.62E+01	**5.27E+01 ± 4.84E+01**

istics chosen from the literature. The focus of the study was to compare the performance of the proposed DEahcSPX algorithm with the original DE algorithm in different experiments. We also studied the performance of DEahcSPX comparing with other EAs, and the efficiency of AHCXLS comparing with other XLS strategies. Here, we use DE to denote the *DE/rand/1/bin* variant (if not otherwise specified) of the algorithm and the DEahcSPX algorithm was implemented by embedding the AHCXLS strategy in the same variant of DE.

A. Performance Evaluation Criteria

For evaluating the performance of the algorithms, several of the performance criteria of [41] were used with the difference that 50 instead of 25 trails were conducted, respectively. We compared the performance of DEahcSPX with DE for the test suite using the *function error value*. The *function error value* for a solution x is defined as $(f(x)-f(x^*))$, where x^* is the global optimum of the function. The maximum number of fitness evaluations that we allowed for each algorithm to minimize this error was $10\,000 \times N$, where N is the dimension of the problem. The fitness evaluation criteria were as follows.

1) **Error**: The minimum function error value that an algorithm can find, using $10\,000 \times N$ fitness evaluations at maximum, was recorded in each run and the average and standard deviation of the error values were calculated. The number of trials, in which the algorithms could reach the accuracy level ε (explained in next paragraph) using maximum $10\,000 \times N$ fitness evaluations, were counted and denoted by CNT. For this criterion, the notation $\mathrm{AVG_{Er}} \pm \mathrm{SD_{Er}(CNT)}$ was used in different tables.
2) **Evaluation**: The number of function evaluations (FEs) required to reach an error value less than ε (provided that the maximum limit is $10\,000 \times N$ FEs) was also recorded in different runs and the average and standard deviation of the number of evaluations were calculated. For the functions F_1 to F_5, the accuracy level ε was fixed at 10^{-6} and for the functions F_6–F_{10} ε was fixed at 10^{-2}, as in [41]. We fixed the accuracy level ε for the rest of the functions at 10^{-6}. For this criterion, the notation $\mathrm{AVG_{Ev}} \pm \mathrm{SD_{Ev}(CNT)}$ was used where CNT is the number of runs in which the algorithms could reach this accuracy level ε using $10\,000 \times N$ FEs at maximum.
3) **Convergence graphs**: Convergence graphs of the algorithms. These graphs show the average **Error** performance of the total runs, in respective experiments.

B. Experimental Setup

In our experimentation, we used the same set of initial random populations to evaluate different algorithms in a similar way done in [28] and [42]. Though classic DE uses only three control parameters, namely, *Population Size* P, *Scaling Factor* F, and *Crossover Rate* C_r, choice of these parameters is critical for its performance [24], [35]. F is generally related to the convergence speed. To avoid premature convergence, it is crucial for F to be of sufficient magnitude [23]. $F = 0.9$ is suggested as a good compromise between convergence speed and convergence probability in [43]. Between C_r and F, C_r is much more sensitive to problem's property and multimodality. For searching in nonseparable and multimodal landscapes $C_r = 0.9$ is a good choice [43]. Therefore, we chose $F = 0.9$ and $C_r = 0.9$ for all the functions in every experiment without tuning them to their optimal values for different problems. These parameter settings are also studied elsewhere [24], [43]. Population size is a critical choice for the performance of DE. In our experiments, we investigate the performance of the DE and DEahcSPX with population size $P = N$. We also studied the effect of population size. For the proposed DEahcSPX, no additional parameter setting is required. For the SPX operation, we chose the number of parents participating in the crossover operation to be $n_p = 3$ as suggested in [4] and changes to this setting are also examined later.

The experiments were performed on a computer with 4400 MHz AMD Athlon TM 64 dual core processors and 2 GB of RAM in Java 2 Runtime Environment.

C. Effect of AHCXLS on DE

The results of this section are intended to show how the proposed AHCXLS strategy can improve the performance of DE. In order to show the superiority of the newly proposed DEahcSPX, we compared it with DE carrying out experiments on the test suite at dimension N = 30 and the results are presented in Tables I and II. The functions for which no convergence was achieved were removed from Table II. All the

TABLE II
FEs REQUIRED TO ACHIEVE ACCURACY LEVELS LESS THAN $\varepsilon(\mathrm{N} = 30)$

	DE	DEahcSPX		DE	DEahcSPX
F_{sph}	148650.8 ± 6977.7 (50)	**87027.4 ± 3967.3 (50)** †	F_{pn2}	156016.9 ± 31515.8 (48)	**85360.2 ± 6390.6 (50)** †
F_{ros}	-	**299913.0 ± 519.5 (2)**	F_1	153450.1 ± 5780.4 (50)	**89417.8 ± 4117.6 (50)** †
F_{ack}	215456.1 ± 9721.4 (50)	**129211.6 ± 5168.6 (50)** †	F_2	-	**299279.4 ± 3685.9 (3)**
F_{grw}	190292.5 ± 63478.8 (38)	**121579.2 ± 79563.4 (43)** †	F_7	211778.8 ± 70080.3 (33)	**148067.7 ± 68996.3 (42)** †
F_{pn1}	160955.2 ± 63176.3 (43)	**96149.0 ± 61787.7 (46)** †	-	-	-

† The t value is significant at a 1^{-04} level of significance by two-tailed t-test

settings are the same as mentioned in Section V-B. Some representative graphs comparing the convergence characteristics of DE with DEahcSPX are shown in Fig. 4.

Depending on the relative performance of DEahcSPX and DE, we divided the functions into three classes. The *first class* contains the functions ($F_{\mathrm{sph}}, F_{\mathrm{ack}}, F_{\mathrm{grw}}, F_{\mathrm{pn1}}, F_{\mathrm{pn2}}, F_1$, and F_7) for which DEahcSPX reached the target accuracy level using fewer fitness evaluation, or achieved that in an equal or higher number of trials compared with DE (Table II). The *second class* consists of the functions in which none of the algorithms achieved the desired accuracy level but the newly proposed one reached at a smaller error value. This class contains the functions $F_{\mathrm{ros}}, F_2, F_3, F_4, F_5, F_6$, and F_{10} (Table I). The *third class* contains the functions in which no significant difference was observed in the achieved error values attained by the algorithms. This class consists of the functions $F_{\mathrm{ras}}, F_{\mathrm{sch}}, F_{\mathrm{sal}}, F_{\mathrm{wht}}$, and F_9. Although no significant difference was noticed in the error values, it was revealed by the convergence curves that these error values were achieved using fewer fitness evaluation in the DEahcSPX algorithm compared with DE (Fig. 4). Only in the case of F_8, was no significance difference observed in the algorithms' performance. It seems that the learning strategy of (1) and (2) used in DE was not good enough to locate the global optimum for the functions belonging to the second class and the third class. Since the DEahcSPX algorithm depends mostly on the working principle of DE, it is natural that it also could not locate the global optimal using the same learning strategy. However, hybridization of DE with the AHCXLS scheme notably speeds up the original algorithm. In general, the overall results of Tables I and II and the graphs of Fig. 4 substantiate our claim that the proposed AHCXLS strategy accelerates the classic DE algorithm.

D. Sensitivities to Population Size

Performance of DE is always sensitive to the selected population size [28], [35]. This is easily conceivable because DE employs a one-to-one reproduction strategy. Therefore, if a very large population size is selected, then DE exhausts the fitness evaluations very quickly without being able to locate the optimum. Storn and Price suggested a larger population size (between $5\mathrm{N}$ to $10\mathrm{N}$) for DE [5], although later studies found that DE performs better with a smaller population [28], [43]. To investigate the sensitivity of the proposed algorithm to variations of population size, we experimented with different population sizes at dimension $\mathrm{N} = 30$. Results, reported in Table III, show how drastically the performance of DE changes with the population size for a given maximum number of evaluations. For some functions, DE converged for all trials using a smaller population size (e.g., $\mathrm{P} = \mathrm{N}$) but failed to reach even a single convergence with a larger population (e.g., $\mathrm{P} = 10\mathrm{N}$). Since DEahcSPX is just an improvement of basic DE using AHCXLS, it is expected that its sensitivity to variation in population size is more or less similar to that of the basic algorithm. However, Table III shows that in all experiments the error values achieved by DEahcSPX were always better than those achieved by DE. The graphs of Fig. 5 show that AHCXLS scheme has improved the convergence characteristics of the original algorithm, regardless to population size. Though for some functions ($F_{\mathrm{ras}}, F_{\mathrm{sch}}, F_8, F_9, F_{10}$), the performance of both algorithms were more or less indifferent to population size, we believe that it was because of the inadequacy of the learning strategy used. Nevertheless, the results presented in this section confirm that the proposed DEahcSPX algorithm exhibits a higher convergence velocity and greater robustness to the population size compared with DE.

E. Scalability Study

So far, we have experimented in $\mathrm{N} = 30$ dimensional problem space. In order to study the effect of problem dimension on the performance of the DEahcSPX algorithm, we carried out a scalability study comparing with the original DE algorithm. Since the functions F_1–F_{10} are defined up to $\mathrm{N} = 50$ dimensions, we studied them at $\mathrm{N} = 10$ and 50 dimensions. The other functions were studied at $\mathrm{N} = 10, 50, 100$, and 200 dimensions. For $\mathrm{N} = 10$ dimensions, population size was chosen as $\mathrm{P} = 30$ and for all other dimensions, it was selected as $\mathrm{P} = \mathrm{N}$. The accuracy achieved using $N \times 10\,000$ fitness evaluations are presented in Table IV, and some representative convergence graphs are shown in Fig. 6. In order to focus on the comparison between the proposed algorithm DEahcSPX and its parent algorithm DE, in Table V we also compared the fitness evaluations required by the algorithms to achieve the accuracy level ε at $\mathrm{N} = 10$ dimensions. In general, the same conclusion as in Section V-C can be drawn about the relative performance of the algorithms, i.e., DEahcSPX outperformed DE at every dimension. Moreover, the results also show that the performance improvement becomes more substantial with the increase in problem dimensionality. So, from the experimental results of this section, we can conclude that the AHCXLS scheme speeds up DE in general, but particularly significant improvements are obtained at higher dimensionality.

