A Hybrid Artificial Bee Colony with Differential Evolution

Chukiat Worasucheep

***Abstract*—Artificial bee colony (ABC) is a relatively new stochastic algorithm with competitive performance and minimal tuning parameter. This paper proposes a hybrid ABC algorithm with differential evolution (DE), but without additional parameters. DE is a well-known efficient evolutionary algorithm with proven records but its parameter setting is complicated. This proposed hybrid algorithm called ABCDE incorporates the powerful mutation strategies of DE into ABC, in order to increase convergence while diversity is not compromised. The performance of ABCDE is evaluated against both original ABC and opposition-based DE (ODE), a recent DE variant with high performance. The experiment uses twelve widely accepted non-linear benchmark functions with various characteristics, such as difficult landscape, multimodality, shift and rotation, to evaluate the ABCDE's performance on many complex functions. The experimental results demonstrate a superior performance of ABCDE against original ABC and ODE.**

***Index Terms*—Artificial bee colony, hybridization, differential evolution.**

I. Introduction

Artificial bee colony (ABC) algorithm [1] is a biological-inspired population-based stochastic algorithm, recently proposed by D. Karaboga, which mimics the foraging behavior of honey bee swarm. Performance of ABC algorithm has been demonstrated to be competitive to other population-based algorithms with an advantage of simplicity and having fewer control parameters [2], [3]. Thus ABC has been applied to solve different optimization problems such as machining process [4], scheduling [5], structural design problem [6] and power electric [7]. The ABC employed in this work is a hybrid with differential evolution (DE) in order to increase exploitation, the ability of searching near a candidate solution. DE is a very efficient evolutionary algorithm proposed by Storn and Price [8], whose performance has been improved and widely accepted in many areas from engineering and science to finance and economic since [9]. The hybridization in this work increases ABC's exploitation without additional algorithmic parameters. Performance of the proposed hybrid algorithm, namely ABCDE, is evaluated using a set of widely accepted benchmark of nonlinear minimization problems with different characteristics including multimodality, shift and rotation.

The remainder of the paper has the following structure. Section II presents background of ABC and overview of DE and ANN. Section III describes the proposed hybrid ABCDE whose performance is evaluated using benchmark functions in Section IV. Section V concludes the paper with some future works.

Manuscript received October 1, 2014; revised December 23, 2014.

C. Worasucheep is with the King Mongkut University of Technology Thonburi, Bangkok, Thailand (e-mail: chukiat.wor@kmutt.ac.th).

II. Background

Without loss of generality, this paper considers minimization problems only.

A. Artificial Bee Colony

Artificial Bee Colony (ABC) algorithm is a relatively new population-based algorithm proposed by D. Karaboga in 2005 to solve continuous problems [1]. ABC is modeled from the foraging behavior of honey bee colony that consists of three different groups of bees: employed bees, onlooker bees and scout bees. The employed bees search food sources around the hive and carry information about food source positions. Onlooker bees are waiting in the hive for the information shared by the employed bees about their found food sources, given that each employed bee carries information of one different food source. An onlooker bee evaluates the information showed by the dancing employed bees and choose a food source according to the probability proportional to the quality, the amount of nectar, of that food source. Once an onlooker is attracted to one food source information, it updates food information, keeps only the better one, and shares its information on the dance platform. If a food source could not be improved by the employed or onlooker bees in a certain time (called as abandon limit), the employed bee abandons this food source and becomes a scout bee exploring a new food source.

In ABC algorithm, there are N food sources, each of which is a candidate solution whose position is represented as x_i (i = 1,…, N) of D dimensional vector, where D is the number of optimization variables. Initially, the population of solutions is created and each x_i is randomized with D-dimensional real-value vector:

$$x_{i,j} = x_{\min,j} + rand(0,1)(x_{\max,j} - x_{\min,j}) \quad (1)$$

where j= 1,2,…,D. $x_{\min,j}$ and $x_{\max,j}$ are the lower and upper bounds of the dimension j, respectively. $rand(0,1)$ is a random number drawn from a uniform distribution in a range of 0 and 1. Fitness values of all solutions, representing quality of the food sources, are then evaluated. Similar to other population-based algorithms, ABC is an iterative process and the following steps will be executed repeatedly until some termination criteria is met.

Each employed bee x_i generates a new food source v_i in the neighborhood of its current position by using the following search equation:

$$v_{i,j} = x_{i,j} + \phi_{i,j}(x_{i,j} - x_{k,j}), \quad (2)$$

where $k = \{1, 2,\ldots, N\}$ and $j = \{1, 2, \ldots, D\}$ are randomly chosen indexes; k has to be different from i. $\phi_{i,j}$ is a random number in the range [-1, 1]. The new food source v_i will be evaluated and compared to x_i. If v_i is better than x_i, the employed bee will memorize the new food source v_i and forget the previous one; otherwise x_i is retained.

After all employed bees complete their searches, they return to the dancing area in the hive and share information to the onlooker bees. An onlooker chooses a food source depending on the probability value associated with the food source p_i. In original ABC, roulette wheel selection method is employed as follows.

$$p_i = fit_i \Big/ \sum_{j=1}^{N} fit_j, \quad (3)$$

where fit_i is the fitness value of solution i. The higher value the fit_i is, the more probability that the i-th food source is selected. Once an onlooker selects its food source, it produces a modification on the x_i by using equation (2) and evaluates the new food source. Similar to the case of the employed bee, an onlooker replaces the selected food source with the new one if the new one is better.

