Parameter Control by the Entire Search History: Case Study of History-driven Evolutionary Algorithm

Shing Wa Leung, Shiu Yin Yuen and Chi Kin Chow

***Abstract*—History-driven Evolutionary Algorithm (HdEA) is an EA that uses the entire search history to improve searching performance. By building the approximated fitness landscape and estimating the gradient using the entire history, HdEA performs a parameter-less adaptive mutation. In order to decrease the number of parameters that makes the HdEA more robust, this paper proposes a novel adaptive parameter control system. This system is as an add-on component to HdEA, which uses the whole search history in HdEA to control the parameters in an automatic manner. The performance of the proposed system is examined on 34 benchmark functions. The results shows that the parameter control system gives similar or better performance in 24 functions and has the benefit that two parameters of the HdEA are eliminated; they are set and varied automatically by the system.**

I. Introduction

History Driven Evolutionary Algorithm (HdEA) is an evolutionary algorithm proposed by Chow and Yuen [1]. By implementing the Binary Search Partitioning (BSP) tree to store the entire search history, HdEA gives a non-parametric fitness landscape approximation of the objective function and performs a gradient-descent-like mutation, which is adaptive and parameter-less. The performance of HdEA has been tested on 34 well known benchmark functions with dimensions ranging from 2 to 40. HdEA ranks 1st compared with eight benchmark Evolutionary Algorithms (EA), which including real coded GA, differential evolution (DE), two improved DE, covariance matrix adaptation evolution strategy (CMA-ES), two improved particle swarm optimization (PSO), and estimation of distribution algorithm (EDA).

There are three main differences of HdEA compared with other EA. They are listed below:

The work described in this paper was supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China [Project No. CityU 124409]. The first author was also supported by a Postgraduate Research Studentship of City University of Hong Kong.

Shing Wa Leung is with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong SAR, China. E-mail: shinleung8@student.cityu.edu.hk

Shiu Yin Yuen is with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong SAR, China. E-mail: kelviny.ee@cityu.edu.hk

Chi Kin Chow is with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong SAR, China. E-mail: chowchi@cityu.edu.hk

1) Like the Non-revisiting Genetic Algorithm (NrGA) [2], it stores and uses the entire history to improve the performance.
2) It uses only a few parameters. Except the two main parameters, namely, population size and number of generations, it only needs to define the crossover (with its attendant probability) and selection operation.
3) When the HdEA is running, more and more chromosomes will be evaluated, thus the fitness landscape will becomes finer to give better approximation. This can be seen as an incremental learning process.

Thanks to fitness landscape approximation, the HdEA can estimates the fitness of chromosomes rather than evaluates the chromosomes to get the fitness, which saves a lot of resources in evaluations, especially for applications involving computationally expensive and/or time consuming fitness evaluations. It is used in the mutation of HdEA, by estimating the fitness landscape of the objective function, HdEA mutates chromosomes with an adaptively mutation towards the direction containing better evaluated chromosomes.

A. History implementation and fitness estimation in HdEA

HdEA uses a BSP tree as the memory archive which stores the position and the fitness values of evaluated chromosomes. It partitions the whole search space according to the distribution of evaluated chromosomes. It uses the same process as NrGA in tree construction, tree management and chromosomes recording. For the detailed description and working example, please refer to [1]. Note that there are four differences between the memory archives of HdEA and NrGA in terms of the tree operation and represented information:

1) The memory archive of HdEA stores both position and fitness values of the evaluated chromosomes but the memory archive of NrGA only stores the position.
2) By storing fitness of evaluated chromosomes, the archive of HDEA approximates the fitness landscape, but the archive of NrGA only represents the density of the evaluated chromosomes.
3) Search space of HdEA is continuous such that the number of possible chromosomes in every partition is infinite and no node-pruning is performed.
4) Since no node-pruning is preformed, the number of nodes in BSP tree is exactly the same as the number of evaluated chromosomes.

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

Definition 1: The sub-region of **x**

Suppose **x** is a chromosome in the search space S, i.e. $\mathbf{x} \in S$, and S is partitioned as the sub-region set $H=\cup_i h_i$ by the BSP tree, the sub-region $h \subseteq H$ is defined as the '*sub-region of* **x**' if $\mathbf{x} \in h$ and h is represented by a leaf node of the BSP tree.

In the approximated fitness landscape, the whole search space is divided into many sub-regions h_i by different evaluated chromosome $\mathbf{x_i}$. It is assumed that all chromosomes inside the same sub-region h share the same fitness f, i.e.: $f(\mathbf{x}) = f(\boldsymbol{a})$ for all $\mathbf{a} \in h$. Therefore, by determining the sub-region h which contains chromosome **a,** the estimated fitness of chromosome **a** can be determined. Fig.1 shows an example of the approximated fitness landscape of a 2-dimensional function.