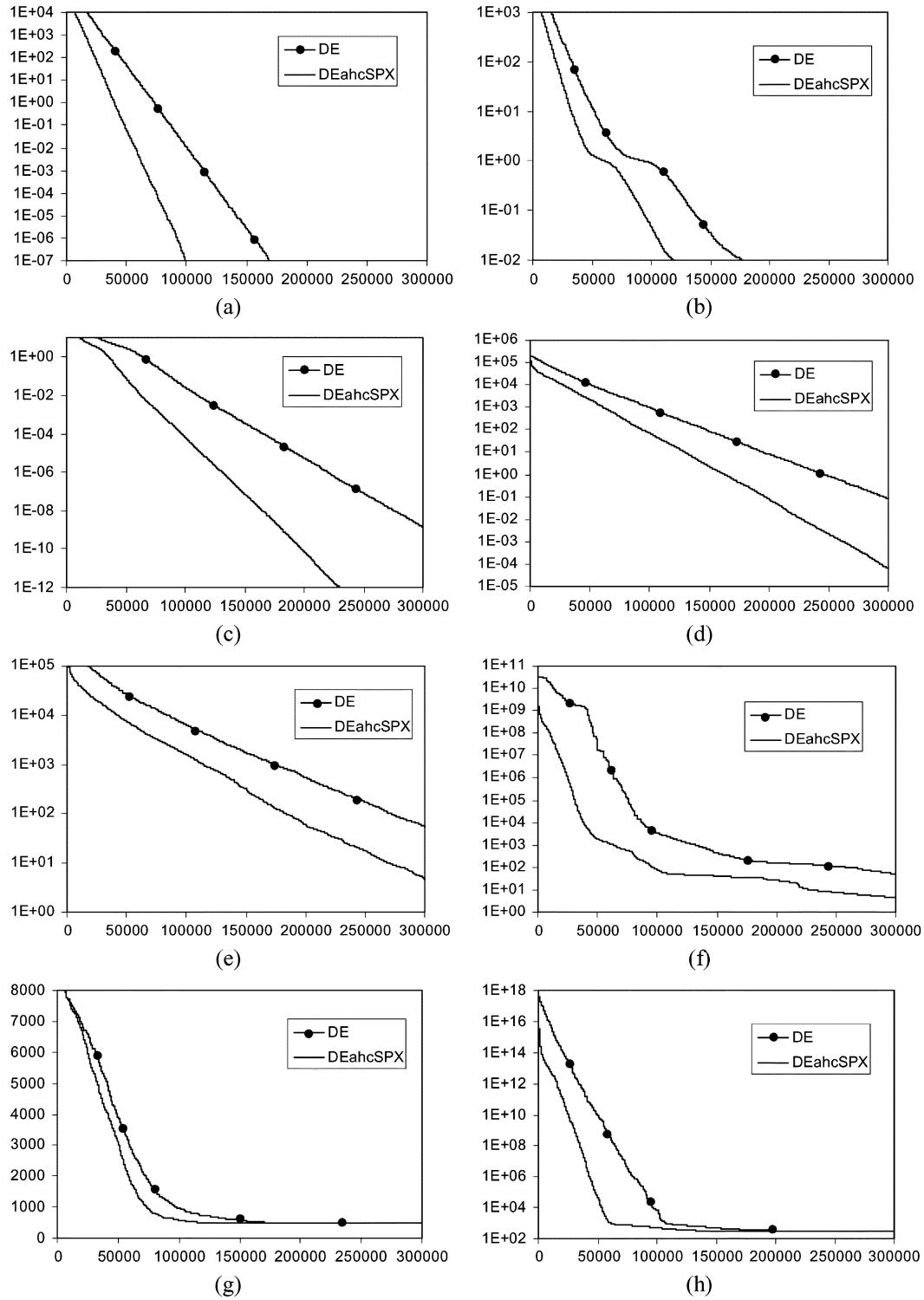

Fig. 4. Convergence curves of DE and DEahcSPX algorithm for selected functions ($\mathrm{N} = 30$). X axis represents fitness evaluations (FEs) and Y axis represents **Error** values. (a) F_1. (b) F_7. (c) F_{ack}. (d) F_2. (e) F_4. (f) F_{ros}. (g) F_{sch}. (h) F_{wht}.

F. Comparison With Other XLS

In order to show the superiority of the newly proposed AHCXLS scheme, we also compared it with two other XLS strategies applying in DE algorithm. The first one is the FIR strategy proposed by Noman and Iba [28], and we denote this memetic version of DE as DEfirSPX. The other algorithm, denoted as DExhcSPX, was implemented by using the XHC strategy proposed by Lozano *et al.* [17]. Both FIR and XHC belong to the fixed length XLS category and were implemented using SPX crossover operation, in order to have an unbiased comparison. Experiments were performed on the test suite at dimension $\mathrm{N} = 30$. Results are presented in Tables VI and VII. The settings for FIR and XHC schemes were chosen, as

TABLE III
BEST **ERROR** VALUES FOR VARYING POPSIZE AT $\mathrm{N} = 30$, AFTER 300 000 FES

PopSize=50			**PopSize=100**		
	DE	DEahcSPX		DE	DEahcSPX
F_{sph}	2.31E-02 ± 1.92E-02	**6.03E-09 ± 6.86E-09 (50)**	F_{sph}	3.75E+03 ± 1.14E+03	**3.11E+01 ± 1.88E+01**
F_{ros}	3.70E+02 ± 4.81E+02	**4.98E+01 ± 6.22E+01**	F_{ros}	4.03E+08 ± 2.59E+08	**1.89E+05 ± 1.47E+05**
F_{ack}	3.60E-02 ± 1.82E-02	**1.89E-05 ± 1.19E-05**	F_{ack}	1.36E+01 ± 1.48E+00	**3.23E+00 ± 5.41E-01**
F_{grw}	5.00E-02 ± 6.40E-02	**1.68E-03 ± 4.25E-03 (42)**	F_{grw}	3.57E+01 ± 1.26E+01	**1.29E+00 ± 1.74E-01**
F_{ras}	5.91E+01 ± 2.65E+01	**2.77E+01 ± 1.31E+01**	F_{ras}	2.63E+02 ± 2.79E+01	1.64E+02 ± 2.16E+01
F_{sch}	7.68E+02 ± 8.94E+02	**2.51E+02 ± 1.79E+02**	F_{sch}	6.56E+03 ± 4.25E+02	6.30E+03 ± 4.80E+02
F_{sal}	8.72E-01 ± 1.59E-01	**2.44E-01 ± 5.06E-02**	F_{sal}	5.97E+00 ± 6.54E-01	**1.20E+00 ± 2.12E-01**
F_{wht}	8.65E+02 ± 1.96E+02	**4.58E+02 ± 7.56E+01**	F_{wht}	1.29E+14 ± 1.60E+14	**3.16E+08 ± 4.48E+08**
F_{pn1}	2.95E-04 ± 1.82E-04	**1.12E-09 ± 2.98E-09 (50)**	F_{pn1}	6.94E+04 ± 1.58E+05	**2.62E+00 ± 1.31E+00**
F_{pn2}	9.03E-03 ± 2.03E-02	**4.39E-04 ± 2.20E-03 (47)**	F_{pn2}	6.60E+05 ± 7.66E+05	**4.85E+00 ± 1.59E+00**
F_1	1.69E-02 ± 1.80E-02	**1.67E-08 ± 2.19E-08 (50)**	F_1	5.68E+03 ± 2.63E+03	**4.31E+01 ± 2.16E+01**
F_2	8.38E+02 ± 7.20E+02	**1.55E+01 ± 1.09E+01**	F_2	5.79E+04 ± 1.53E+04	**4.34E+03 ± 1.57E+03**
F_3	5.86E+07 ± 2.61E+07	**4.75E+06 ± 1.82E+06**	F_3	8.82E+08 ± 2.61E+08	**1.97E+07 ± 4.84E+06**
F_4	3.65E+03 ± 2.03E+03	**2.31E+02 ± 1.42E+02**	F_4	9.45E+04 ± 2.77E+04	**9.55E+03 ± 3.93E+03**
F_5	3.20E+03 ± 1.31E+03	**1.04E+03 ± 3.67E+02**	F_5	2.33E+04 ± 4.03E+03	**5.88E+03 ± 1.24E+03**
F_6	5.64E+02 ± 7.58E+02	**7.00E+01 ± 1.28E+02**	F_6	7.27E+08 ± 5.08E+08	**4.05E+05 ± 3.01E+05**
F_7	9.54E-01 ± 9.75E-02	**3.19E-03 ± 5.14E-03 (44)**	F_7	5.73E+02 ± 1.85E+02	**1.18E+01 ± 5.78E+00**
F_8	2.09E+01 ± 5.94E-02	2.09E+01 ± 5.25E-02	F_8	2.09E+01 ± 3.84E-02	2.09E+01 ± 5.89E-02
F_9	5.23E+01 ± 2.36E+01	**2.56E+01 ± 1.48E+01**	F_9	2.73E+02 ± 1.53E+01	**1.83E+02 ± 2.25E+01**
F_{10}	2.24E+02 ± 1.85E+01	1.55E+02 ± 4.53E+01	F_{10}	3.31E+02 ± 3.53E+01	**2.05E+02 ± 1.55E+01**
PopSize=200			**PopSize=300**		
	DE	DEahcSPX		DE	DEahcSPX
F_{sph}	4.01E+04 ± 6.26E+03	**1.10E+03 ± 2.98E+02**	F_{sph}	1.96E+04 ± 2.00E+03	**6.93E+02 ± 1.34E+02**
F_{ros}	1.53E+10 ± 4.32E+09	**1.49E+07 ± 7.82E+06**	F_{ros}	3.97E+09 ± 8.92E+08	**5.35E+06 ± 2.82E+06**
F_{ack}	2.02E+01 ± 2.20E-01	**9.11E+00 ± 7.81E-01**	F_{ack}	1.79E+01 ± 3.51E-01	**7.23E+00 ± 4.50E-01**
F_{grw}	3.73E+02 ± 6.03E+01	**1.08E+01 ± 2.02E+00**	F_{grw}	1.79E+02 ± 1.60E+01	**7.26E+00 ± 1.74E+00**
F_{ras}	3.62E+02 ± 2.12E+01	**2.05E+02 ± 1.85E+01**	F_{ras}	2.75E+02 ± 1.27E+01	2.03E+02 ± 1.49E+01
F_{sch}	6.88E+03 ± 2.55E+02	6.72E+03 ± 3.24E+02	F_{sch}	6.87E+03 ± 2.72E+02	6.80E+03 ± 3.37E+02
F_{sal}	1.34E+01 ± 8.41E-01	**3.25E+00 ± 4.55E-01**	F_{sal}	1.52E+01 ± 5.43E-01	**3.59E+00 ± 4.54E-01**
F_{wht}	2.29E+16 ± 1.16E+16	**5.47E+10 ± 6.17E+10**	F_{wht}	2.96E+16 ± 1.09E+16	**1.83E+11 ± 1.72E+11**
F_{pn1}	2.44E+07 ± 7.58E+06	**9.10E+00 ± 2.42E+00**	F_{pn1}	3.71E+07 ± 1.29E+07	**1.09E+01 ± 3.76E+00**
F_{pn2}	8.19E+07 ± 1.99E+07	**6.18E+01 ± 6.30E+01**	F_{pn2}	1.03E+08 ± 1.87E+07	**3.42E+02 ± 4.11E+02**
F_1	5.51E+04 ± 6.74E+03	**2.04E+03 ± 5.09E+02**	F_1	5.18E+03 ± 7.23E+02	**4.37E+02 ± 8.16E+01**
F_2	1.16E+05 ± 1.60E+04	**1.09E+04 ± 3.00E+03**	F_2	2.88E+04 ± 3.54E+03	**7.08E+03 ± 1.22E+03**
F_3	1.19E+09 ± 1.63E+08	**4.02E+07 ± 1.48E+07**	F_3	1.56E+08 ± 3.07E+07	**2.69E+07 ± 6.84E+06**
F_4	1.43E+05 ± 2.63E+04	**1.68E+04 ± 3.80E+03**	F_4	3.49E+04 ± 4.96E+03	**1.10E+04 ± 1.93E+03**
F_5	3.29E+04 ± 2.71E+03	**9.12E+03 ± 1.63E+03**	F_5	1.10E+04 ± 5.32E+02	**7.64E+03 ± 4.72E+02**
F_6	2.61E+10 ± 9.11E+09	**4.64E+07 ± 1.65E+07**	F_6	1.86E+08 ± 4.07E+07	**1.48E+06 ± 5.45E+05**
F_7	3.46E+03 ± 4.31E+02	**1.50E+02 ± 2.82E+01**	F_7	4.01E+03 ± 5.13E+02	**2.57E+02 ± 5.97E+01**
F_8	2.09E+01 ± 6.07E-02	2.09E+01 ± 5.99E-02	F_8	2.10E+01 ± 4.44E-02	2.09E+01 ± 5.22E-02
F_9	4.13E+02 ± 2.46E+01	**2.31E+02 ± 2.10E+01**	F_9	2.22E+02 ± 1.03E+01	2.05E+02 ± 1.35E+01
F_{10}	6.00E+02 ± 5.28E+01	**2.65E+02 ± 1.61E+01**	F_{10}	2.75E+02 ± 1.30E+01	2.13E+02 ± 1.21E+01

suggested in [17] and [28], respectively. All the other settings are the same, as mentioned in Section V-B.