If a food source x_i cannot be improved by the bees for a predetermined number of times, called abandon *limit*, the food source is abandoned and could be newly produced by a scout bee using equation (1).

B. Improved Variants of ABC

It is widely known that both exploration (diversification) and exploitation (intensification) are necessary for any population-based search algorithms. Unfortunately, they are contradict to each other and both must be well balanced during the search for good performance. The original ABC is well known for its exploration but poor at exploitation[10], [11]. Thus many techniques have been employed to enhance the convergence and performance reliability of ABC. Some of them include local search [5], Tagushi method [6], orthogonal learning strategy [11], multi-strategy ensemble [12]. Kang *et al.*'s Rosenbrock ABC algorithm [13] used a modified Rosenbrock's rotational direction method to implement the exploitation phase to assist ABC in solving complex problems. Bose *et al.* [14] proposed the idea of decentralization of attraction from super-fit members along with neighborhood information and wider exploration of search space and applied it to optimal filter design problems.

In additions, hybridization with other well-known population-based algorithms such as particle swarm optimization (PSO) or differential evolution (DE), have been proposed to enhance the performance of ABC [7], [15]. Inspired by PSO, the gbest-guided ABC exploits the information of global best solution into the search equation to improve the exploitation [10]. Inspired by DE/best/1, the modified ABC searches only around the best solution to improve the exploitation [11] in addition to an improved initialization with chaotic system and opposition-based learning. A new probability parameter is added for balancing of the exploration and exploitation. More recently, a hybrid ABC with DE integrates a modified DE into the modified gbest-guided ABC to further accelerate the convergence [16]. A catastrophe-like scheme is used to prevent stagnation at the later stage of evolution by abruptly initializing some worse individuals of the population. Despite its promising performance, this hybrid ABC has four additional parameters: two for the improved search equations and two for the stagnation prevention scheme. Yang *et al.* [17], applied the mutation and crossover strategies of DE to the employed bees to enforce their exploration ability while onlooker bees keep their original updating strategy to retain the exploitation ability. Li and Yin proposed another simple hybrid ABC and DE where the DE operation works as main structure and ABC works only when the solution created by DE operator does not make an improvement [18]. Unfortunately, a comprehensive simulation of the algorithms as proposed in Yang *et al.* [17] and Li and Yin [18] did not achieve a comparable results as reported, therefore, both algorithms are not included in this study for a comparative analysis.

C. Differential Evolution

Differential evolution is a population-based stochastic search algorithm that evolves a population of candidate solutions, called vectors, towards global optima. Firstly, a population of vectors are randomly initialized within the search space and evaluated with the objective function provided. Then each vector, so-called target vector, undergoes three operations in sequence: mutation, crossover and replacement [8].

1) Mutation. A base vector is first selected from a randomly chosen or the best vector of the population. Then the difference(s) of one or sometimes two pairs of randomly chosen vectors are scaled and added to the basis vector to produce a mutant vector.

2) Crossover. To enhance the potential diversity of the population, the mutant vector exchanges its components with the target vector to form a trial vector.

3) Selection. The new trial vector and the corresponding target vector will be compared to keep only one of them to survive. If the trial vector has an equal or better value of the fitness value, it replaces the target vector in the next generation; otherwise the target vector is retained. This selection is thus in a greedy way and the overall fitness value will keep better or remain the same [19].

There exist abundant methods developed so far to create the base vector and the difference vectors, which correspond to different mutation strategies. One of the most widely used mutation strategy is DE/rand/1, meaning that one randomly chosen vector is used as the base vector and one pair of randomly chosen vectors are used to create the difference vector in mutation. This strategy provides a moderate to good performance for a wide range of problems. Another popular strategy is DE/best/1 which is similar to DE/rand/1 except that the currently best vector is selected to be the base vector. DE/best/1 strategy yields a good convergence for simple unimodal problems. Many more strategies can be found in a recent survey [9].

An interesting mutation strategy is DE/current-to-rand/1 which is rotationally invariant and has superior performance on difficult problems [9], [19]. DE/current-to-rand/1 replaces the binomial crossover operator with the rotationally invariant arithmetic line recombination operator to generate the trial

vector by linearly combining the target vector and the corresponding trial vector as follows:

$$\vec{u}_i = \vec{x}_i + k_i \cdot (\vec{x}_{r1} - \vec{x}_i) + F' \cdot (\vec{x}_{r2} - \vec{x}_{r3}) \quad (4)$$

where $\vec{u}_i$ and $\vec{x}_i$ are the trial vector and the target vector. r_1, r_2 and r_3 are indexes of vector randomly chosen and $r_1 \neq r_2 \neq r_3 \neq i$. k_i and F' are the combination coefficients, which are recommended to be a random value from a uniform distribution between 0 and 1.

III. The Proposed Hybrid ABCDE

The original ABC is well known for its exploration but poor at exploitation [10], [11]. This is mainly because of the following three reasons found in the search equation (2). Firstly, this equation is used by both the employed bees and the onlooker bees. Thus no variety search behavior can be anticipated. Secondly, the new food source is created depending on the *k*-th member, which is selected only with a random. No directional improvement can be foreseen at this point. And third, the original ABC modified only one dimension, say the *j*-th dimension, at a time. Updates on more dimensions can expect higher diversity improvement.