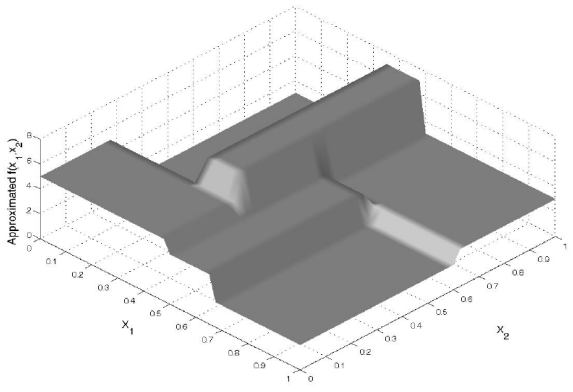

Fig. 1: Example of approximated fitness landscape of a 2-D function.

B. Idea of parameter-control system in HdEA

In this paper, we use "parameters" to represent both 1) operator choice and 2) parameter of the operator once the choice has been made.

As mentioned by Eiben et al. in [3], how to set the parameters of an EA is a challenging problem for researchers. Even EA experts who work in parameter control feel that this is a largely unresolved problem in the field. Note that the performance (i.e. property) of an EA may critically depend on their parameters; given the same design, setting good values for those parameters are very difficult.

Moreover, some researches empirically and theoretically demonstrated that different values of parameters should be used in different stages of the evolutionary process in order to optimize the performance of the EA [4-7]. Thus given a problem, we should not use a fixed set of parameters; it is more reasonable to change the parameters during the EA run. Similarly, different problems require different initial parameter settings. Though HdEA gives significantly better results than many other EAs [1], it runs with a fixed set of operators and parameters. In this paper, we aim to design a parameter control system for HdEA to decrease the number of parameters and make it more robust.

In this paper, we propose an adaptive parameter control system to automatically find the operators and parameters for HdEA. Different from other parameter control system, our system changes parameters *depending on the entire search history*. Using the taxonomy in [3], the method is a novel addition to the class of adaptive parameter control, which adjusts parameters using feedback. Meanwhile, since the changes in parameters will result in a change in the search strategy of the EA as well as the performance of the EA, search history is a good reference to directly measure the performance of an operator choice or a parameter value. Therefore, in the proposed system, the entire search history is used as a feedback to the system to estimate the "good" settings at different stages of different problems.

During the HdEA runs, the proposed system stores the entire search history and maintains the approximated fitness landscape for estimating the fitness of chromosomes. Besides storing the search history, it also keeps checking whether adjustment in parameters is needed. If adjustment is necessary, the system generates chromosomes with different sets of parameters. By calculating the average estimated fitness of those chromosomes – *without* actually evaluating them, the system measures the performance of parameters and changes parameters to the best settings. In general, the system has the following properties:

1) It is an adaptive parameter control system which adjusts parameters depending on whole search history.
2) Performance of parameters is obtained by estimating fitness in the history rather than evaluating the chromosomes, which can save a lot of resources in fitness evaluations when choosing parameters.
3) The learning process increases the accuracy of estimation continuously during the EA runs.
4) It can be considered as a natural extension of HdEA, the central idea being to reuse the entire search history for parameter control.

Since the parameter control system uses the entire history to learn an appropriate parameter setting, it is expected that the proposed system sacrifices a small amount of accuracy of the HdEA. Our goal is to sacrifice as little accuracy as possible in exchange for a decrease in the number of parameters in EA. It is doubtless that such a decrease is very valuable if one can also maintain the performance.

The rest of this paper is organized as follows: Section II presents a general framework of the proposed parameter control system using entire search history. Section III reports the application of the general framework on HdEA. Section IV presents experimental results and discussions. Finally, Section V draws a conclusion.

II. GENERAL FRAMEWORK OF PARAMETER CONTROL USING ENTIRE SEARCH HISTORY

A general framework for parameter control using the entire search history that is applicable to any EA is presented below. The framework consists of four main modules: (1) *Problem*, (2) *EA*, (3) *Learning process* and (4) *Parameter control*.

The *problem module* is the fitness function which is used to evaluate the fitness of chromosomes generated by EA.

The *EA module* contains the EA for which our parameter control system would set the parameters. It selects chromosomes which represents possible solutions to the problem and performs evolution operations (e.g. crossover and mutation). In this paper, we use HdEA as the EA module.

The *learning process module* records all chromosomes and their fitness in the memory and estimates the approximated fitness landscape of the problem. In this paper, we reuse the Binary Space Partitioning (BSP) Tree in HdEA as the memory archive to record the individuals and estimate the approximated fitness landscape. As the EA runs, more and more information will be recorded; the approximated landscape will become more detailed, such that the learning module can give better and better estimates to the parameter control system.