The performance difference among these three XLS methods is not obvious from Table VI because at the end of the search all of them reached similar error values, though DEahcSPX found slightly better error values in almost every case. However, the results presented in Table VII reveals that the newly proposed DEahcSPX algorithm was faster than the other two variants of DE. Statistical analysis of the numbers of FEs needed to reach the given accuracy level (i.e., the results of Table VII) was per-

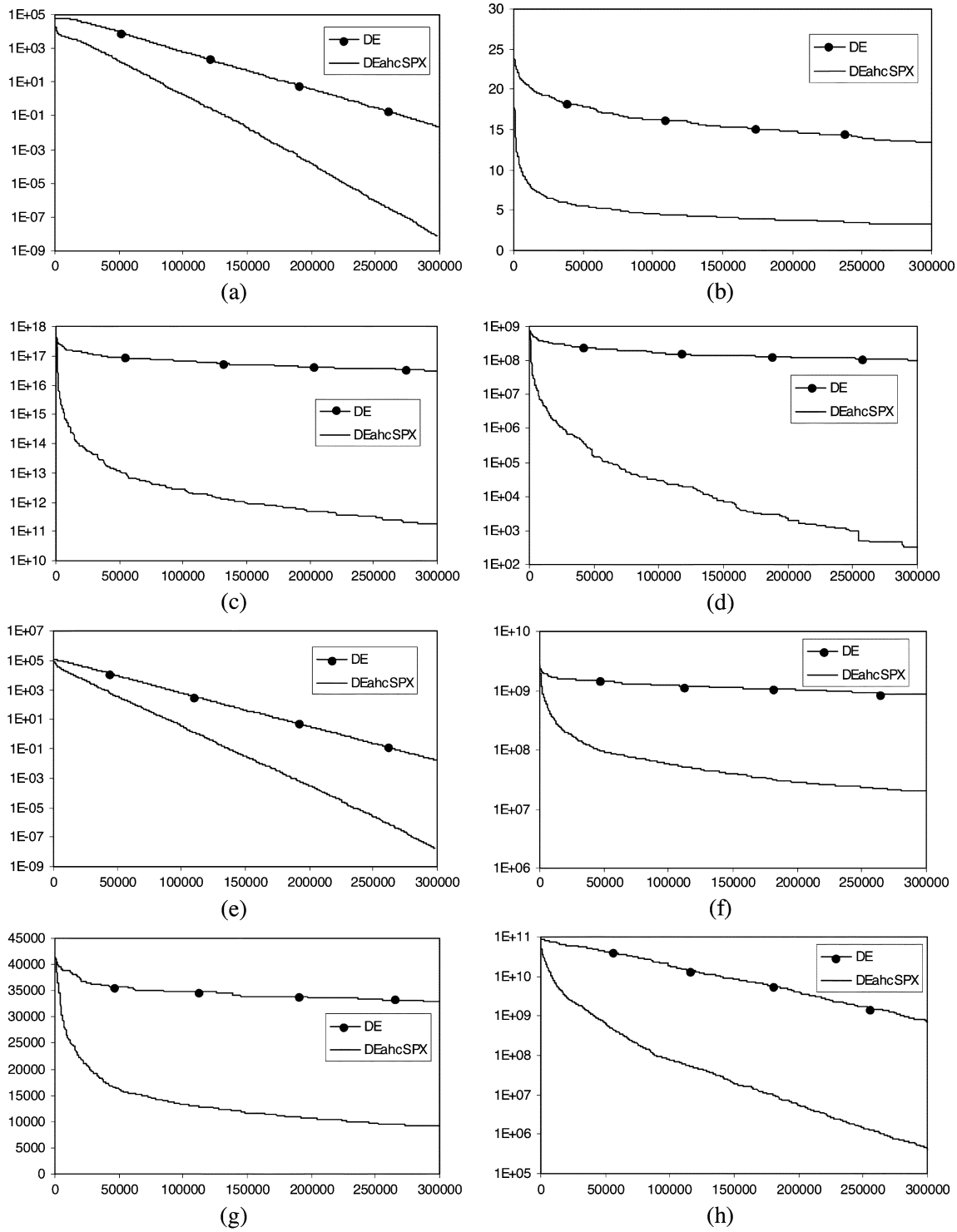

Fig. 5. Convergence curves to show the sensitivities of DE and DEahcSPX to population size for selected functions ($\mathrm{N} = 30$). X axis represents FEs and Y axis represents **Error** values. (a) $F_{\mathrm{sph}}(\mathrm{P} = 50)$. (b) $F_{\mathrm{sal}}(\mathrm{P} = 200)$. (c) $F_{\mathrm{wht}}(\mathrm{P} = 300)$. (d) $F_{\mathrm{pn2}}(\mathrm{P} = 300)$. (e) $F_1(\mathrm{P} = 50)$. (f) $F_3(\mathrm{P} = 100)$. (g) $F_5(\mathrm{P} = 200)$. (h) $F_6(\mathrm{P} = 100)$.

formed using two-tailed Student's *t*-test, and it was found that the differences between the results of DEahcSPX and the other two algorithms are statistically significant at a level of 0.05 for all the functions in which the algorithms found convergences in at least 40 trials (i.e., $F_{\mathrm{sph}}, F_{\mathrm{ack}}, F_{\mathrm{pn1}}, F_{\mathrm{pn2}}$, and F_1). Besides, the most prominent advantage of the AHCXLS scheme over the other two is that it is free from the lookup for the best length for the LS and thereby does not need any additional parameter. In contrast, for best results, XHC and FIR schemes need to tune two and one parameters, respectively, which in turn should be determined experimentally. Moreover, AHCXLS is also useful for lower dimensional problems, whereas the FIR scheme is only suitable for high dimensional optimization. At lower dimension (e.g., at $\mathrm{N} = 10$), the performance of DEfirSPX was not significantly different from that of DE, and even poor in some cases. Furthermore, in our brief experimentations,

TABLE IV
SCALABILITY STUDY IN TERMS OF **ERROR** VALUES

N=10			**N=50**		
	DE	DEahcSPX		DE	DEahcSPX
F_{sph}	3.26E-28 ± 5.83E-28 (50)	**1.81E-38 ± 4.94E-38 (50)**	F_{sph}	5.91E-02 ± 9.75E-02	**8.80E-09 ± 2.80E-08 (50)**
F_{ros}	4.78E-01 ± 1.32E+00 (43)	**3.19E-01 ± 1.10E+00 (46)**	F_{ros}	1.13E+10 ± 2.34E+10	**1.63E+02 ± 3.02E+02**
F_{ack}	8.35E-15 ± 8.52E-15 (50)	**2.66E-15 ± 0.00E+00 (50)**	F_{ack}	2.39E-02 ± 8.90E-03	**1.69E-05 ± 8.86E-06**
F_{grw}	5.75E-02 ± 3.35E-02	**4.77E-02 ± 2.55E-02**	F_{grw}	7.55E-02 ± 1.14E-01	**2.96E-03 ± 5.64E-03 (36)**
F_{ras}	1.85E+00 ± 1.68E+00 (13)	**1.60E+00 ± 1.61E+00 (18)**	F_{ras}	6.68E+01 ± 2.36E+01	**3.47E+01 ± 9.23E+00**
F_{sch}	14.21272743 ± 39.28155167	**4.73766066 ± 23.68766692**	F_{sch}	1.07E+03 ± 5.15E+02	**9.56E+02 ± 2.88E+02**
F_{sal}	0.107873375 ± 0.027688791	**0.099873361 ± 3.47E-08**	F_{sal}	1.15E+00 ± 1.49E-01	**4.00E-01 ± 1.00E-01**
F_{wht}	18.11229734 ± 15.85783313	18.00697444 ± 13.11270338	F_{wht}	1.43E+05 ± 4.10E+05	**1.41E+03 ± 2.90E+02**
F_{pn1}	3.85E-29 ± 7.28E-29 (50)	**4.71E-32 ± 1.12E-47 (50)**	F_{pn1}	3.07E-02 ± 7.93E-02	**2.49E-03 ± 1.24E-02 (48)**
F_{pn2}	1.49E-28 ± 2.20E-28 (50)	**1.35E-32 ± 5.59E-48 (50)**	F_{pn2}	2.24E-01 ± 3.35E-01	**2.64E-03 ± 4.79E-03 (38)**
F_1	0.00E+00 ± 0.00E+00 (50)	**0.00E+00 ± 0.00E+00 (50)**	F_1	1.50E-02 ± 1.09E-02	**1.06E-08 ± 1.22E-08 (50)**
F_2	2.27E-15 ± 1.14E-14 (50)	**0.00E+00 ± 0.00E+00 (50)**	F_2	2.89E+04 ± 1.03E+04	**1.44E+03 ± 5.95E+02**
F_3	8.76E-06 ± 2.78E-05 (38)	**2.42E-06 ± 7.11E-06 (40)**	F_3	5.40E+08 ± 2.62E+08	**2.27E+07 ± 8.01E+06**
F_4	8.87E-14 ± 1.24E-13 (50)	**0.00E+00 ± 0.00E+00 (50)**	F_4	6.04E+04 ± 1.74E+04	**1.04E+04 ± 3.90E+03**
F_5	1.07E-03 ± 2.40E-03	**1.12E-05 ± 1.75E-05 (12)**	F_5	5.81E+03 ± 1.12E+03	**3.71E+03 ± 6.57E+02**
F_6	3.19E-01 ± 1.10E+00 (46)	3.19E-01 ± 1.10E+00 (46)	F_6	1.29E+03 ± 1.98E+03	**2.24E+02 ± 3.99E+02**
F_7	1.56E-01 ± 1.63E-01	**1.47E-01 ± 1.16E-01 (3)**	F_7	1.05E+00 ± 6.24E-02	**2.11E-02 ± 2.29E-02 (18)**
F_8	2.04E+01 ± 1.08E-01	2.04E+01 ± 1.45E-01	F_8	2.11E+01 ± 2.93E-02	2.11E+01 ± 3.63E-02
F_9	2.01E+00 ± 1.41E+00 (5)	**1.23E+00 ± 9.65E-01 (10)**	F_9	7.65E+01 ± 2.30E+01	**5.23E+01 ± 1.53E+01**
F_{10}	1.26E+01 ± 7.26E+00	1.06E+01 ± 4.06E+00	F_{10}	4.24E+02 ± 2.98E+01	3.35E+02 ± 2.80E+01
N=100			**N=200**		
	DE	DEahcSPX		DE	DEahcSPX
F_{sph}	4.28E+03 ± 1.27E+03	**5.01E+01 ± 8.94E+01**	F_{sph}	1.26E+05 ± 1.06E+04	**7.01E+03 ± 1.07E+03**
F_{ros}	3.33E+08 ± 1.67E+08	**1.45E+05 ± 1.11E+05**	F_{ros}	2.97E+10 ± 3.81E+09	**1.11E+08 ± 2.63E+07**
F_{ack}	8.81E+00 ± 8.07E-01	**1.91E+00 ± 3.44E-01**	F_{ack}	1.81E+01 ± 2.26E-01	**8.45E+00 ± 4.13E-01**
F_{grw}	3.94E+01 ± 8.01E+00	**1.23E+00 ± 2.14E-01**	F_{grw}	1.15E+03 ± 9.22E+01	**6.08E+01 ± 9.30E+00**
F_{ras}	8.30E+02 ± 6.51E+01	**4.75E+02 ± 6.55E+01**	F_{ras}	2.37E+03 ± 7.24E+01	1.53E+03 ± 8.31E+01
F_{sch}	2.54E+04 ± 2.15E+03	2.48E+04 ± 2.17E+03	F_{sch}	6.66E+04 ± 1.32E+03	6.61E+04 ± 1.44E+03
F_{sal}	1.02E+01 ± 7.91E-01	**3.11E+00 ± 5.79E-01**	F_{sal}	3.69E+01 ± 1.80E+00	**1.10E+01 ± 4.38E-01**
F_{wht}	5.44E+15 ± 5.07E+15	**4.06E+10 ± 6.57E+10**	F_{wht}	3.13E+18 ± 9.48E+17	**4.21E+13 ± 1.74E+13**
F_{pn1}	6.20E+05 ± 7.38E+05	**4.34E+00 ± 1.75E+00**	F_{pn1}	3.49E+08 ± 7.60E+07	**2.27E+01 ± 5.73E+00**
F_{pn2}	4.34E+06 ± 2.30E+06	**7.25E+01 ± 2.44E+01**	F_{pn2}	8.08E+08 ± 1.86E+08	**6.24E+04 ± 4.77E+04**

we found that the performance difference among the proposed algorithm and the other two variants became more significant at higher dimensions.