Algorithm 1: Hybrid ABCDE algorithm

Given objective function f and dimension D
Set running parameter MAXNFC
Create and randomly initialize all solutions x_i, $i = 1..PS$
Designate the bee with minimum$f(x_i)$ as *best*
Set $trial_i = 0$, $i = 1..PS$
Set $limit = 0.6 \cdot D \cdot PS$
nfc = 0
while Stop condition is not satisfied, i.e. nfc<= MAXNFC **do**
  // the employed bee phase
  Randomly create $F' \in [0, 1]$
  for $i = 1$ **to**PS**do**
    Create random integer numbers $r1, r2, r3 \in [0, PS] \wedge r1 \neq r2 \neq r3$
    Randomly create $k \in [0, 1]$
    Generate and evaluate a new food source v_i according to equation (5)
    nfc = nfc + 1
    if $f(v_i) < f(x_i)$ **then**
      $x_i = v_i$, $trial_i = 0$
      Designate x_i as the *best* bee if$f(x_i) < f(x_{best})$
    else
      $trial_i = trial_i + 1$
    end if
  end for
  if nfc> MAXNFC **then**
    break
  end if
  // the onlooker phase
  Calculate the probability values p_i for the solutions p_i according to equation (3)
  for $i = 1$ **to**PS**do**
    Create two random integer numbers $src, dest \in [0, PS] \wedge src \neq dest \wedge f(x_{dest}) < f(x_{src})$
    Randomly create $F \in [0, 1]$
    Generate and evaluate a new solution v_i according to equation (6)
    nfc = nfc + 1
    if $f(v_i) < f(x_i)$ **then**
      $x_i = v_i$, $trial_i = 0$
      Designate x_i as the *best* bee if$f(x_i) < f(x_{best})$
    else
      $trial_i = trial_i + 1$
    end if
  end for
  // the scout bees
  for $i = 1$ **to**PS**do**
    if $trial_i >$ limit **then**
      Randomly initialize the bee x_i with equation (1)
      Evaluate the bee
      nfc = nfc + 1
  end for
endwhile

TABLE I: The Benchmark Problems (F_{MIN}: Known Optimum Value, U: Unimodal, M: Multimodal, S: Separable, N: Non-separable, Sh: Shifted, Ro: Rotated)

Group	Subgroup	Function	Search range	f_{min}	Type	Sh/Ro
Basic		$f1$ Sphere	$[-50, 50]^D$	0	US	-
		$f2$ Quartic with noise	$[-1.28, 1.28]^D$	0	US	-
		$f3$ Rosenbrock	$[-30, 30]^D$	0	MN	-
		$f4$ Ackley	$[-32, 32]^D$	0	MN	-
		$f5$ Griewank	$[-600, 600]^D$	0	MN	-
		$f6$ Rastrigin	$[-5.12, 5.12]^D$	0	MS	-
Complex	CEC05	$f7$ Shifted Sphere	$[-100, 100]^D$	-450	US	Sh
		$f8$ Shifted Rosenbrock	$[-100, 100]^D$	390	MN	Sh
		$f9$ Shifted Rastrigin	$[-5, 5]^D$	-330	MS	Sh
	CEC13	$f10$ Rotated Ackley	$[-100, 100]^D$	-700	MN	Ro
		$f11$ Rotated Griewank	$[-100, 100]^D$	-500	MN	Ro
		$f12$ Rotated Expanded Griewank plus Rosenbrock	$[-100, 100]^D$	500	MN	Ro

To address the first two reasons, we apply the search equation of DE/current-to-rand, which is rotationally invariant [9], [19] for the employed bees. The employed bees use the following equation (5) instead of (2).

$$\vec{u}_i = \vec{x}_i + k_i \cdot (\vec{x}_{r1} - \vec{x}_i) + F' \cdot (\vec{x}_{r2} - \vec{x}_{r3}) \quad (5)$$

where $r1$, $r2$ and$r3$ are indexes of food sources randomly chosen and $r1 \neq r2 \neq r3 \neq i$. $k_{i,j}$ is a random value from a uniform distribution between 0 and 1. F' is a random value from a uniform distribution between 0 and 1, created once for each iteration.

For the onlooker bees, we apply a different DE variant for a

different search behavior. DE/best/1 provides a fast convergence on a wide range of characteristics of functions. DE/rand/2/dir [20] provides performances of similar quality with DE/rand/1 but is slightly faster to converge than DE/rand/1 on multimodal and separable functions [9]. Therefore, inspired by DE/best/1 and DE/rand/2/dir, the search is around the bee with the best fitness value and the objective function information is incorporated to guide the direction as in the following way:

$$v_{i,j} = x_{best,j} + F(x_{dest,j} - x_{src,j}) \quad (6)$$

where x_{best} denotes the bee with the currently best fitness value. x_{src} and x_{dest} are the bees chosen with random such that $f(x_{dest}) < f(x_{src})$, for a minimization problem. F is a random value from a uniform distribution between 0 and 1.