The *parameter control module* is the main part of the proposed system; it (i) checks whether the parameters of the EA should be adjusted and (ii) propose a set of adjusted parameter(s). This module includes a *triggering condition*. When the condition is satisfied, it implies that some changes should be made to the parameters, whence the parameter control system will be enabled to decide the necessary changes to the parameters. The process is as follows: It firstly generates chromosomes by different candidate sets of parameters – the possible candidate sets are designed into the system by the parameter control designer (e.g. One set of parameters may be: Crossover operator choice: Uniform crossover; crossover rate 0.9; the possible candidate crossover operator and the possible candidate crossover rates are designed into the system by the parameter control designer), then estimates the fitness of those chromosomes by the approximated landscape stored in learning process module. Note that this does not involve any actual fitness evaluation. For each candidate parameter set, P chromosomes are generated. This gives P estimated fitness. The average estimated fitness of the chromosomes is used to estimate the performance of a candidate parameter set. P chromosomes are used instead of 1 to give a more reliable estimated performance. The set of parameters with the best performance is chosen to be the parameter settings in the next generation.

The flow of the proposed system is as follows: first the EA selects chromosomes, performs evolution operations, then evaluates the fitness. Each time the problem responds by reporting the fitness of a chromosome to the EA, the learning process will record the chromosome and update the approximated fitness landscape. Also, when a generation is finished, the system will determine whether the triggering condition is satisfied or not. In this paper we use the decrease of slope of average fitness as the triggering condition. Let $f(x)$, f_{avg}, m and P be the fitness of chromosome x, average fitness, slope of average fitness and the population size respectively. Assume, without loss of generality, that we are solving minimization problems. The slope of average fitness in the generation n is calculated by eqn (1), which $f_{avg}(n)$ and $m(n)$ represents the average fitness and the slope of average fitness in the generation n respectively. When the difference of average fitness between two generations decreases, the slope of average fitness will be decreased, i.e., $m(n+1) < m(n)$.

$$m(n) = f_{avg}(n-1) - f_{avg}(n) \tag{1}$$

$$f_{avg} = \frac{\sum_{i=1}^{i=P} f(x_i)}{P} \tag{2}$$

Once this condition is satisfied, the parameter control system will be enabled to change parameters. In turn, the parameter control system generates chromosomes with different candidate sets of parameters, and reuses the fitness estimation process in HdEA to obtain the estimated fitness of those chromosomes by the approximated landscape in the learning process module. The system makes a decision to change the parameters to the set that generates better chromosomes on average. After that, the learning process and the EA will continue to run with the new parameter set, and the parameter control system will be disabled until adjustment in parameters is needed again.

A fine point to note is that, as the slope of average fitness is calculated by the difference of average fitness between the previous and the current generation, the slope of average fitness is undefined in the generation *0* (i.e., the initial generation) since $f_{avg}(-1)$ does not exist. Thus the parameter control system will be enabled once during parameter initialization. In other words, at the end of the initialization, the search history will consist of the P evaluated solution; parameter control is invoked; as a result, the parameters in the initial generation is determined automatically, without requiring user settings. The proposed framework is shown in fig. 2.

The algorithm of the parameter control system is shown below:

1. Initialization of the EA
2. Trigger the parameter control system;
3. **For** *generation* = 1: N
4. Generate a new population of offspring by the current set of parameters.
5. Evaluate the offspring.
6. Call the learning process to save information in the history.
7. Survivors selection operation.

8. Check whether the triggering condition is satisfied.
9. **If** the triggering condition satisfied
10. Generate individuals by different candidate sets of parameters.
12. Estimate the fitness of individuals by the history.
13. Change the parameter set to the set which gives the best estimated fitness.
14. **EndIf**
15. **Next** *generation*

Steps 1, 3, 4, 5 and 7 are the steps performed in a generic EA. The other steps are introduced to perform parameter control using the entire search history. For example, suppose one uses a canonical GA as the EA module. The system will initialize an empty memory archive before the GA initialization. The first generation of individual (generation *0*) will be generated randomly and evaluated during initialization (step 1). Then the parameter control system will be enabled once to initialize parameters based on the search history in the generation *0* (step 2). After that, the GA starts the evolutionary cycle (steps 3 – 15). In each generation, a pool of offspring will be generated by individuals in the previous generation using crossover and mutation (step 4). Those offspring will be evaluated and stored into the memory archive (steps 5-6). After the survivor selection operation, the average fitness of offspring $f_{avg}(n)$ and that of their parents $f_{avg}(n-1)$ will be used to calculate the slope of average fitness $m(n)$ (step 8). If the triggering condition satisfied, i.e., $m(n)<m(n-1)$ the parameter control system will be enabled once to change the parameters in the next generation (steps 9-14). Then the GA continues to run the next generation (step 4) .