G. Comparison With Other EC

Many XLS-oriented EAs for real parameter optimization are now available in the literature. This subsection presents a performance comparison between the proposed algorithm and some other hybrid GAs with LS. Two GA models, minimal generation gap (MGG) [44] and generalized generation gap (G3) [45], have drawn much attention. Both of these models, in fact, induce an XLS on the neighborhood of the parents by generating multiple offspring using some crossover operation [17]. Over the past few years, substantial research effort has been spent to develop more sophisticated crossover operations for GA and many outstanding schemes have been proposed, such as BLX-α crossover [46], unimodal normal distribution crossover (UNDX) [3], simplex crossover (SPX) [4], and parent centric crossover (PCX) [45]. Respective researches have shown that UNDX and SPX perform best with the MGG and PCX performs best with the G3 generational models. Therefore, in our experiments, we perform comparisons using the algorithms MGG+UNDX, MGG+SPX, G3+PCX and G3+SPX, and the results are shown in Tables VIII and IX. The performance of G3+SPX was similar to or worse than that of MGG+SPX. Therefore, only the results of MGG+SPX were presented. The MGG model was setup with $P = 300, \lambda = 4$ offspring, generated from μ parents, where $\mu = 6$ was used for UNDX and $\mu = 3$ was used for SPX. For G3 model $P = 100, \mu = 3$, and $\lambda = 2$ were used.

In our experiments, the MGG+SPX algorithm could not achieve the target accuracy levels for any function of the test

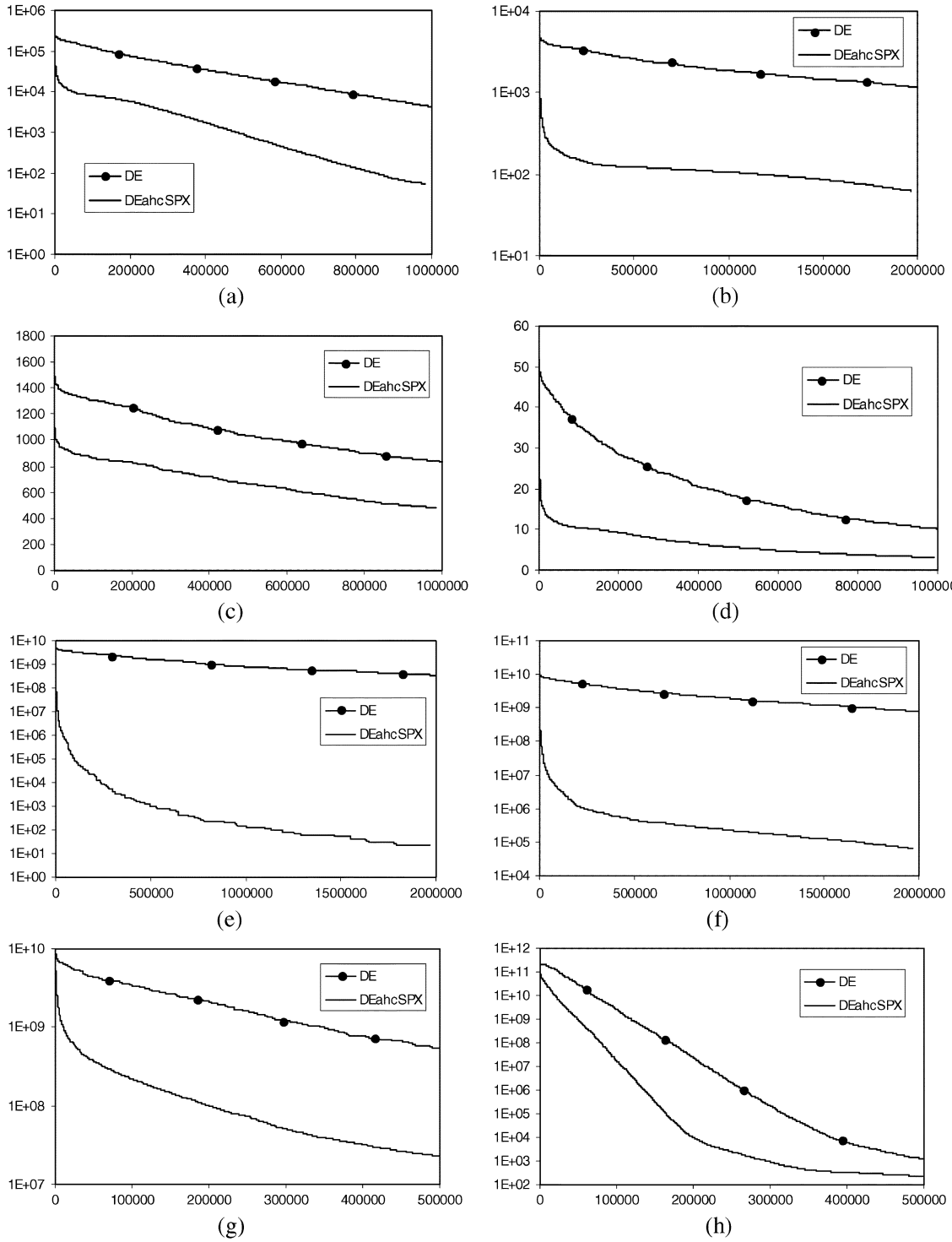

Fig. 6. Convergence curves to compare the scalability of DE and DEahcSPX algorithm for selected functions. X axis represents FEs and Y axis represents **Error** values. (a) $F_{\mathrm{sph}}(\mathrm{N}=100)$. (b) $F_{\mathrm{grw}}(\mathrm{N}=200)$. (c) $F_{\mathrm{ras}}(\mathrm{N}=100)$. (d) $F_{\mathrm{sal}}(\mathrm{N}=100)$. (e) $F_{\mathrm{pn1}}(\mathrm{N}=200)$. (f) $F_{\mathrm{pn2}}(\mathrm{N}=200)$. (g) $F_3(\mathrm{N}=50)$. (h) $F_6(\mathrm{N}=50)$.

suite. The MGG+UNDX algorithm achieved a slightly better error average for some functions ($F_{\mathrm{grw}}, F_{\mathrm{ras}}, F_{\mathrm{sal}}, F_3, F_7$, and F_{10}) but was outperformed by DEahcSPX for the other functions. Moreover, according to Table IX, the average fitness evaluations used by DEahcSPX were fewer than that used by MGG+UNDX to achieve the target accuracy levels ε. The performance of G3+PCX was outstanding for unimodal functions like $F_{\mathrm{sph}}, F_{\mathrm{ros}}, F_1$, and F_2. However, its performance was poor for the multimodal functions. In most of the cases, the algorithm converged quickly without reaching the error accuracy level and without exhausting the maximum fitness evaluations, as indicated in Tables VIII and IX. So, in general, it can be concluded from Tables VIII and IX that the proposed DEahcSPX exhibits overall better performance than the other

TABLE V
FEs REQUIRED TO ACHIEVE ACCURACY LEVELS LESS THAN $\varepsilon(\mathrm{N} = 10)$

	DE	DEahcSPX		DE	DEahcSPX
F_{sph}	31639.7 ± 1347.0 (50)	**22926.4 ± 1300.3 (50)**	F_2	49683.4 ± 2184.3 (50)	**34677.8 ± 2203.9 (50)**
F_{ros}	73803.8 ± 12550.9 (43)	**59275.7 ± 14998.0 (46)**	F_3	94850.3 ± 4906.4 (38)	**89217.0 ± 9466.3 (40)**
F_{ack}	48898.2 ± 1977.7 (50)	**36389.3 ± 1764.4 (50)**	F_4	58143.8 ± 3372.4 (50)	**39192.2 ± 2673.2 (50)**
F_{ras}	94089.0 ± 12818.3 (13)	**84309.0 ± 22045.5 (18)**	F_5	-	**99328.9 ± 1606.6 (12)**
F_{pn1}	28885.8 ± 2394.4 (50)	**20543.5 ± 1162.8 (50)**	F_6	61808.5 ± 18899.9 (46)	**50167.7 ± 19785.8 (46)**
F_{pn2}	30812.6 ± 1684.9 (50)	**21633.5 ± 1293.9 (50)**	F_7	-	**97258.7 ± 13794.1 (3)**
F_1	32165.7 ± 1415.3 (50)	**22594.7 ± 1255.7 (50)**	F_9	97860.2 ± 7475.7 (5)	**89685.7 ± 21519.0 (10)**

TABLE VI
COMPARISON WITH OTHER XLS IN TERMS OF **ERROR** VALUES

	DEahcSPX	DEfirSPX	DExhcSPX
F_{sph}	**1.75E-31 ± 4.99E-31**	1.22E-27 ± 2.95E-27	7.66E-29 ± 1.97E-28
F_{ros}	**4.52E+00 ± 1.55E+01**	4.84E+00 ± 3.37E+00	5.81E+00 ± 4.73E+00
F_{ack}	**2.66E-15 ± 0.00E+00**	8.35E-15 ± 1.03E-14	5.22E-15 ± 2.62E-15
F_{grw}	**2.07E-03 ± 5.89E-03**	3.54E-03 ± 7.55E-03	3.45E-03 ± 7.52E-03
F_{ras}	2.14E+01 ± 1.23E+01	2.27E+01 ± 7.39E+00	**1.86E+01 ± 7.05E+00**
F_{sch}	**4.70E+02 ± 2.96E+02**	5.23E+02 ± 3.73E+02	4.91E+02 ± 4.60E+02
F_{sal}	**1.80E-01 ± 4.08E-02**	1.84E-01 ± 7.46E-02	1.92E-01 ± 4.93E-02
F_{wht}	3.06E+02 ± 1.10E+02	3.11E+02 ± 9.38E+01	**2.84E+02 ± 1.10E+02**
F_{pn1}	**2.07E-02 ± 8.46E-02**	3.24E-02 ± 3.44E-02	2.49E-02 ± 8.61E-02
F_{pn2}	**1.71E-31 ± 5.35E-31**	1.76E-03 ± 4.11E-03	4.39E-04 ± 2.20E-03
F_1	**0.00E+00 ± 0.00E+00**	**0.00E+00 ± 0.00E+00**	**0.00E+00 ± 0.00E+00**
F_2	**6.52E-05 ± 4.84E-05**	1.05E-03 ± 1.29E-03	9.40E-04 ± 1.80E-03
F_3	**1.29E+06 ± 9.22E+05**	1.73E+06 ± 1.22E+06	1.54E+06 ± 1.15E+06
F_4	**4.62E+00 ± 8.78E+00**	1.04E+01 ± 1.75E+01	6.69E+00 ± 1.06E+01
F_5	**9.00E+02 ± 4.79E+02**	1.15E+03 ± 6.68E+02	1.01E+03 ± 4.31E+02
F_6	**3.84E+00 ± 3.75E+00**	1.65E+01 ± 4.72E+01	1.41E+01 ± 1.86E+01
F_7	7.39E-03 ± 6.32E-03	**4.53E-03 ± 6.92E-03**	7.98E-03 ± 9.48E-03
F_8	2.09E+01 ± 1.12E-01	2.10E+01 ± 4.61E-02	2.09E+01 ± 7.41E-02
F_9	**2.04E+01 ± 8.19E+00**	2.47E+01 ± 7.72E+00	2.80E+01 ± 7.75E+00
F_{10}	**5.27E+01 ± 4.84E+01**	6.96E+01 ± 5.39E+01	6.79E+01 ± 4.80E+01

algorithms shown in the tables. These results also establish it as a competitive alternative for real parameter optimization problems.