In addition, after the roulette wheel selection method is performed, the probability for the bee with best fitness (p_{best}) is set to 1.0 so as to guarantee that the best bee will be updated with the new onlooker equation (6).

All the modifications described here do not introduce additional algorithmic parameters, and hence the strength of the ABC algorithm is retained.

For the third weakness of equation (2) mentioned earlier, we do not make any improvement in this work. The common idea is to have more dimensions be altered in an iteration, controlled with a newly-introduced probability parameter. By this way the solution is generated more like the crossover operator of DE rather than the ABC algorithm, and hence we leave for further investigation in next work.

IV. Performance Evaluation

A. Experimentation Setup

Twelve benchmark functions are used in the following experiment to evaluate the performance of the proposed hybrid ABCDE algorithm. All these functions are scalable minimization problems widely used in literature and have various characteristics. Table I summarizes their brief characteristics and search ranges. The first six functions are basic unimodal and more difficult multimodal functions. The Sphere function ($f1$) is for testing an algorithm's convergence speed on a very simple function. Function 2 is irregular from being altered with the addition of noise. The Rosenbrock function ($f3$) has a very narrow valley from local optimum to global optimum. It is considered a multimodal function at a higher dimension than 3 [21]. The Rastrigin function ($f6$) contains a huge number of local optima dispersed throughout the search space.

Functions 7 to 12 are more difficult forms of selected basic functions by having axis rotation or their optimum shifted. Functions 7 to 9 are selected from the special session and competition held under the IEEE Congress on Evolutionary Computation (CEC) 2005 while functions 9 to 12 are selected from CEC 2013. Details of the benchmark set are given in the corresponding technical reports [22], [23].

The experiment is conducted on all these 12 functions at 30 dimensions (or the number of decision variables). The maximum number of function evaluations (MAXNFC) is set to 200000. All experiments were run 60 times independently. Mean, s.d., highest value and lowest values of the final fitness obtained from 60 runs by the proposed ABCDE are compared with those by original ABC as well as opposition-based DE (ODE) [24]. ODE is a state-of-the-art DE enhanced with the opposition-based learning in population initialization, generation jumping, and local improvement. Despite its simple implementation, performance of ODE was comprehensively proven using a large set of complex benchmark functions [24].

Population size (PS) is known to have some impact to the search diversity and on the performance of population-based algorithms. PS for ABCDE and ABC are set equally to 20 and 60. PS for a DE is recommended widely from $3{\cdot}D$ to $10{\cdot}D$ [8], [9]. In this experiment, we tested ODE at 60, 90, 120, 150 and 180 to investigate the results and found that the overall best results are obtained from PS = 60 and 90, and thus will only be reported in the next section. Other important parameters for ODE are as follows: CR = 0.85, F = 0.5, and jumping rate J_R = 0.3, as recommended [24]. The abandon limit for both ABC and ABCDE is set to $0.6 \cdot \text{PS}{\cdot}D$ as recommended [11].

All algorithms are implemented in Java 7 using NetBeans IDE 7.3 and executed on a computer running Windows 7 SP1 with an Intel i7 Quad Core 3.40 GHz.

B. Comparison Results and Discussions

Table II lists the means, standard deviations (s.d.), the lowest and the highest fitness obtained from running each algorithm for 60 times. Any values smaller than 1e-80 are reported as 0. The lowest mean and s.d. values indicate best performance of algorithm for that function and are highlighted in bold. Table III summarizes statistical ranking of final fitness values obtained for each benchmark function; a lower-number rank is better. To rank algorithms for each function, the two-tailed Wilcoxon signed-rank test, a well-known nonparametric statistical hypothesis test, is conducted to test the difference in final fitness values obtained from a pair of algorithms at 0.05 significant level. An algorithm j is ranked better than algorithm k (r_j<r_k) if the Wilcoxon signed-rank test result of algorithms j against k gets a p-value below 0.05. Two algorithms are ranked equally if the Wilcoxon signed-rank test result is not significant. Then the ranks of each algorithm are averaged by function groups: a group of six basic functions and a group of six complex functions (with shift and rotation) as well as all twelve functions, to obtain the Average Aggregated Ranks (AAR). A lower value of AAR means that such algorithm performs better. The last column (#1) in Table III indicates the number of functions, in which each algorithm gets the first rank. A higher value of #1 means the algorithm is better.

Fig. 1 and Fig. 2 display the average convergence graphs of all algorithms-PS's for each function. To avoid overcrowding in the graphs, the convergence graphs of ODE with PS = 60 (ODE-60) are omitted since it has a lower performance in overall compared to ODE-90.

Last column in Table II shows the running time (in seconds) for 60 runs of each algorithm as a comparison. It is clear that

ODE took a longer average time than ABC and ABCDE to finish its running (with MAXNFC) in every function. This is due to the overhead time for sorting in the ODE's opposition-based learning. In addition, we notice that on average ABCDE and ABC took about the same time to run.

1) Basic functions

From Table II and Fig. 1 (a) and (b), both ABCDE and ODE clearly outperforms original ABC on basic unimodal functions (*f*1 and *f*2). ABCDE with PS = 20 (ABCDE-20) has the fastest convergence in *f*1, but ODE wins in the case of noisy function (*f*2).