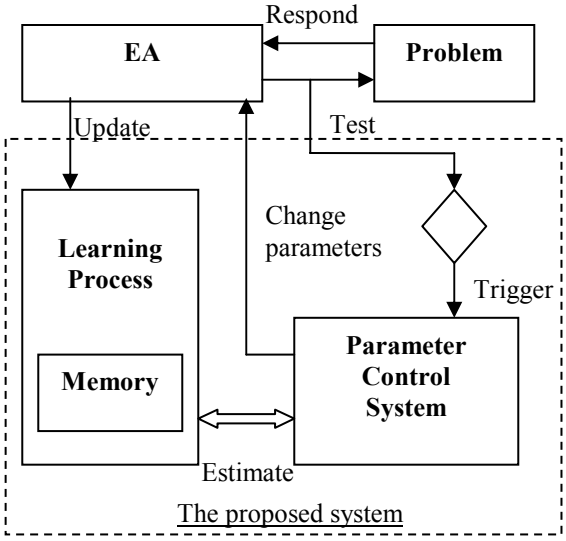

Fig. 2: Block diagram of the framework

III. Experiment of Parameter control system using HdEA

A. Experiment Settings

In this paper, the general framework that we have outlined above is applied to the parameter control of HdEA. The resulting novel HdEA shall be called HdEA-PC, which stands for HdEA with parameter control using the entire search history. The settings of the experiment are shown in Table 1. The learning module in this experiment is reusing the memory archives and fitness approximation method in HdEA [1], which is a nearest neighbour search; the system estimates that the new chromosome has the same fitness as the most similar evaluated chromosomes. The original crossover operation settings of HdEA are uniform crossover with crossover rate 0.9. For more details about the settings of HdEA, please refer to [1].

TABLE I
Experiment setting

EA module	HdEA
Memory	BSP tree
Learning module	Nearest Neighbor Search
Triggering Condition	Slope of average fitness decreases
Target parameter	Crossover operation
Target Set	Gene, Arithmetic, Uniform, One-point
Target parameter set	0.1,0.4,0.7,0.9
Population size	20
No. of Generation	2000
Mutation	Anisotropic parameter-less adaptive mutation (as used in HdEA)
Selection scheme	(μ+μ) elitism selection (as used in HdEA)
Problem Environment	34 benchmark functions

In this experiment, we wish to test the framework on its ability to select automatically, online, and in a parameter-less manner the crossover operator and the crossover rate of HdEA. Four different crossover methods are in the candidate set:

i) *Gene Crossover*: It randomly selects one gene to swap between two parents. Crossover rate is not necessary in this operator.
ii) *Arithmetic Crossover*: The child chromosome is calculated by interpolating the two parents. The crossover rate is used as the interpolation ratio.

iii) *Uniform Crossover*: Each gene in the parents has a probability of swapping. The crossover rate is the probability to swap.

iv) *One-point Crossover*: It randomly selects one position to swap all genes in the parents after that position. The crossover rate is the probability to swap.

Apart from the *Gene crossove*r, each crossover operator is has four candidate crossover rates. Thus in total, there are 13 different combinations of crossover operators and crossover rates involved in this experiment. Besides the crossover operation, all other settings follow the original HdEA design [1].

B. Test functions

The same 34 real valued functions used in [1] are used in this paper to show the performance of HdEA after adding the parameter control system. The 34 functions (*f1-f34)* are listed below:

1. Sphere function [8]
2. Schwefel's problem 2.22 [8]
3. Schwefel's problem 1.2 [8]
4. Schwefel's problem 2.21 [8]
5. Generalized Rosenbrock function [8]
6. Quartic function [8]
7. Generalized Rastrigin function [8]
8. Generalized Griewank function [8]
9. Generalized Schwefel's problem 2.26 [8]
10. Ackley function [8]
11. Shekel's Foxholes function [8]
12. Six-Hump Camel-Back function [8]
13. Branin function [8]
14. Goldstein-Price function [9]
15. High conditioned elliptic function [9]
16. Weierstrass's function [9]
17. Hybrid composition function [9]
18. Levy function [10]
19. Zakharov function [10]
20. Alpine function [10]
21. Pathological function [10]
22. Inverted cosine wave function (Masters) [10]
23. Inverted cosine mixture problem [11]
24. Epistatic michalewicz problem [11]
25. Levy and Montalvo 2 problem [11]
26. Neumaier 3 problem [11]
27. Odd Square problem [11]
28. Paviani problem [11]
29. Periodic problem [11]
30. Salomon problem [11]
31. Shubert problem [11]
32. Sinusoidal problem [11]
33. Michalewicz function [10]
34. Whitely's function [8]

Seven of them are uni-modal functions; the remaining twenty-seven are multi-modal functions designed with a considerable amount of local minima. Additionally, function *f6* is a noisy function and function *f17* is a hybrid composition function. The dimensions of the first ten and the last twenty functions are adjustable while the dimensions of *f11 - f14* are fixed at two, as they are two-dimensional functions as defined in the original references.

C. Test algorithms

HdEA-PC has been compared with HdEA and 8 other EA. Apart from the crossover operator and the crossover rate, HdEA-PC uses the same settings as HdEA. The 8 other EA uses the recommended parameter settings in the original literature. For more details, please refer to [1].