We also compared the proposed DEahcSPX algorithm with other MAs with binary coding and real coding using the published results. To show that the proposed AHCXLS is equally suitable for the exponential crossover scheme, in these comparisons, we used exponential crossover in DE and DEahcSPX instead of binary crossover. First, we performed comparison with self-adaptive MA scheme MA-S2, which is the best of the two adaptive MAs proposed in [19], and also exhibited overall superior performance compared to nine other traditional MAs. The comparative results are presented in Table X in terms of eight benchmark functions used in [19], among which the Bump function (F_{Bump}) is a constrained maximization problem whether all the others are unconstrained minimization problems. The maximum FEs allowed to solve each function was 40 000 except for F_{Bump}, where it was 100 000. The results presented are average of 20 repeated runs, as in [19]. From Table X, it can be found that for $F_{\mathrm{Bump}}, F_{\mathrm{Rosenbrock}}, F_{\mathrm{Step}}$, and $F_{\mathrm{Rastrigin}}$ functions, the DEahcSPX algorithm clearly outperformed the MA-S2 algorithm. For the other four functions, it seems that MA-S2 exhibited superior performance.

Then, we compared our algorithm with the results of the RCMA presented in [17]. Comparing with the other 21 variants of real coded MAs, Lozano *et al.* showed that in general, their proposed RCMA outperforms all the other algorithms [17]. Table XI shows comparative results for five benchmark functions and three real-world problems as used in [17]. We used the same performance measure criteria as used in [17]; **A**: average of minimum fitness found in 50 repeated runs; **B**: best of all minimum fitness in 50 runs or the percentage of run in which the global optimum was found (if some runs located the global minimum). The maximum FEs allowed

TABLE VII
COMPARISON WITH OTHER XLS IN TERMS OF **FEs**

	DEahcSPX	DEfirSPX	DExhcSPX
F_{sph}	**87027.4 ± 3967.3 (50)**	96588.2 ± 6260.4 (50)	92111.4 ± 4951.5 (50)
F_{ros}	**299913.0 ± 519.5 (2)**	-	-
F_{ack}	**129211.6 ± 5168.6 (50)**	142169.88 ± 8137.9 (50)	139982.1 ± 7096.2 (50)
F_{grw}	**121579.2 ± 79563.4 (43)**	146999.76 ± 87855.0 (38)	153119.1 ± 93613.4 (37)
F_{pn1}	**96149.0 ± 61787.7 (46)**	126486.56 ± 66369.7 (44)	122129.1 ± 68013.8 (44)
F_{pn2}	**85360.2 ± 6390.6 (50)**	135395.48 ± 73557.7 (43)	106820.1 ± 41154.6 (48)
F_1	**89417.8 ± 4117.6 (50)**	101022.68 ± 7656.8 (50)	97470.6 ± 7700.0 (50)
F_2	**299279.4 ± 3685.9 (3)**	-	-
F_7	**148067.7 ± 68996.3 (42)**	169019.16 ± 84155.5 (36)	175486.0 ± 80658.6 (37)

TABLE VIII
COMPARISON WITH OTHER RCMAS IN TERMS OF **ERROR** VALUES ($\mathrm{N} = 30$)

	DEahcSPX	MGG+UNDX	G3+PCX	MGG+SPX
F_{sph}	1.75E-31 ± 4.99E-31	1.37E-11 ± 1.94E-11	**3.58E-81 ± 1.36E-81** ‡	8.75E+00 ± 2.87E+00
F_{ros}	4.52E+00 ± 1.55E+01	2.81E+01 ± 1.23E+01	**4.18E+00 ± 9.68E+01**	1.38E+03 ± 6.45E+02
F_{ack}	**2.66E-15 ± 0.00E+00**	8.23E-07 ± 4.64E-07	1.48E+01 ± 4.17E+00 ‡	1.68E+00 ± 2.99E-01
F_{grw}	2.07E-03 ± 5.89E-03	**2.96E-04 ± 1.48E-03**	1.07E-02 ± 1.30E-02 ‡	1.09E+00 ± 2.24E-02
F_{ras}	2.14E+01 ± 1.23E+01	**1.35E+00 ± 1.03E+00**	1.75E+02 ± 3.37E+01 ‡	5.78E+00 ± 1.83E+00
F_{sch}	**4.70E+02 ± 2.96E+02**	4.12E+03 ± 1.72E+03	4.04E+03 ± 1.09E+03 ‡	8.70E+03 ± 2.41E+02
F_{sal}	1.80E-01 ± 4.08E-02	**1.50E-01 ± 4.95E-02**	4.64E+00 ± 4.74E+00 ‡	3.82E-01 ± 4.29E-02
F_{wht}	**3.06E+02 ± 1.10E+02**	4.28E+02 ± 3.82E+01	7.90E+02 ± 1.27E+02 ‡	3.28E+03 ± 2.77E+03
F_{pn1}	**2.07E-02 ± 8.46E-02**	4.93E-02 ± 3.50E-02	4.35E+00 ± 6.94E+00 ‡	2.57E-01 ± 6.90E-02
F_{pn2}	**1.71E-31 ± 5.35E-31**	4.39E-04 ± 2.20E-03	1.50E+01 ± 1.58E+01 ‡	2.29E+00 ± 3.72E-01
F_1	**0.00E+00 ± 0.00E+00**	2.83E-11 ± 3.33E-11	3.52E-13 ± 1.22E-13 ‡	4.71E+04 ± 4.21E+03
F_2	6.52E-05 ± 4.84E-05	1.41E+00 ± 7.15E-01	**4.14E-12 ± 1.21E-12** ‡	3.96E+04 ± 3.89E+03
F_3	1.29E+06 ± 9.22E+05	8.76E+05 ± 2.98E+05	**1.07E+03 ± 1.29E+03**	7.16E+08 ± 1.34E+08
F_4	**4.62E+00 ± 8.78E+00**	5.01E+01 ± 3.62E+01	9.35E+04 ± 2.66E+04	4.45E+04 ± 3.73E+03
F_5	**9.00E+02 ± 4.79E+02**	1.67E+03 ± 6.01E+02	8.13E+03 ± 2.65E+03 ‡	3.34E+04 ± 2.11E+03
F_6	**3.84E+00 ± 3.75E+00**	1.79E+02 ± 2.38E+02	1.34E+02 ± 2.48E+02	1.56E+10 ± 1.47E+09
F_7	7.39E-03 ± 6.32E-03	**7.26E-03 ± 8.19E-03**	2.01E-02 ± 1.85E-02 ‡	1.02E+04 ± 4.71E+02
F_8	2.09E+01 ± 1.12E-01	2.09E+01 ± 5.62E-02	2.11E+01 ± 6.67E-12 ‡	2.10E+01 ± 4.06E-02
F_9	**2.04E+01 ± 8.19E+00**	4.65E+01 ± 5.41E+01	2.44E+02 ± 3.98E+01 ‡	3.15E+02 ± 1.04E+01
F_{10}	5.27E+01 ± 4.84E+01	**4.76E+01 ± 5.03E+01**	3.89E+02 ± 9.96E+01 ‡	5.31E+02 ± 2.85E+01

‡ Algorithm converged before using the maximum allowed fitness evaluations

in each run was 100 000. From Table XI, it can be found that the performance of RCMA was better than DEahcSPX only for F_{Sph} and F_{Sch}, and in all other cases, the average performance of the proposed algorithm was better than that of RCMA. Hence, in an average, the DEahcSPX algorithm outperformed the RCMA on the studied benchmark and on the real-world problems.

Finally, we compared our proposed algorithm with the dynamic multiswarm particle swarm optimizer with LS (DMS-PSO) algorithm using the results reported in [47]. Table XII compares DMS-PSO, DE, and DEahcSPX for the ten benchmark functions used in our suite (F_1–F_{10}). The results are the average of 25 runs under the same experimental conditions. As shown in Table XII, for F_1, F_2, F_3, and F_{10}, DMS-PSO outperformed DEahcSPX. In particular, the performance of DMS-PSO was extraordinary for the first three unimodal functions. In contrast, for F_4, F_6, F_7, and F_9, DEahcSPX outperformed DMS-PSO considerably. For the other two functions F_5 and F_8, no performance difference was observed. The results of Table XII suggest that DMS-PSO is exceptional in solving unimodal problems, and can also handle multimodal problems competitively. On the other hand, DEahcSPX exhibited superior performance in solving multimodal functions compared with DMS-PSO.

In all of the above comparisons in Tables X–XII, DEahcSPX consistently exhibited superior performance compared with the original DE which establishes that AHCXLS scheme is equally suitable for the exponential crossover scheme.

TABLE IX
COMPARISON WITH OTHER RCMAS IN TERMS OF **FES** ($\mathrm{N} = 30$)

	DEahcSPX	MGG+UNDX	G3+PCX	MGG+SPX
F_{sph}	87027.4 ± 3967.3 (50)	200515.5 ± 6743.2 (50)	**2640.1 ± 104.9 (50)** ‡	-
F_{ros}	299913.0 ± 519.5 (2)	-	**177783.9 ± 71145.1 (34)** ‡	-
F_{ack}	**129211.6 ± 5168.6 (50)**	294226.7 ± 4359.5 (40)	-	-
F_{grw}	**121579.2 ± 79563.4 (43)**	238310.7 ± 13968.8 (42)	14560.6 ± 12576.4 (20) ‡	-
F_{ras}	-	299583.6 ± 2102.0 (2)	-	-
F_{pn1}	**96149.0 ± 61787.7 (46)**	185258.4 ± 7436.3 (44)	-	-
F_{pn2}	**85360.2 ± 6390.6 (50)**	209272.8 ± 19680.5 (48)	-	-
F_1	89417.8 ± 4117.6 (50)	206630.6 ± 5771.4 (50)	**2649.2 ± 121.3 (50)** ‡	-
F_2	299279.4 ± 3685.9 (3)	-	**13290.7 ± 569.1 (50)** ‡	-
F_6	-	-	**177617.4 ± 110147.4 (24)**	-
F_7	**148067.7 ± 68996.3 (42)**	257533.8 ± 34064.9 (35)	9314.4 ± 4281.8 (20) ‡	-

‡ Algorithm converged before using the maximum allowed fitness evaluations

TABLE X
COMPARISON WITH MA-S2 [19]

	Range	MA-S2	DE	DEahcSPX
F_{Sphere}	$[-5.12, 5.12]^{30}$	[Global min in 100%] 7198 Evals	4.27E-05 ± 1.18E-05	2.56E-06 ± 1.06E-06
$F_{Griewank}$	$[-600, 600]^{10}$	2.80E-04 ± 8.28E-04	3.44E-02 ± 2.05E-02	2.77E-02 ± 1.24E-02
F_{Bump}	$[0, 10]^{20}$	7.34E-01 ± 2.22E-02	0.7999 ± 0.0024	**8.02E-01 ± 3.87E-03**
$F_{ShekelsFoxHole}$	$[-65.536, 65.536]^2$	[Global min in 100%] 9634 Evals	0.998 ± 5.09E-16	0.998 ± 7.20E-17
$F_{Schwefel}$	$[-500, 500]^{10}$	[Global min in 80%]	1.27E-04 ± 3.73E-13	1.27E-04 ± 0.00E+00
$F_{Rosenbrock}$	$[-2.048, 2.048]^{30}$	2.57E+04 ± 6.00E+01	2.44E+01 ± 7.32E-01	**2.15E+01 ± 7.35E-01**
F_{Step}	$[-5.12, 5.12]^{30}$	1.63E-01 ± 3.10E-02	[13967.7 ± 738.07]†	**[13846.6 ± 621.95]†**
$F_{Rastrigin}$	$[-5.12, 5.12]^{20}$	1.55E-01 ± 7.10E-02	2.14E-01 ± 1.53E-01	**6.02E-02 ± 1.00E-01**