Functions *f*3 to *f*6 are more difficult multimodal functions. ABCDE clearly outperforms ODE and beats original ABC in *f*3 Rosenbrock and *f*6 Rastrigin functions which are very challenging. Every algorithm can achieve a minimum value of 0 for Griewank function. Although on average ODE-90 is better than all ABC variants in Ackley and Griewank functions, ODE-60 performs the worst. ABCDE performs better than ABC in both functions.

Considering the AAR columns in Table III, we can observe that ABCDE achieves the top two best AAR_{BAS}, i.e. 2.33 and 2.67 for ABCDE-20 and ABCDE-60, respectively.

From Fig. 1, we can see that ABCDE-20 converges fastest for 5 out of 6 functions. The improved solution generations of both employed bees and onlooker bees significantly accelerate their convergence, while the main structure of bee colony algorithm and its abandon limit help preserve the exploration capability.

2) Complex functions

For shifted functions (*f*7 – *f*9) according to Table III, both ABC and ABCDE perform equally in terms of rank regardless of population size and clearly beat ODE. For rotated functions (*f*10 – *f*12), the winners are ABC-60 and ABCDE-20 with equal AAR_{CPX} of 1.17.

From Fig. 2, the convergence of ODE is faster than other algorithms only in *f*11 Rotated Griewank function. However, ABCDE-20 converges the fastest for 3 out of 6 complex functions.

In summary for all twelve functions, ABCDE-20 achieved the best (lowest) AAR (= 1.75) as well as the highest number of #1 (= 7). This confirms the competitiveness of the proposed hybrid ABCDE.

TABLE II: BASIC STATISTICS OF THE RESULTS AND RUNNING TIME (IN SECONDS) ANY VALUES SMALLER THAN 1E-80 ARE REPORTED AS 0

#	Algo.	PS	Mean	s.d.	Lowest	Highest	Time
*f*1	ABC	20	5.924E-78	3.182E-77	0	2.456E-76	1.04
		60	2.883E-23	8.286E-23	1.212E-25	6.250E-22	1.08
	ABCDE	20	**0**	**0**	0	0	1.03
		60	1.165E-35	4.207E-35	2.131E-39	2.486E-34	1.05
	ODE	60	1.665E-04	1.274E-03	0	9.954E-03	5.02
		90	**0**	**0**	0	0	5.40
*f*2	ABC	20	5.486E-02	1.069E-02	2.977E-02	7.998E-02	6.88
		60	6.526E-02	1.449E-02	4.075E-02	1.123E-01	6.96
	ABCDE	20	2.903E-02	6.534E-03	1.201E-02	4.300E-02	7.01
		60	2.431E-02	4.746E-03	1.255E-02	3.737E-02	7.03
	ODE	60	**6.588E-04**	**2.848E-04**	3.475E-04	2.319E-03	14.02
		90	9.450E-04	2.841E-04	3.765E-04	1.598E-03	14.11
*f*3	ABC	20	0.260	0.303	2.72E-02	1.877	12.35
		60	0.438	1.192	4.74E-03	9.005	12.47
	ABCDE	20	0.494	1.472	7.58E-05	10.028	12.53
		60	**0.183**	**0.298**	5.67E-03	1.597	12.59
	ODE	60	27.143	0.529	2.54E+01	27.969	23.02
		90	26.805	0.416	25.595	27.689	23.12
*f*4	ABC	20	4.266E-11	2.915E-11	1.030E-11	1.299E-10	4.02
		60	3.946E-14	4.228E-15	2.887E-14	5.018E-14	4.10
	ABCDE	20	3.147E-14	2.817E-15	2.887E-14	3.952E-14	3.99
		60	3.183E-14	3.122E-15	2.176E-14	3.952E-14	4.03
	ODE	60	2.078E-06	1.239E-05	3.997E-15	9.386E-05	10.02
		90	**3.997E-15**	**3.155E-30**	3.997E-15	3.997E-15	10.08
*f*5	ABC	20	5.351E-11	3.366E-10	0	2.620E-09	3.92
		60	6.077E-09	4.029E-08	0	3.130E-07	4.02
	ABCDE	20	4.003E-09	1.080E-08	0	5.907E-08	4.02
		60	3.574E-08	1.994E-07	0	1.552E-06	4.05
	ODE	60	8.008E-04	2.790E-03	0	1.232E-02	10.02
		90	**0**	**0**	0	0	10.05
*f*6	ABC	20	2.43E-13	8.29E-13	0	6.01E-12	3.52
		60	**0**	**0**	0	0	3.59
	ABCDE	20	**0**	**0**	0	0	4.04
		60	**0**	**0**	0	0	4.13
	ODE	60	61.137	43.580	2.985	140.020	10.50
		90	79.979	42.117	6.965	135.869	10.52
*f*7	ABC	20	**-450**	**0**	-450	-450	0.53