1) History Driven Evolutionary Algorithm with Parameter Control System (**HdEA-PC**).
2) History Driven Evolutionary Algorithm **(HdEA)** [1].
3) Real coded GA with Uni-modal Normal Distribution Crossover **(RCGA-UNDX)** [12].
4) Covariance Matrix Adaptation Evolutionary Strategy (**CMA-ES)** [13].The source code of CMA-ES is taken from [13] (Aug. 2007 version).
5) Differential Evolution **(DE)** [14].
6) Opposition-Based Differential Evolution **(ODE)** [15].
7) Differential Evolution with Adaptive Hill-Climbing Simplex Crossover **(DEahcSPX)** [16].
8) Dissipative Particle Swarm Optimization **(DPSO)** [17].
9) PSO with Spatial Particle Extension **(SEPSO)** [18].
10) Estimation of Distribution Algorithm **(EDA)** [19].The source code of the EDA is taken from [20] (Feb. 2009 version).

IV. RESULTS

The number of independent trials for each function is 100. Except functions *f11-f14* whose dimensions are fixed at 2, all other functions are tested in 30 and 40 dimensions. The experimental results are shown in Table II to Table VII.

To illustrate the difference between HdEA-PC with other algorithms clearly, Table II to Table VI are shown in this format: The first three columns of the tables show the tested fitness function, and the performance in ranks of HdEA and that of HdEA-PC respectively. An algorithm with rank 1 (10) is the best (worst) algorithm respectively. In the fourth and fifth columns of those tables, upper *rank* and *lower rank* show the average fitness of the algorithm that ranks higher and lower than HdEA-PC by 1 respectively. For example, if the rank of HdEA-PC is 2, the *upper rank* and *lower rank* show the average fitness of the algorithm whose ranks are 1 and 3 respectively.

HdEA-PC gives a similar or better performance in 26 out of 34 functions. Amongst these 26 functions, the parameter control system gives significant improvements to 8 functions, and gives similar results to 18 functions. It empirically shows that the parameter control system can set "good" parameters to the HdEA.

The parameter control system gives significant improvement to the original HdEA in 8 functions: *f3-4, f6, f14, f19, f26, f30 and f32*. In those functions, the HdEA-PC performs better than original HdEA, and the difference in fitness is relatively large compared to the algorithms within 1 rank. For example, in *f3*, the average fitness of HdEA is 16920.23, the difference between HdEA and HdEA-PC is relatively large when compared to the difference between HdEA-PC and *upper rank* (ODE: 78.17) and *lower rank* (DEahcSPX: 1955.22). It means in those functions, the original settings of HdEA, which uses uniform crossover operator with crossover rate 0.9, is not suitable for those functions, but the parameter control system *corrects* the parameters in crossover operation to the right set.

The HdEA-PC gives similar results as HdEA in 18 functions: *f1-2, f7-10, f16-18, f21-24, f27-29, f31 and f33*. Nonetheless, it is worthy to note that the difference between HdEA and HdEA-PC is relatively smaller than the differences between HdEA-PC and other algorithms. Results in this group shows that the parameters suggested by the parameter control system are *almost the same* as the original settings of HdEA.

Finally, only in 8 functions, *f5, f11-13, f15, f20, f25 and f34*, the average performance decreases after adding the parameter control system. In those cases, the difference in fitness is relatively high. It means that the parameter control system has suggested the wrong parameters settings to the HdEA-PC. As a result, many generations are sacrificed because of improper learning.

The average ranking of all algorithms are shown in Table VII for reference. In this table, we can see that the HdEA and the HdEA-PC ranks 1 and 2 respectively in the test algorithms set. Although the HdEA-PC does not outperform HdEA after adding the parameter control system, HdEA-PC still works better than other algorithms in the test algorithms set. This is an achievement because unlike HdEA, the algorithm designer has not manually specified a good crossover operator and a good crossover rate that would work well on the 34 benchmark function by his experience – which uses human intelligence, but HdEA-PC finds them out *by itself*. In other words, HdEA-PC releases the heavy burden of the algorithm designer on the settings of two parameters (the type of crossover, and its crossover rate as a function of the stages of the optimization)

V. CONCLUSION

In this paper, we have proposed a novel general Evolutionary Algorithm (EA) framework, "Parameter Control Using the Entire Search History", EA-PC, for controlling parameters in an EA. By reusing the memory archives in History driven Evolutionary Algorithm (HdEA), the proposed system uses the entire search history in a novel way for parameter control (parameters include both operators and parameters). It estimates fitness of chromosomes generated by different parameters to help the EA to choose parameters online, automatically, and in a parameter-less manner. As a result, the algorithm designer is relieved of the burden to set some parameters in the EA.