† Global optimum 0.0 found using these FEs counts

TABLE XI
COMPARISON WITH RCMA [17]

	Range	RCMA-XHC		DE		DEahcSPX	
		A	B	A	B	A	B
F_{Sph}	$[-5.12, 5.12]^{25}$	**6.50E-101**	**1.10E-105**	6.91E-18	8.63E-19	2.58E-20	6.16824E-21
F_{Ros}	$[-5.12, 5.12]^{25}$	2.20E+00	**6.00E-04**	1.32E-01	2.48E-02	**1.20E-02**	1.13E-03
F_{Sch}	$[-65.536, 65.536]^{25}$	**3.80E-07**	**4.50E-09**	1.47E+01	5.53E+00	2.48E-01	4.98E-02
F_{Ras}	$[-5.12, 5.12]^{25}$	1.40E+00	**32%**	6.59E-11	6.09E-12	**3.48404E-12**	6.75016E-14
F_{Gri}	$[-600, 600]^{25}$	1.30E-02	**30%**	1.48E-04	1.54E-14	**3.75566E-14**	18%
P_{sle}	$[-127, 127]^{10}$	5.50E+01	7.90E-01	8.25E-04	1.98E-05	**6.17E-04**	**8.76E-08**
P_{Cheb}	$[-6.4, 6.35]^6$	1.40E+02	9.20E+00	**0.00E+00**	**100%**	**0.00E+00**	**100%**
P_{fms}	$[-512, 512]^9$	7.70E+00	**40%**	2.68E+00	1.10E-23	**2.22E+00**	1.43E-04

A: Average of the minimum fitness found; B: best of all minimum fitness or percentage of run that found global optimum

H. Other Studies of AHCXLS Scheme

In all experiments, we fixed $F = 0.9$ and $C_r = 0.9$ as the parameter setting for all algorithms. As mentioned earlier, because of the sensitivity of DE to its control parameter, some variants with adaptive control parameter have been proposed [24], [27], [36]. In order to show that the proposed AHCXLS scheme can also accelerate such adaptive DE variants, we incorporated it in a recent DE variant with self-adaptive control parameters (DESP), proposed in [27]. We call the new variant DESPahcSPX. The comparative results (average of 25 runs) with the same settings as in [27] ($\mathrm{N} = 30$ and $\mathrm{P} = 100$) are reported in Table XIII. The results of Table XIII suggest that integration of AHCXLS in DESP has certainly accelerated the algorithm. These results also indicate that the acceleration of DE by AHCXLS scheme is not influenced by the parameter settings. Hence, the AHCXLS scheme can be similarly useful for performance enhancement of other self-adaptive DE variants.

The only parameter AHCXLS scheme includes is n_p, the number of parents participating in the crossover operation. The

TABLE XII
COMPARISON WITH DMS-PSO [47] AT $N = 30$

	DMS-PSO	DE	DEahcSPX
F_1	**[5026.3 ± 72.463]** †	[64546.85 ± 1097.25] †	[54959.52 ± 1184.39] †
F_2	**[125520 ± 17371]** †	1.33E-02 ± 6.72E-03	4.87E-05 ± 2.74E-05
F_3	**1.63E-06 ± 3.92E-06 (84%)**	1.85E+06 ± 1.05E+06	9.80E+05 ± 5.56E+05
F_4	2.55E+03 ± 3.06E+02	9.16E+01 ± 4.53E+01	**1.14E+00 ± 6.16E-01**
F_5	2.19E+03 ± 8.26E+02	3.27E+03 ± 8.75E+02	2.15E+03 ± 7.04E+02
F_6	4.78E-01 ± 1.32E+00 (98%)	[161762.1 ± 3765.823] †	**[146963.32 ± 5613.87]** †
F_7	7.00E-03 ± 4.54E-03 (96%)	7.96E-02 ± 5.99E-02 (16%)	**5.64E-04 ± 2.03E-03 (98%)**
F_8	2.00E+01 ± 2.30E-04	2.09E+01 ± 5.35E-02	2.10E+01 ± 5.52E-02
F_9	1.76E+01 ± 3.02E+00	[60441.55 ± 2015.97] †	**[57163.12 ± 1879.38]** †
F_{10}	**3.74E+01 ± 5.29E+00**	1.20E+02 ± 1.42E+01	9.51E+01 ± 1.45E+01

† Target accuracy level achieved using these FEs counts

TABLE XIII
STUDY ON THE SUITABILITY OF AHCXLS FOR DESP

	DESP	DESPahcSPX		DESP	DESPahcSPX
F_{sph}	[50808.08 ± 1178.07] †	**[40956.12 ± 967.89]** †	F_1	[50269.24 ± 1183.61] †	**[41245.92 ± 1167.10]** †
F_{ros}	1.42E+01 ± 1.61E+01	**1.30E+01 ± 1.72E+01**	F_2	6.86E-07 ± 5.81E-07 (80%)	**2.83E-07 ± 3.94E-07 (92%)**
F_{ack}	[71779.96 ± 1507.12] †	**[58920.68 ± 1316.83]** †	F_3	1.89E+05 ± 1.04E+05	**1.74E+05 ± 1.04E+05**
F_{grw}	[53679.6 ± 3113.15] †	**[43412.68 ± 2199.17]** †	F_4	3.74E-02 ± 8.86E-02	**2.11E-02 ± 3.17E-02**
F_{ras}	[108502.52 ± 4027.63] †	**[102456.6 ± 4775.19]** †	F_5	4.25E+02 ± 4.07E+02	**3.78E+02 ± 4.37E+02**
F_{sch}	3.82E-04 ± 0.00E+00	3.82E-04 ± 0.00E+00	F_6	2.60E+01 ± 2.79E+01	**2.20E+01 ± 2.55E+01**
F_{sal}	1.92E-01 ± 2.77E-02	**1.36E-01 ± 4.90E-02**	F_7	1.60E-02 ± 1.17E-02 (52%)	**1.28E-02 ± 8.51E-03 (76%)**
F_{wht}	3.47E+01 ± 5.88E+01	**2.83E+01 ± 5.60E+01 (4%)**	F_8	2.09E+01 ± 3.66E-02	2.09E+01 ± 5.90E-02
F_{pn1}	[44659.36 ± 1361.99] †	**[36064.04 ± 1181.63]** †	F_9	[83861.84 ± 2510.90] †	**[81091.8 ± 3372.61]** †
F_{pn2}	[49224.72 ± 1224.12] †	**[39508.88 ± 1035.98]** †	F_{10}	5.71E+01 ± 9.83E+00	**5.35E+01 ± 8.13E+00**

† Target accuracy level achieved using these FEs counts

authors of SPX operation suggested that the number of parents should be 3 or 4 [4], and hence we used $n_p = 3$ in this study. However, we studied the effect of n_p on the performance using $n_p = 4, 5, 6, 8, 10, 12$, and 15. Table XIV compares the performance for some of the choices. From Table XIV, it seems that in general, a higher number of parents n_p can slightly improve the performance of the algorithm. However, the effect should be studied in more detail by varying population size and problem dimension which is out of the scope of this research.

Another issue in the AHCXLS scheme is the selection of parents other than the best individual of the generation. In this work, we have chosen them randomly. However, incorporating the knowledge obtained during the search in selecting parents (other than the best) for SPX operation can further improve the performance. We briefly studied the effect of positive assortative mating (PAM) and negative assortative mating (NAM) on the algorithm performance. After selecting the first parent, PAM (NAM) selects other individuals with most (least) phenotype similarity [48]. Here, we used Euclidean distance between chromosomes as a measure of similarity between them. The results shown in Table XV suggest that NAM can be useful to improve the performance of the algorithm, mostly for unimodal functions. However, considering the performance improvement achieved and the additional computational cost incurred in NAM, the random mating used in this work can be used as a computationally less expensive approach. Many other sophisticated mechanisms available can be used for getting online feedback from the search that can help to improve the quality of the local tuning at the expense of some computational effort [20]–[22].

VI. DISCUSSION

Most of the real-world problems involve variables in continuous domain and thereby need fine tuning of these variables. However, traditional GAs are often not efficient for fine-tuning solutions close to optimum [49]. A more competitive form of algorithm can be obtained through intelligent incorporation of local improvement processes in EAs. Traditionally, these hybrid EAs or MAs have been implemented by incorporating problem dependent heuristics for refining individuals (i.e., improving their fitness through fine tuning). However, the field of EA has always enjoyed the superior characteristic of being problem independent. Therefore, a recent interest is to include the LS in EA in a problem independent manner. The availability of many sophisticated crossover operations, that are capable of generating offspring according to the distribution of

TABLE XIV
STUDY OF n_p FOR AHCXSL OPERATION (N = 30)

	$n_p = 5$	$n_p = 8$	$n_p = 10$	$n_p = 15$
F_{sph}	[83198.68 ± 4951.30] †	[82987.64 ± 3546.02] †	[80547.52 ± 3112.62] †	**[77132.88 ± 4603.41]** †
F_{ros}	2.23E+00 ± 2.27E+00	1.56E+00 ± 2.44E+00	1.87E+00 ± 2.11E+00	**1.49E+00 ± 2.15E+00**
F_{ack}	[124043.36 ± 5526.48] †	4.62E-02 ± 2.31E-01 (96%)	[121583.2 ± 5885.61] †	**[116291.16 ± 4177.22]** †
F_{grw}	2.76E-03 ± 4.73E-03 (72%)	**1.58E-03 ± 4.91E-03 (88%)**	3.44E-03 ± 7.99E-03 (76%)	2.86E-03 ± 4.48E-03 (68%)
F_{ras}	2.09E+01 ± 6.72E+00	1.82E+01 ± 5.42E+00	1.80E+01 ± 5.47E+00	**1.73E+01 ± 6.32E+00**
F_{sch}	5.50E+02 ± 3.11E+02	5.26E+02 ± 3.98E+02	**4.91E+02 ± 2.20E+02**	5.73E+02 ± 2.97E+02
F_{sal}	1.72E-01 ± 8.91E-02	1.61E-01 ± 4.90E-02	**1.50E-01 ± 5.00E-02**	1.54E-01 ± 5.72E-02
F_{wht}	3.03E+02 ± 1.00E+02	3.13E+02 ± 1.02E+02	3.17E+02 ± 9.11E+01	**2.81E+02 ± 1.07E+02**
F_{pn1}	1.66E-02 ± 3.88E-02 (84%)	**8.29E-03 ± 4.15E-02 (96%)**	2.49E-02 ± 1.05E-01 (92%)	2.49E-02 ± 8.61E-02 (88%)
F_{pn2}	6.54E-02 ± 3.18E-01 (80%)	1.45E-01 ± 7.19E-01 (88%)	1.44E-01 ± 7.19E-01 (92%)	**8.79E-04 ± 3.04E-03 (92%)**
F_1	[84284 ± 3930.25] †	[81488.92 ± 3881.04] †	[79089.92 ± 3227.048] †	**[78427.72 ± 4491.97]** †
F_2	2.36E-05 ± 3.78E-05	**1.84E-05 ± 2.73E-05 (4%)**	3.62E-05 ± 3.17E-05	2.35E-05 ± 2.88E-05
F_3	3.91E+05 ± 1.93E+05	**3.65E+05 ± 2.03E+05**	4.17E+05 ± 2.44E+05	4.05E+05 ± 1.80E+05
F_4	5.36E-01 ± 5.40E-01	**5.01E-01 ± 7.05E-01**	5.98E-01 ± 7.76E-01	1.38E+00 ± 3.60E+00
F_5	8.20E+01 ± 1.31E+02	1.19E+02 ± 1.76E+02	7.41E+01 ± 8.41E+01	**6.98E+01 ± 7.52E+01**
F_6	1.55E+00 ± 2.49E+00 (20%)	9.37E-01 ± 1.56E+00 (12%)	**1.65E+00 ± 3.22E+00 (24%)**	7.18E-01 ± 1.16E+00 (16%)
F_7	6.89E-03 ± 9.63E-03 (76%)	4.63E-03 ± 6.45E-03 (84%)	**3.16E-03 ± 4.90E-03 (92%)**	4.83E-03 ± 7.22E-03 (92%)
F_8	2.10E+01 ± 4.69E-02	2.09E+01 ± 4.26E-02	2.09E+01 ± 4.18E-02	2.09E+01 ± 4.29E-02
F_9	1.98E+01 ± 6.45E+00	**1.86E+01 ± 4.91E+00**	1.98E+01 ± 6.95E+00	1.99E+01 ± 8.55E+00
F_{10}	5.22E+01 ± 6.11E+01	5.96E+01 ± 6.69E+01	5.79E+01 ± 6.67E+01	**4.55E+01 ± 5.69E+01**