#	Algo.	PS	Mean	s.d.	Lowest	Highest	Time
		60	**-450**	**0**	-450	-450	0.58
	ABCDE	20	**-450**	**0**	-450	-450	1.02
		60	**-450**	**0**	-450	-450	1.08
	ODE	60	-449.992	0.0403	-450	-449.715	5.52
		90	**-450**	**0**	-450	-450	5.55
*f*8	ABC	20	395.234	6.088	390.668	428.030	0.52
		60	399.310	13.346	390.037	448.613	0.55
	ABCDE	20	400.095	14.787	390.005	448.380	1.03
		60	**393.631**	**6.723**	390.049	427.358	1.09
	ODE	60	2.22E+06	1.51E+07	416.494	1.18E+08	5.11
		90	2.90E+04	2.13E+05	414.890	1.67E+06	5.50
*f*9	ABC	20	**-330**	**0**	-330	-330	4.00
		60	**-330**	**0**	-330	-330	4.07
	ABCDE	20	**-330**	**0**	-330	-330	4.01
		60	**-330**	**0**	-330	-330	4.09
	ODE	60	-254.038	43.933	-317.833	-187.514	10.01
		90	-246.656	42.790	-316.071	-179.829	10.50
*f*10	ABC	20	-679.029	0.053	-679.186	-678.932	55.93
		60	**-679.047**	**0.054**	-679.204	-678.954	57.60
	ABCDE	20	-679.042	0.045	-679.215	-678.972	56.01
		60	-679.041	0.051	-679.168	-678.953	57.69
	ODE	60	-679.044	0.056	-679.241	-678.941	76.94
		90	-679.032	0.051	-679.181	-678.939	78.97
*f*11	ABC	20	-493.331	1.757	-497.818	-488.195	15.13
		60	-498.937	0.248	-499.515	-498.415	15.47
	ABCDE	20	-499.535	0.281	-499.922	-498.764	15.44
		60	-497.398	0.686	-498.554	-495.624	15.59
	ODE	60	-499.290	0.664	-499.971	-497.192	23.70
		90	**-499.944**	**0.075**	-500.000	-499.483	24.15
*f*12	ABC	20	**18011.0**	**2628.9**	13160.6	22328.5	7.31
		60	20245.9	2985.1	13113.6	25769.6	7.81
	ABCDE	20	20352.9	3470.1	13079.9	27297.3	7.33
		60	23668.0	2780.6	18127.2	30102.9	7.95
	ODE	60	34353.0	24.6	34343.9	34485.2	14.29
		90	34348.4	2.5	34344.9	34358.3	14.31

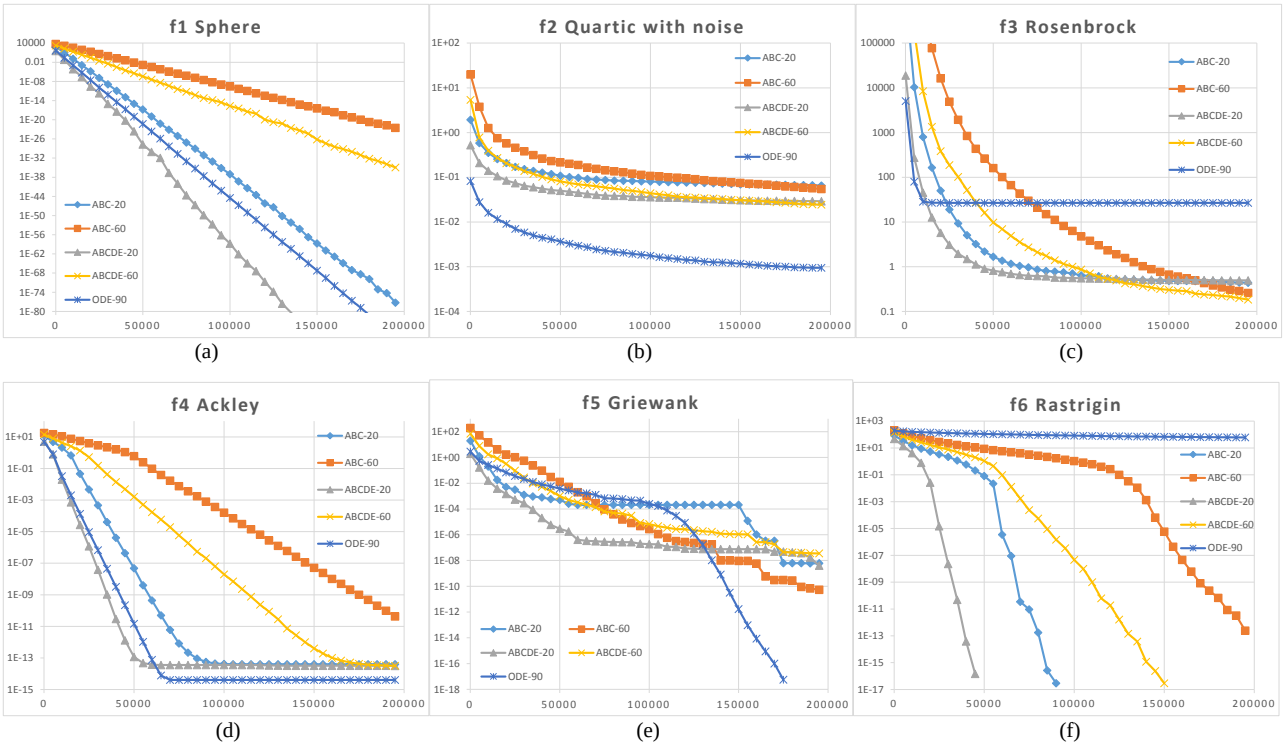

Fig. 1. Average convergence performance of ABC, ABCDE and ODE from running of 60 times for minimization of basic functions. The number after dash (-) indicates population size.