The proposed framework has five main advantages:

1) It adaptively changes the parameters without using any prior knowledge about the problem.
2) We propose to measure the performance of the candidate sets of parameters by the average estimated fitness of corresponding candidate solutions. The fitness values of these candidate solutions are estimated from the search history instead of performing an actual fitness evaluation. Large amount of computation effort is saved as a result for computationally expensive and/or time consuming fitness functions.
3) The proposed system uses an incremental learning approach. It is expected that the suggested parameters from the system are more and more accurate as the search progresses.
4) It is not necessary to set the initial settings for the parameters.
5) Instead of running the proposed system at each iteration, we proposed a novel fitness improvement based triggering condition which optimizes the frequency of the parameter adjustment.

HdEA-PC is compared with HdEA (with designer choice of fixed crossover operator and fixed crossover rate) and other 8 famous EA in 34 benchmark functions with dimensions from 2 to 40. It is shown in 26 functions that the average performance of HdEA-PC is better (in 8 functions) or similar (in 18 functions) than the original HdEA. In the remaining 8 functions, HdEA-PC performs worse than HdEA because of incorrect learned knowledge. Nevertheless, HdEA-PC achieves impressive performance; it ranks only 2nd to HdEA and outperforms the other 8 EAs. The result that it does not outperform HdEA is expected, as HdEA uses designer choices of parameters, while HdEA-PC relieves the burden of the designer to set two parameters (crossover operator and crossover rate) but still remains very competent. We should have reason to believe that HdEA-PC will perform well in novel, unfamiliar problems beyond those of the 34 well known benchmarks.

From the empirical results, we conclude that the parameter control system using entire search history can help to find an appropriate setup for HdEA without significant performance drop. Since it seems reasonable to use the entire history in a non-parametric manner as the feedback in controlling the parameters in EA; and the results shows that it works in the HdEA, in the future, we plan to study the effect of parameter control using the entire search history on other EAs.

REFERENCES

[1] C.K. Chow and S.Y. Yuen, "An Evolutionary Algorithm that Makes Decision Based on the Entire Previous Search History," *IEEE Trans. Evol.Comput.*, to be published.

[2] S.Y. Yuen and C.K. Chow, "A Genetic Algorithm that Adaptively Mutates and Never Revisits," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 454-472, Apr. 2009.

[3] A.E. Eiben, Z. Michalewicz, M. Schoenauer, and J.E. Smith, "Parameter control in Evolutionary Algorithms," *Studies in Computational Intelligence*, Berlin: Springer, vol. 54, 2007, pp. 19-46.

[4] T. Bäck, "The interaction of mutation rate, selection and self-adaptation within a genetic algorithm," in *Proc. 2nd Conf. on Parallel Problem Solving from Nature,* 1992, pp. 85-94.

[5] T. Bäck, "Self-adaptation in genetic algorithms," in *Toward a Practice of Autonomous Systems: Proc. 1st European conf. on Artificial Life,* 1992, pp. 263-271.

[6] T. Bäck, "Optimal mutation rates in genetic search," in *Proc. 5th Int. Conf. on Genetic Algorithms,* 1993, pp.2-8.

[7] G. Syswerda, "A study of reproduction in generational and steady state genetic algorithms," in *Foundations of Genetic Algorithms,* G. J. E. Rawlins, Ed . Los Altos: Morgan Kaufmann, 1991, pp. 94-101.

[8] X. Yao, Y. Liu, and G.M. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.,* vol.3, no.2, pp. 124-141, Jul.1999.

[9] 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," *Technical Report,* Nanyang Technological University, Singapore, *KanGAL Report* #2005005, IIT Kanpur, India, 2005.

[10] V.K. Koumousis, C.P. Katsaras, "A sawtooth genetic algorithm comvining the effects of variable population size and reinitialization to enchance performance," *IEEE Trans. Evol. Comput.,* vol. 10, no.1, pp. 19-28, Feb. 2006.

[11] M.M. Ali, C. Khompatraporn, and Z.B. Zabinsky, "A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems," *Journal of Global Optimization,* vol. 31, no. 4, pp. 635-672, Apr. 2005.

[12] I. Ono and S. Kobaashi, "A Real-code Genetic Algorithm for Function Optimization Using Unimodal Normal Distribution Crossover," in *Proc. 7th Int. Conf. on Genetic Algorithm,* pp.246-253, 2007.

[13] N. Hansen, "The CMA Evolutionary Strategy: A Tutorial," Technical Report, 31 Aug. 2007. Link: www.bionik.tu-berlin.de/user.niko/cmatutorial.pdf

[14] R. Storn and K. Price, "Differential evolution-A simple and efficient adaptive scheme for global optimization over continuous spaces." Berkeley, CA, Technical Report TR-95-012, 1995.

[15] H.R. Tizhoosh, "Opposition-based learning: A new scheme for machine intelligence," in *Proc. of Int. Conf. Comput. Intell. Modeling Control and Autom.,* 2005, pp. 695-701.