† Target accuracy level achieved using these FEs counts

TABLE XV
COMPARISON WITH DIFFERENT MATING SELECTION MECHANISMS FOR THE SPX OPERATION IN DEAHCSPX

	PAM	NAM		PAM	NAM
F_{sph}	[71637.24 ± 3602.63] †	**[57435.96 ± 3656.022]** †	F_1	[93801.62 ± 4809.75] †	**[78479.86 ± 4113.097]** †
F_{ros}	1.16E+00 ± 1.53E+00	**2.55E-01 ± 7.90E-01**	F_2	2.27E-04 ± 2.59E-04	**9.34E-05 ± 1.53E-04**
F_{ack}	[139184.34 ± 6239.022] †	**[118600.28 ± 5720.94]** †	F_3	**4.38E+05 ± 2.52E+05**	5.16E+05 ± 2.27E+05
F_{grw}	4.87E-03 ± 9.30E-03 (66%)	**1.72E-03 ± 5.19E-03 (88%)**	F_4	6.51E+00 ± 3.38E+01	**7.74E-01 ± 1.05E+00**
F_{ras}	2.38E+01 ± 7.81E+00	**1.88E+01 ± 6.08E+00**	F_5	1.52E+02 ± 1.73E+02	**5.14E+01 ± 5.60E+01**
F_{sch}	6.28E+02 ± 3.82E+02	**5.10E+02 ± 2.94E+02**	F_6	4.78E+00 ± 1.06E+01 (10%)	**2.07E+00 ± 2.39E+00 (10%)**
F_{sal}	2.45E-01 ± 5.71E-02	**1.90E-01 ± 3.64E-02**	F_7	9.50E-03 ± 9.31E-03 (33%)	**5.42E-03 ± 7.71E-03 (78%)**
F_{wht}	**2.80E+02 ± 1.11E+02**	3.39E+02 ± 6.35E+01	F_8	2.09E+01 ± 5.37E-02	2.09E+01 ± 5.80E-02
F_{pn1}	5.62E-02 ± 2.00E-01 (86%)	**6.22E-03 ± 2.49E-02 (94%)**	F_9	2.11E+01 ± 6.01E+00	**1.97E+01 ± 5.62E+00**
F_{pn2}	8.79E-04 ± 3.01E-03 (92%)	**4.39E-04 ± 2.17E-03 (96%)**	F_{10}	8.65E+01 ± 7.53E+01	**7.62E+01 ± 7.42E+01**

† Target accuracy level achieved using these FEs counts

the parents, has opened the door for designing such problem independent LS process. By producing offspring densely around the parents, these crossover operators are capable of exploring the neighborhood of the parents in the continuous search space. Taking advantage of this characteristic, some very successful EAs have been designed.

Success of a MA depends considerably on balancing of LS and global search capability [38]. However, the depth of the LS, best suited for the exploration of an individual's neighborhood, is essentially problem dependent and even varies with the problem dimension or with the progress of the global search (i.e., the EA under consideration). Therefore, it is very difficult to find a single length for the LS that performs best for all sort of problems in all dimensions, and such problem dependent tuning of the LS length is not easy and makes the LS heuristics problem dependent again. In an attempt to design a completely problem independent crossover-based LS process, we proposed the AHCXLS scheme in this work. The AHCXLS scheme was designed by borrowing concepts from both LLS and XLS, to take the advantages of both paradigms.

DE is one of the most prominent new generation EAs for global optimization over continuous spaces. The intense research in the field of DE has shown that the algorithm can be improved in many different ways. Therefore, in this work, we attempt to accelerate DE algorithm using the AHCXLS process. Since we want to increase the convergence velocity

of DE without sacrificing the convergence probability, it is safe to allow some additional fitness evaluations to explore the neighborhood of the most promising individuals. Therefore, we applied AHCXLS on the best individual of the generation in the proposed DEahcSPX algorithm.

In order to study the performance of the proposed algorithm, we experimented using a test suite consisting of functions with different characteristics. The size and the diversity of the test suite is adequate enough to make a general conclusion about the performance of the algorithms. In different experimental results presented in this paper, the proposed DEahcSPX outperformed the classic DE algorithm. The speedup of the algorithm has been also demonstrated. The scalability study and the population size study highlighted the robustness of the proposed algorithm over the original DE algorithm. Different experimental results and comparison with other MAs show that the performance of DEahcSPX is superior to, or at least comparable to, many of the state-of-the-art EAs, particularly for multimodal problems, but it can also deal with the unimodal problems very competitively.

Generally, incorporation of a LS cannot modify the overall behavior of an algorithm; but can improve some of its characteristics. More or less the same phenomenon was observed in the case of DEahcSPX. From different experimental results, and from the shape of the convergence graphs, it was found that for a particular class of problem, the proposed memetic version of DE behaves like its parent algorithm. However, in almost every case it exhibited a higher convergence velocity compared with DE.

We compared the proposed AHCXLS with other XLS applying in DE and showed that the newly proposed LS scheme performs better. We hypothesize that the adaptive nature of the AHCXLS guides the algorithm to explore the neighborhood of each individual most effectively and locate the global optimum at a minimum cost. Furthermore, the scheme sets us free from the search for the best length for LS.

The principle of AHCXLS is so simple and generalized that it can be hybridized with any of the newly proposed DE variants without increasing the algorithm complexity, and in a brief study, we found that AHCXLS scheme can accelerate some other variants of basic DE algorithm proposed by Storn and Price. Experimental results also showed the potential of the AHCXLS scheme in accelerating the self-adaptive variants of DE.

In our experiments, we used SPX as the crossover operation in the proposed XLS because it is one of the elite crossover operations with adaptive quality. We also experimented with other crossover operations like UNDX [3], PCX [45], BLX-α [46], parent centric BLX-α (PBX-α) [17], and we found that the performance of those XLS were influenced by the adaptive capability of the crossover schemes. For more sophisticated crossover schemes, such as SPX, UNDX, and PCX, the performance of the XLS was better than the others. This reestablishes that crossover operations with adaptive capability can be used for the exploration of the neighborhood of an individual and our AHCXLS scheme used this characteristic of the SPX operation for modeling a successful LS scheme for DE algorithm.

VII. Conclusion

DE is a reliable and versatile function optimizer over continuous search spaces. Due to its simple and compact structure, ease of implementation and use of few parameters, DE has already seen many real-world applications in various fields such as pattern recognition, communication, mechanical and chemical engineering, and biotechnology. In real-world applications, the optimization algorithm should be able to locate the global minimum using as few fitness evaluations as possible, because the fitness evaluation is often the most expensive part of the search. Therefore, in an attempt to accelerate the classic DE algorithm, we proposed an adaptive crossover-based LS in this work.

We investigated the performance of the proposed version of the DE algorithm using a benchmark suite consisting of functions carefully chosen from the literature. The experimental results showed that the proposed algorithm outperforms the classic DE in terms of convergence velocity in all experimental studies. The overall performance of the adaptive LS scheme was better than the other crossover-based LS strategies and the overall performance of the newly proposed DE algorithm was superior to or at least competitive with some other MAs selected from literature. The proposed LS scheme was also found prospective for adaptive DE variants. We hope that this work will encourage further research into the self-adaptability of DE.

In our future study, we will apply the proposed algorithm to solve some real-world problems. We will also want to verify the potential of the adaptive LS algorithm for other EAs.

Appendix I
Benchmark Functions

The test suite that we have used for different experiments consists of 20 benchmark functions. The first ten test functions of the suite are functions commonly found in the literature and the other benchmarks are the first ten functions from the newly defined test suite for CEC 2005 Special Session on real-parameter optimization [41]. Our test suite was as follows.

1) F_{sph}: Sphere Function.
2) F_{ros}: Rosenbrock's Function.
3) F_{ack}: Ackley's Function.
4) F_{grw}: Griewank's Function.
5) F_{ras}: Rastrigin's Function.
6) F_{sch}: Generalized Schwefel's Problem 2.26.
7) F_{sal}: Salomon's Function.
8) F_{wht}: Whitely's Function.
9) F_{pn1}: Generalized Penalized Function 1.
10) F_{pn2}: Generalized Penalized Function 2.
11) F_1: Shifted Sphere Function.
12) F_2: Shifted Schwefel's Problem 1.2.
13) F_3: Shifted Rotated High Conditioned Elliptic Function.
14) F_4: Shifted Schwefel's Problem 1.2 With Noise in Fitness.
15) F_5: Schwefel's Problem 2.6 With Global Optimum on Bounds.
16) F_6: Shifted Rosenbrock's Function.
17) F_7: Shifted Rotated Griewank's Function Without Bounds.
18) F_8: Shifted Rotated Ackley's Function With Global Optimum on Bounds.
19) F_9: Shifted Rastrigin's Function
20) F_{10}: Shifted Rotated Rastrigin's Function.