TABLE III: RANKS OF THE ALGORITHMS FOR EACH FUNCTION
THE LAST COLUMN INDICATES THE NUMBER OF FUNCTIONS THE ALGORITHM CAN ACHIEVE THE FIRST RANK

Algorithm	PS	Basic							Complex							*AAR*	#1
		*f*1	*f*2	*f*3	*f*4	*f*5	*f*6	AAR_{BAS}	*f*7	*f*8	*f*9	*f*10	*f*11	*f*12	AAR_{CPX}		
ABC	20	3	5	2	5	2	4	3.50	1	1	1	1	6	1	1.83	2.67	5
	60	5	5	4	2	3	1	3.33	1	1	1	1	1	2	**1.17**	2.25	6
ABCDE	20	1	3	4	2	3	1	**2.33**	1	1	1	1	1	2	**1.17**	**1.75**	**7**
	60	4	3	1	2	5	1	2.67	1	1	1	1	5	4	2.17	2.42	6
ODE	60	6	1	6	6	6	5	5.00	6	5	5	1	1	5	3.83	4.42	3
	90	1	2	6	4	1	5	3.17	1	5	6	1	1	5	3.17	3.17	5

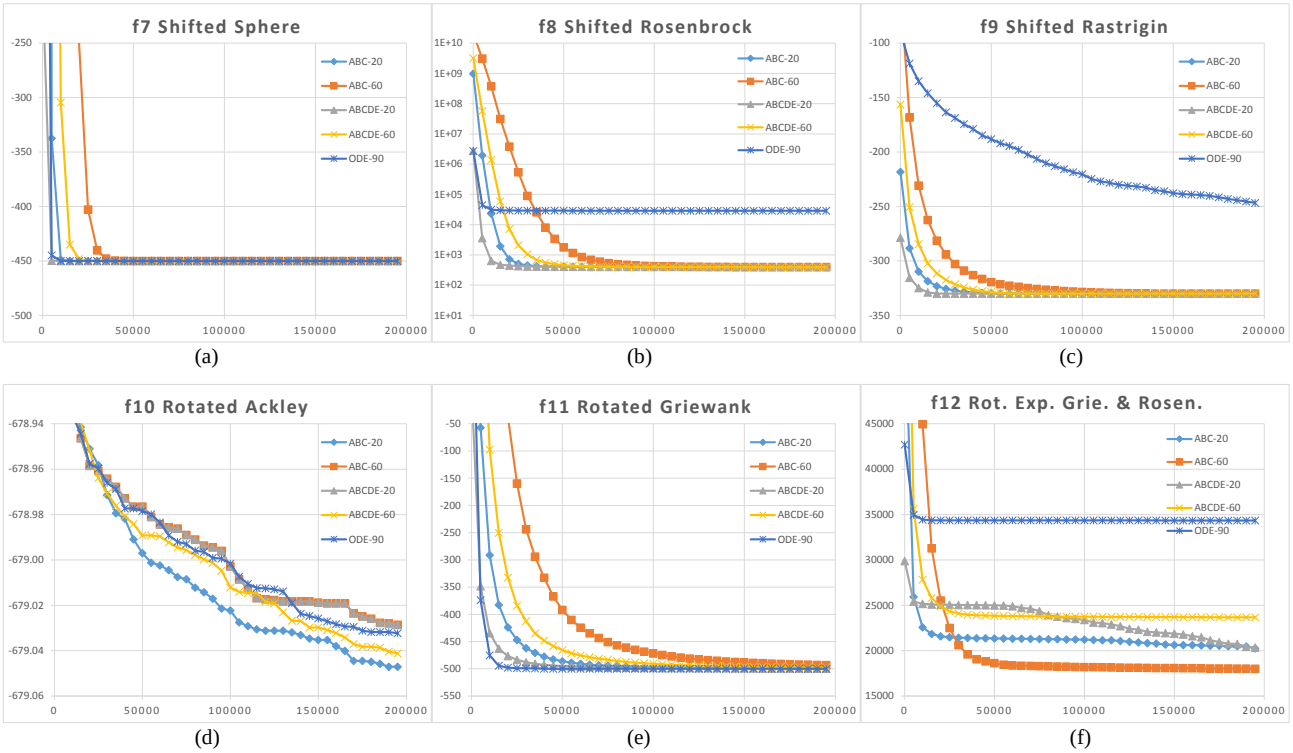

Fig. 2. Average convergence performance of ABC, ABCDE and ODE from running of 60 times for minimization of complex functions. The number after dash (-) indicates population size.

V. CONCLUSION

This paper proposes a hybrid Artificial Bee Colony algorithm with Differential Evolution, called ABCDE. The proposed ABCDE takes advantage of some advance DE's mutation strategies to speed up search convergence while the exploration capability is still maintained and no additional parameters are introduced. An experiment is conducted to evaluate the performance of ABCDE against original ABC and ODE, a highly competent DE, with twelve widely accepted benchmark of non-linear minimization problems. The experimental results indicate that the proposed ABCDE has a significant improved convergence speed not only on simple unimodal and multimodal functions but also on rotated and shifted functions. Future works include a more systematic hybridization of ABC with DE, PSO and other meta-heuristic algorithms with an application of optimizing neural networks for financial applications.