[16] N.Noman and H. Iba, "Accelerating Differential Evolution Using an Adaptive Local Search," *IEEE Trans. Evol. Comput.,* vol. 12, no.1, pp.107-125, Feb. 2008.

[17] X.F. Xie, W.J. Zhang and Z.L. Yang, "A dissipative particle swarm optiziation," in *Proc. IEEE Cong. Evol. Comput.,*2002, pp. 1666-1670.

[18] T. Krink, J.S. Vesterstrom and J. Riget, "Particle swarm optimization with spatial particle extension," in *Proc. IEEE Cong. Evol. Comput.,* 2002, pp. 1474-1497.

[19] R. Santana, C. Echegoyen, A. Mendiburu, C. Bielza, J.A. Lozano, P. Larrañaga, R. Armañanzas and S. Shakya, "MATEDA: A suite of EDA programs in Matlab," Technical Report EHU-KZAA-IK-2/09, University of the Basque Country, February 2009.

[20] Matlab toolbox for Estimation of Distribution Algorithms (MATEDA-2.0). Link: http://www.sc.ehu.es/ccwbayes/members/rsantana/software/matlab/IntEDA.tar.gz

TABLE II
EXPERIMENT RESULT FOR FUNCTION 1 - 7

	functions	HdEA (rank)	HdEA - PC (rank)	Upper rank (algorithm)	Lower rank (algorithm)
f1	30	0.00 (1)	0.000008 (5)	0.00 (ODE)	0.02 (DE)
	40	0.00 (1)	0.000875 (5)	0.0001 (ODE)	0.01 (DEahcSPX)
f2	30	0.00 (1)	0.000079 (4)	0.00 (CMA-ES)	0.03 (DEahcSPX)
	40	0.0034 (4)	0.005384 (5)	0.0034 (HdEA)	0.35 (ODE)
f3	30	16920.23 (9)	259.83525 (4)	78.17 (ODE)	1955.22 (DPSO)
	40	34006.13 (9)	645.41114 (4)	409.99 (ODE)	3986.09 (DPSO)
f4	30	10.8802 (5)	0.210545 (3)	0.01 (ODE)	10.54 (DPSO)
	40	22.6509 (7)	0.570278 (3)	0.15 (ODE)	13.3 (DPSO)
f5	30	21.1276 (2)	141.75715 (5)	71.13 (RCGA-UNDX)	614.46 (DE)
	40	93.6767 (4)	300.21829 (5)	93.6767 (HdEA)	2023.17 (DE)
f6	30	10.4615 (7)	8.495496 (2)	0.23 (CMA-ES)	8.52 (ODE)
	40	16.1244 (8)	12.410698 (2)	0.27 (CMA-ES)	12.43 (ODE)
f7	30	0.00 (1)	0.000074 (2)	0.00 (HdEA)	25.71 (DE)
	40	0.0082 (1)	0.0082 (2)	0.00 (HdEA)	57.39 (DE)

TABLE III
EXPERIMENT RESULT FOR FUNCTION 7 - 16

	functions	HdEA (rank)	HdEA - PC (rank)	Upper rank (algorithm)	Lower rank (algorithm)
f8	30	0.00 (1)	0.004447 (3)	0.00 (CMA-ES)	0.03 (ODE)
	40	0.001(2)	0.14573 (4)	0.04 (DEahcSPX)	0.24 (ODE)
f9	30	-13780.72 (1)	-13776.7031 (2)	-13780.72 (HdEA)	-12801.84 (DE)
	40	-18374.62 (2)	-18370.321 (2)	-18374.62 (HdEA)	-15568.76 (DE)
f10	30	0.00 (1)	0.000189 (2)	0.00 (HdEA)	0.01 (ODE)
	40	0.005 (1)	0.006266 (2)	0.005 (HdEA)	0.12 (ODE)
f11	2	1.2082 (1)	2.885618 (7)	2.821 (EDA)	6.52 (RCGA-UNDX)
f12	2	-1.0316 (1)	-1.026798 (4)	-1.03 (SEPSO)	-1.02 (CMA-ES)
f13	2	0.401 (3)	0.451461 (7)	0.43 (DEahcSPX)	0.46 (RCGA-UNDX)
f14	2	4.4101 (7)	3.985438 (6)	3.52 (ODE)	4.41101 (HdEA)
f15	30	0.00 (1)	3.190466 (3)	1.16 (ODE)	41.44 (CMA-ES)
	40	1.6503(1)	795.127855 (3)	254.33 (ODE)	2085.96 (DEahcSPX)
f16	30	0.0047 (1)	0.0007041 (2)	0.0047 (HdEA)	0.74 (DEahcSPX)
	40	0.1451 (2)	0.123465 (1)	----------	0.1451 (HdEA)