Definitions of the first ten functions are as follows:

$$F_{\text{sph}}(\vec{x}) = \sum_{i=1}^{N} x_i^2; \quad -100 \le x_i \le 100;$$

$$F^*_{\text{sph}} = F_{\text{sph}}(0,\ldots,0) = 0$$

$$F_{\text{ros}}(\vec{x}) = \sum_{i=1}^{N-1} \left(100\left(x_{i+1} - x_i^2\right)^2 + (1-x_i)^2\right); \quad -100 \le x_i \le 100;$$

$$F^*_{\text{ros}} = F_{\text{ros}}(1,\ldots,1) = 0$$

$$F_{\text{ack}}(\vec{x}) = 20 + exp(1) - 20exp\left(-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}\right) - exp\left(\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi x_i)\right); \quad -32 \le x_i \le 32; \quad F^*_{\text{ack}} = F_{\text{ack}}(0,\ldots,0) = 0$$

$$F_{\text{grw}}(\vec{x}) = \sum_{i=1}^{N} \frac{x_i^2}{4000} - \prod_{i=1}^{N} \cos\frac{x_i}{\sqrt{i}} + 1; \quad -600 \le x_i \le 600; \quad F^*_{\text{grw}} = F_{\text{grw}}(0,\ldots,0) = 0$$

$$F_{\text{ras}}(\vec{x}) = 10N + \sum_{i=1}^{N}(x_i^2 - 10\cos(2\pi x_i)); \quad -5 \le x_i \le 5; \quad F^*_{\text{ras}} = F_{\text{ras}}(0,\ldots,0) = 0$$

$$F_{\text{sch}}(\vec{x}) = 418.9829N - \sum_{i=1}^{N}(x_i \sin(\sqrt{|x_i|})); \quad -500 \le x_i \le 500;$$

$$F^*_{\text{sch}} = F_{\text{sch}}(420.9687,\ldots,420.9687) = 0$$

$$F_{\text{sal}}(\vec{x}) = -cos\left(2\pi\sqrt{\sum_{i=1}^{N} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{N} x_i^2} + 1; \quad -100 \le x_i \le 100;$$

$$F^*_{\text{sal}} = F_{\text{sal}}(0,\ldots,0) = 0$$

$$F_{\text{wht}}(\vec{x}) = \sum_{j=1}^{N}\sum_{i=1}^{N}\left(\frac{y_{i,j}^2}{4000} - \cos(y_{i,j}) + 1\right); \quad \text{where } y_{i,j} = 100\left(x_j - x_i^2\right)^2 + (1-x_i)^2$$

$$-100 \le x_i \le 100; \quad F^*_{\text{wht}} = F_{\text{wht}}(1,\ldots,1) = 0$$

$$F_{\text{pn1}}(\vec{x}) = \frac{\pi}{N}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{N-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_N - 1)^2\right\} + \sum_{i=1}^{N} u(x_i, 10, 100, 4);$$

where $y_i = 1 + \frac{1}{4}(x_i + 1)$ and

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a \le x_i \le a, \\ k(-x_i - a)^m, & x_i < -a. \end{cases}$$

$$-50 \le x_i \le 50; \quad F^*_{\text{pn1}} = F_{\text{pn1}}(-1,\ldots,-1) = 0$$

$$F_{\text{pn2}}(\vec{x}) = 0.1\left\{\sin^2(\pi 3x_1) + \sum_{i=1}^{N-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_N - 1)^2[1 + \sin^2(2\pi x_N)]\right\} + \sum_{i=1}^{N} u(x_i, 5, 100, 4);$$

$$-50 \le x_i \le 50; \quad F^*_{\text{pn2}} = F_{\text{pn2}}(1,\ldots,1) = 0$$

Functions F_1–F_{10} are designed by modifying classical benchmark functions to test the optimizer's ability to locate a global optimum under a variety of circumstances such as translated and/or rotated landscape, optimum placed on bounds, Gaussian noise, and/or bias added, etc. [41]. A complete definition of these functions are available online at http://www.ntu.edu.sg/home/epnsugan and in [41], and a more detailed description of the other functions can be found in [23] and [50]. In our test suit, F_1 to F_5, F_{sph}, and F_{ros} are unimodal and the rest are multimodal functions. All the chosen benchmarks are minimization problems.

ACKNOWLEDGMENT

The authors are grateful to the anonymous associate editor and the anonymous referees for their constructive comments and helpful suggestions to improve the quality of this paper.

REFERENCES

[1] T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds., *Evolutionary Computation 2: Advanced Algorithms and Operators*. Bristol, U.K.: Institute of Physics, 2000.

[2] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, Jun. 2001.

[3] I. Ono, H. Kita, and S. Kobayashi, *Advances in Evolutionary Computing*. New York: Springer, Jan. 2003, ch. A Real-Coded Genetic Algorithm Using the Unimodal Normal Distribution Crossover, pp. 213–237.

[4] S. Tsutsui, M. Yamamura, and T. Higuchi, "Multi-parent recombination with simplex crossover in real coded genetic algorithms," in *Proc. Genetic Evol. Comput. Conf. (GECCO'99)*, Jul. 1999, pp. 657–664.

[5] R. Storn and K. V. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Opt.*, vol. 11, no. 4, pp. 341–359, Dec. 1997.
[6] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, Dec. 1995, pp. 1942–1948.
[7] B. Freisleben and P. Merz, "A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems," in *Proc. IEEE Int. Conf. Evol. Comput.*, 1996, pp. 616–621.
[8] P. Merz and B. Freisleben, "Fitness landscapes, memetic algorithms, and greedy operators for graph bipartitioning," *Evol. Comput.*, vol. 8, no. 1, pp. 61–91, 2000.
[9] P. Moscato and M. G. Norman, "A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems," in *Proc. Parallel Comput. Transputer Appl.*, 1992, pp. 177–186.
[10] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin, Germany: Springer-Verlag, 1996.
[11] Y. Jin, Ed., *Knowledge Incorporation in Evolutionary Computation*, ser. Studies in Fuzziness and Soft Computing. Berlin, Germany: Springer-Verlag, 2005, vol. 167.
[12] L. Davis, *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold, 1991.
[13] D. Goldberg and S. Voessner, "Optimizing global-local search hybrids," in *Proc. Genetic Evol. Comput. Conf. (GECCO'99)*, 1999, pp. 220–228.
[14] R. G. Reynolds, "An introduction to cultural algorithms," in *Proc. 3rd Ann. Conf. Evol. Program.*, 1994, pp. 131–139.
[15] R. G. Reynolds, "Cultural algorithms: Computational modeling of how cultures learn to solve problems: An engineering example," *Cybern. Syst.*, vol. 36, no. 8, pp. 753–771, 2005.
[16] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms," Caltech Concurrent Computation Program, California Inst. Technol., Pasadena, CA, Tech. Rep. 826, 1989.
[17] M. Lozano, F. Herrera, N. Krasnogor, and D. Molina, "Real-coded memetic algorithms with crossover hill-climbing," *Evol. Comput.*, vol. 12, no. 3, pp. 273–302, 2004.
[18] H. G. Beyer and K. Deb, "On self-adaptive features in real-parameter evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 5, no. 3, pp. 250–270, 2001.
[19] Y.-S. Ong and A. J. Keane, "Meta-Lamarckian learning in memetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 99–110, 2004.
[20] Y.-S. Ong, M.-H. Lim, N. Zhu, and K.-W. Wong, "Classification of adaptive memetic algorithms: A comparative study," *IEEE Trans. Syst., Man, Cybern.—Part B*, vol. 36, no. 1, pp. 141–152, 2006.
[21] N. K. Bambha, S. S. Bhattacharyya, J. Teich, and E. Zitzler, "Systematic integration of parameterized local search into evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 137–155, 2004.
[22] N. Krasnogor and J. Smith, "A memetic algorithm with self-adaptive local search: TSP as a case study," in *Proc. Genetic Evol. Comput. Conf.*, 2000, pp. 987–994.
[23] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Berlin, Germany: Springer-Verlag, 2005.
[24] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing—A Fusion of Foundations, Methodologies and Applications*, vol. 9, no. 6, pp. 448–642, 2005.
[25] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2005, pp. 1785–1791.
[26] H. Y. Fan and J. Lampinen, "A trigonometric mutation operation to differential evolution," *J. Global Opt.*, vol. 27, no. 1, pp. 105–129, Sep. 2003.
[27] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 646–657, 2006.
[28] N. Noman and H. Iba, "Enhancing differential evolution performance with local search for high dimensional function optimization," in *Proc. 2005 Conf. Genetic Evol. Comput.*, Jun. 2005, pp. 967–974.
[29] R. Storn, "System design by constraint adaptation and differential evolution," *IEEE Trans. Evol. Comput.*, vol. 3, no. 1, pp. 22–34, Apr. 1999.
[30] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. C. Coello, "A comparative study of differential evolution variants for global optimization," in *Proc. Genetic Evol. Comput. Conf. (GECCO 2006)*, Jul. 2006, pp. 485–492.
[31] J. Sun, Q. Zhang, and E. P. Tsang, "DE/EDE: A new evolutionary algorithm for global optimization," *Inf. Sci.*, vol. 169, pp. 249–262, 2005.
[32] W.-J. Zhang and X.-F. Xie, "DEPSO: Hybrid particle swarm with differential evolution operator," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2003, pp. 3816–3821.
[33] S. Das, A. Konar, and U. K. Chakraborty, "Improving particle swarm optimization with differentially perturbed velocity," in *Proc. Genetic Evol. Comput. Conf. (GECCO)*, Jun. 2005, pp. 177–184.
[34] N. Noman and H. Iba, "A new generation alternation model for differential evolution," in *Proc. Genetic Evol. Comput. Conf. (GECCO 2006)*, Jul. 2006, pp. 1265–1272.
[35] R. Gämperle, S. D. Müller, and P. Koumoutsakos, "A parameter study for differential evolution," in *Proc. WSEAS Int. Conf. Advances Intell. Syst., Fuzzy Syst., Evol. Comput.*, 2002, pp. 293–298.
[36] D. Zaharie, "Critical values for the control parameters of differential evolution algorithms," in *Proc. MENDEL 8th Int. Conf. Soft Comput.* , 2002, pp. 62–67.
[37] S. Das, A. Konar, and U. K. Chakraborty, "Two improved differential evolution schemes for faster global search," in *Proc. Genetic Evol. Comput. Conf. (GECCO)*, Jun. 2005, pp. 991–998.
[38] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 204–223, 2003.
[39] J.-M. Yang and C.-Y. Kao, "Integrating adaptive mutations and family competition into genetic algorithms as function optimizer," *Soft Comput.*, vol. 4, pp. 89–102, 2000.
[40] T. Bäck, "Introduction to the special issue: Self-adaptation," *IEEE Trans. Evol. Comput.*, vol. 9, no. 2, pp. iii–iv, 2001.
[41] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, "Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization," Nanyang Technol. Univ., Singapore, IIT Kanpur, India, KanGAL Rep. 2005005, May 2005.
[42] T. Krink, B. Filipič, G. B. Fogel, and R. Thomsen, "Noisy optimization problems—A particular challenge for differential evolution?," in *Proc. Congr. Evol. Comput. (CEC 2004)*, Jun. 2004, pp. 332–339.
[43] J. Rönkkönen, S. Kukkonen, and K. Price, "Real-parameter optimization with differential evolution," in *Proc. 2005 IEEE Congr. Evol. Comput.*, Sep. 2005, pp. 506–513.
[44] H. Satoh, M. Yamamura, and S. Kobayashi, "Minimal generation gap model for GAs considering both exploration and exploitation," in *Proc. IIZUKA'96*, 1996, pp. 494–497.
[45] K. Deb, A. Anand, and D. Joshi, "A computationally efficient evolutionary algorithm for real-parameter optimization," *Evol. Comput.*, vol. 10, no. 4, pp. 371–395, 2002.
[46] L. J. Eshelman and J. D. Schaffer, *Foundations of Genetic Algorithms 2*. San Mateo, CA: Morgan Kaufmann, 1993, ch. Real-Coded Genetic Algorithms and Interval Schemata, pp. 187–202.
[47] J. J. Liang and P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer with local search," in *Proc. IEEE Congr. Evol. Comput.*, 2005, pp. 522–528.
[48] C. Fernandes and A. Rosa, "A study on non-random mating and varying population size in genetic algorithms using a royal road function," in *Proc. Congr. Evol. Comput. (CEC 2001)*, 2001, pp. 60–66.
[49] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[50] X. Yao, Y. Liu, and G. Liu, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, 1999.

Nasimul Noman received the B.Sc. and M.Sc. degrees in computer science from the University of Dhaka, Dhaka, Bangladesh. He is currently working towards the Ph.D. degree in frontier informatics at the Graduate School of Frontier Sciences, University of Tokyo, Tokyo, Japan.

He is a faculty member in the Department of Computer Science and Engineering, University of Dhaka, since March 2002. His research interests include evolutionary computation and bioinformatics.

Hitoshi Iba (M'99) received the Ph.D. degree from the University of Tokyo, Tokyo, Japan, in 1990.

From 1990 to 1998, he was with the ElectroTechnical Laboratory (ETL), Ibaraki, Japan. He has been with the University of Tokyo, since April 1998. He is currently a Professor at the Graduate School of Frontier Sciences, University of Tokyo. His research interest includes evolutionary computation, genetic programming, bioinformatics, foundation of artificial intelligence, machine learning, and robotics.

Dr. Iba is an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the *Journal of Genetic Programming and Evolvable Machines* (GPEM).