REFERENCES

[1] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Erciyes University, Kayseri, Turkey, Technical Report-TR06, 2005.
[2] D. Karaboga and B. Basturk, "On the performance of Artificial Bee Colony (ABC) algorithm," *Applied Soft Computing*, vol. 8, pp. 687–697, 2008.
[3] D. Karaboga and B. Akay, "A comparative study of Artificial Bee Colony algorithm," *Applied Mathematics and Computation*, vol. 214, pp. 108–132, 2009.
[4] S. Samanta and S. Chakraborty, "Parametric optimization of some non-traditional machining processes using Artificial Bee Colony algorithm," *Engineering Applications of Artificial Intelligence*, vol. 24, pp. 946–957, 2011.
[5] S. Sundar and A. Singh, "A swarm intelligence approach to early/tardy scheduling problem," *Swarm and Evolutionary Computation*, vol. 4, no. 1, pp. 25–32, 2012.
[6] A. R. Yildiz, "A new hybrid Artificial Bee Colony algorithm for robust optimal design and manufacturing," *Applied Soft Computing*, vol. 13, no. 5, pp. 2906–2912, 2013.
[7] P. Lu, J. Zhou, H. Zhang, R. Zhang, and C. Wang, "Chaotic differential bee colony optimization algorithm for dynamic economic dispatch problem with valve-point effects," *Electrical Power and Energy Systems*, vol. 62, pp. 130–143, 2014.
[8] R. Storn and K. Price, "Differential evolution — A simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.
[9] S. Das and P. N. Suganthan, "Differential evolution — A survey of the state-of-the-art," *IEEE Trans. Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, Feb. 2011.
[10] G. Zhu and S. Kwong, "Gbest-guided Artificial Bee Colony algorithm for numerical function optimization," *Applied Mathematics and Computation*, vol. 217, no. 7, pp. 3166–3173, Dec. 2010.

[11] W. Gao and S. Liu, "A modified Artificial Bee Colony algorithm," *Computers and Operation Research*, vol. 39, no. 3, pp. 687–697, Mar. 2012.
[12] H. Wang, Z. Wu, S. Rahnamayan, H. Sun, Y. Liu, and J.-S. Pan, "Multi-strategy ensemble Artificial Bee Colony algorithm," *Information Sciences*, vol. 279, pp. 587-603, 2014.
[13] F. Kang, J. J. Li, and Z. Y. Ma, "Rosenbrock Artificial Bee Colony algorithm for accurate global optimization of numerical functions," *Information Sciences*, vol. 12, pp. 3508–3531, 2011.
[14] D. Bose, S. Biswas, A. V. Vasilakos, and S. Laha, "Optimal filter design using an improved artificial bee colony algorithm," *Information Sciences*, vol. 281, pp. 443–461, 2014.
[15] M. S. Kıran and M. Gündüz, "A recombination-based hybridization of particle swarm optimization and artificial bee colony algorithm for continuous optimization problems," *Applied Soft Computing*, vol. 13, no. 4, pp. 2188–2203, Apr. 2013.
[16] W. Xiang, S. Ma, and M. An, "hABCDE: A hybrid evolutionary algorithm based on Artificial Bee Colony algorithm and differential evolution," *Applied Mathematics and Computation*, vol. 238, pp. 370–386, 2014.
[17] J. Yang, W. T. Li, X.-W. Shi, L. Xin, and J. F. Yu, "A hybrid ABC-DE algorithm and its application for time-modulated arrays pattern synthesis," *IEEE Transactions on Antennas and Propagation*, vol. 61, no. 11, pp. 5485-5495, Nov. 2013.
[18] X. Li and M. Yin, "Hybrid differential evolution with artificial bee colony and its application for design of a reconfigurable antenna array with discrete phase shifters," *Microwaves, Antennas & Propagation, IET*, vol. 6, no. 14, pp. 1573–1582, 2012.
[19] K. V. Price, "An introduction to Differential Evolution," in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds. London, U.K.: McGraw-Hill, 1999, pp. 79–108.
[20] V. Feoktistov and S. Janaqi, "Generalization of the strategies in Differential Evolution," in *Proc. 18th Parallel and Distributed Processing Symposium*, Apr. 2004, p. 165.
[21] Y. W. Shang and Y. H. Qiu, "A note on the extended Rosenbrock function," *Evolutionary Computation*, vol. 14, no. 1, pp. 119–126, March 2006.
[22] 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, Tech. Rep., IIT Kanpur, Kanpur, India, KanGAL Rep. #2005005, May 2005.
[23] J. J. Liang, B. Y. Qu, P. N. Suganthan, and A. G. Hernandez-Diaz, "Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization," Com. Intel. Lab., Zhengzhou University, Zhengzhou, China, Tech. Rep. 201212 and Nanyang Technological University, Singapore, Tech. Rep., 2013.
[24] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 64–79, Jan. 2008.

Chukiat Worasucheep obtained a degree of M.S. in computer science from Oregon State University, USA. and has been working with Applied Computer Science, Faculty of Science, King Mongkut University of Technology Thonburi, Bangkok, Thailand. His current research interests are computational intelligence, particularly in evolutionary computation and swarm intelligence for financial applications and engineering applications. His recent book is *Modern Operating systems* (in Thai) to be published in early 2015.