TABLE IV
EXPERIMENT RESULT FOR FUNCTION 17- 23

	functions	HdEA (rank)	HdEA - PC (rank)	Upper rank (algorithm)	Lower rank (algorithm)
f17	30	8814.832 (6)	7625.46599 (5)	6571.29 (CMA-ES)	8814.832 (HdEA)
	40	14179.81 (5)	14678.6596 (6)	14179.81 (HdEA)	34420.03 (RCGA-UNDX)
f18	30	0.00 (1)	0.00 (1)	----------	0.0032 (DE)
	40	0.00 (1)	0.00003 (3)	0.00 (CMA-ES)	0.000568 (ODE)
f19	30	261.3953 (7)	3.65284 (2)	2.21 (DEahcSPX)	69.74 (EDA)
	40	395.2277 (7)	18.244904 (1)	----------	30.15 (DEahcSPX)
f20	30	0.0004 (1)	0.003572 (3)	0.0013 (DEahcSPX)	0.03 (ODE)
	40	0.0057 (2)	0.011904 (3)	0.0057 (HdEA)	0.09 (ODE)
f21	30	4.8663 (3)	5.065368 (4)	4.8663 (HdEA)	5.4 (DE)
	40	7.6184 (3)	7.909705 (4)	7.6184 (HdEA)	8.22(DE)
f22	30	-24.9443 (2)	-24.794646 (3)	-24.9443 (HdEA)	-18.728 (EDA)
	40	-31.9794 (2)	-31.70918 (3)	-31.9794 (HdEA)	-22.469 (EDA)
f23	30	0.00 (1)	0.00 (1)	----------	0.00 (HdEA)
	40	0.00 (1)	0.000003 (3)	0.18 (DEahcSPX)	0.02 (SDE)

TABLE V
EXPERIMENT RESULT FOR FUNCTION 24- 30

	functions	HdEA (rank)	HdEA - PC (rank)	Upper rank (algorithm)	Lower rank (algorithm)
f24	30	-25.3678 (1)	-24.875671 (2)	-25.3678 (HdEA)	-19.18 (CMA-ES)
	40	-32.2103 (1)	-31.824718 (2)	-32.2103 (HdEA)	-24.01 (CMA-ES)
f25	30	0.1626 (2)	28.717005 (5)	26.1 (ODE)	37.17 (DEahcSPX)
	40	2.4995 (2)	39.576957 (4)	18.692 (EDA)	47.35 (ODE)
f26	30	8025.425 (5)	113.72684 (3)	-2428.19 (ODE)	1911.3 (DEahcSPX)
	40	73996.06 (5)	4209.59937 (3)	413.73 (ODE)	8562.51 (DEahcSPX)
f27	30	-0.0004 (8)	-0.001928 (7)	-0.0107 (ODE)	-0.0004 (HdEA)
	40	-0.0001 (8)	-0.000825 (7)	-0.0032 (ODE)	-0.0001 (HdEA)
f28	30	-997867 (2)	-997867.463 (1)	------	-997867 (HdEA)
	40	-1.00E+8 (1)	-99993721 (2)	-1.00E+8 (HdEA)	-98215006 (CMA-ES)
f29	30	1.0004 (2)	1.001484 (3)	1.0004 (HdEA)	1.45 (DE)
	40	1.0046 (2)	1.008876 (3)	1.0046 (HdEA)	2.14 (DE)
f30	30	1.2051 (5)	0.6309999 (3)	0.5 (DEahcSPX)	1.2 (CMA-ES)
	40	2.1347 (5)	1.009196 (3)	0.9 (ODE)	1.46 (CMA-ES)

TABLE VI
EXPERIMENT RESULT FOR FUNCTION 31-34

	functions	HdEA (rank)	HdEA - PC (rank)	Upper rank (algorithm)	Lower rank (algorithm)
f31	30	-2.00E+34 (1)	-1.602E+34 (2)	-2.00E+34 (HdEA)	-9.03E+29 (DE)
	40	-2.71E+45 (1)	-1.07E+45 (2)	-2.71E+45 (HdEA)	-1.55E+38 (DE)
f32	30	-1.521 (9)	-3.498522 (2)	-3.5 (ODE)	-3.37 (RCGA-UNDX)
	40	-1.3187 (8)	-3.478247 (3)	-3.49 (ODE)	-2.52 (CMA-ES)
f33	30	-29.559 (1)	-29.493198 (2)	-29.559 (HdEA)	-24.87 (DE)
	40	-38.8299 (1)	-38.644751 (2)	-38.8299 (HdEA)	-30.02 (DE)
f34	30	20.6012 (1)	746.85978 (3)	319.37 (CMA-ES)	766.95 (ODE)
	40	694.993 (2)	1507.15997 (4)	1438.33(ODE)	1765.99 (DE)

TABLE VII
AVERAGE RANKING OF ALGORITHMS

Algorithms	Average Ranking
HdEA	3.109375
RCGA-UNDX	7.6875
CMA-ES	4.375
DE	5.90625
ODE	3.390625
DEahcSPX	4.5
DPSO	7.0625
SEPSO	6.703125
EDA	7.34375
HdEA - PC	3.265625