Artif Intell Rev
DOI 10.1007/s10462-011-9276-0

A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms

Pinar Civicioglu · Erkan Besdok

Abstract In this paper, the algorithmic concepts of the Cuckoo-search (CK), Particle swarm optimization (PSO), Differential evolution (DE) and Artificial bee colony (ABC) algorithms have been analyzed. The numerical optimization problem solving successes of the mentioned algorithms have also been compared statistically by testing over 50 different benchmark functions. Empirical results reveal that the problem solving success of the CK algorithm is very close to the DE algorithm. The *run-time complexity* and the required *function-evaluation number* for acquiring global minimizer by the DE algorithm is generally smaller than the comparison algorithms. The performances of the CK and PSO algorithms are statistically closer to the performance of the DE algorithm than the ABC algorithm. The CK and DE algorithms supply more robust and precise results than the PSO and ABC algorithms.

Keywords Cuckoo search algorithm · Particle swarm optimization · Differential evolution algorithm · Artificial bee colony algorithm

1 Introduction

Optimization is an applied science which explores the best values of the parameters of a problem that may take under specified conditions (Corne et al. 1999; Horst et al. 2000). Optimization, in its most simple way, aims to obtain the relevant parameter values which enable an objective function to generate the *minimum* or *maximum* value. The design of an optimization problem generally starts with the design of an objective function (Rashedi et al. 2009; Karaboga and Akay 2009a; Del Valle et al. 2008; Storn and Price 1997). The objective function must correctly define the related problem *mathematically* and clearly express the

P. Civicioglu
Department of Aircraft Electrics and Electronics, College of Aviation, Erciyes University, Kayseri, Turkey
e-mail: civici@erciyes.edu.tr

E. Besdok (✉)
Engineering Faculty, Department of Geomatics Engineering, Erciyes University, Kayseri, Turkey
e-mail: ebesdok@erciyes.edu.tr

Published online: 06 July 2011

relation between the parameters of the problem. Furthermore, if any pre-defined constraints are to be used in respect of the parameters of the related problem, these should be considered during the design of the optimization problem. Generally, the optimization problems are classified under many subtitles; *Linear Programming*, *Integer Programming*, *Quadratic Programming*, *Combinatorial Optimization* and *Metaheuristics* are the terms often used to classify the optimization methods (Del Valle et al. 2008).

The *classical* optimization methods frequently used in scientific applications consist of *hessian matrix* based methods (Min-Jea et al. 2009) and *gradient* based methods (Haupt 1995). In practice, it is required many of the classical optimization methods to comply with the structure of the objective function intended to be solved. However, if the *derivative* of the objective function cannot be calculated, it gets difficult to search the optimal solution via classical optimization means (Rashedi et al. 2009). It can be proved by mathematical methods that a solution obtained by using classical techniques is globally optimum (Nowak and Cirpka 2004). But however, it is common to use *metaheuristic algorithms* in solving non-differentiable nonlinear-objective functions the solution of which is either impossible or extremely difficult by using the classical optimization techniques (Karaboga and Akay 2009a; Nowak and Cirpka 2004; Clerc and Kennedy 2002; Trelea 2003; Dorigo et al. 1996; Ong et al. 2006; Storn and Price 1997; Yang and Deb 2010; Das et al. 2011; Yang 2005, 2009; Zhang et al. 2007). Since a method, which can be used to prove that a solution obtained by metaheuristic algorithms is the optimum solution, has not been proposed by the researchers yet, the solutions obtained via these methods are referred to as *sub-optimal* solutions (Rashedi et al. 2009; Ong et al. 2006; Storn and Price 1997; Yang and Deb 2010). The metaheuristic optimization algorithms that are most widely used in scientific applications are Genetic Algorithm (Deb et al. 2002), Particle swarm optimization algorithm (PSO) (Clerc and Kennedy 2002; Trelea 2003; Yoshida et al. 2000; Juang 2004), Differential evolution algorithm (DE) (Storn and Price 1997; Ferrante and Ville 2010; Price and Storn 1997; Storn 1999; Liu and Lampinen 2005; Ali and Torn 2004; Shahryar et al. 2008; Das and Suganthan 2009; Kaelo and Ali 2006; Swagatam et al. 2009; Janez et al. 2007), Artificial bee colony algorithm (Karaboga and Akay 2009a; Karaboga and Basturk 2007a,b; Janez et al. 2007; Karaboga 2009; Fei et al. 2009), Cuckoo Search Algorithm (Yang and Deb 2009; Deb et al. 2002), Gravitational Search Algorithm (Rashedi et al. 2009; Esmat et al. 2010, 2011; Duman et al. 2010; Chaoshun and Jianzhong 2011) Harmony Search Algorithm (Geem et al. 2001; Lee and Geem 2004, 2005; Mahdavi et al. 2007) and their derivatives.

Many metaheuristic algorithms use a *pattern matrix*, which includes random solutions of the related problem. Since this situation enables the performance of information exchange between the patterns, the success archived in the solution of the related problem prominently improves. The methods which predicate on the *collective search* of the patterns for the solutions of the related problem are generally known as the artificial swarm intelligence methods (Dorigo et al. 2000, 2004; Martinoli et al. 2004; Sousa et al. 2004; Mahamed and Mehrdad 2008; Karaboga and Akay 2009b). The basic definitions used in the development of metaheuristic algorithms are based on the basic concepts used in Pattern Recognition (such as *pattern matrix*, *pattern* and *attribute*). In the PSO algorithm, the *pattern matrix* is referred to as swarm and each pattern corresponds to an *artificial particle*. Similarly, each pattern corresponds to a nectar source in the ABC algorithm. A pattern is considered as an artificial nest in the Cuckoo algorithm. In the genetic algorithm, each pattern corresponds to an artificial chromosome and the pattern matrix is referred to as population. Many metaheuristic algorithms somehow use the basic *genetic rules* (i.e., *mutation*, *crossover*, *selection*, and *adaptation*) while developing the existing random solutions (Juang 2004; Ferrante and Ville 2010; Das and Suganthan 2009; Karaboga and Akay 2009b). This situation

significantly increases the success of the population based metaheuristic algorithms on searching the related search space in compliance with their purposes. As the metaheuristic algorithms recursively develop the *pattern matrix* at every iteration, the *attribute variety* of the pattern matrix decreases with the progressing iteration steps. The researchers have proposed various methods generally based on injecting new patterns or attributes to the pattern matrix to maintain *diversity*.

The metaheuristic optimization algorithms use two basic strategies while searching for the global optimum; *exploration* and *exploitation* (Rashedi et al. 2009). While the *exploration* process succeeds in enabling the algorithm to reach the best local solutions within the search space, the *exploitation* process expresses the ability to reach the global optimum solution which is likely to exist around the local solutions obtained. A metaheuristic algorithm must be able to rapidly converge to the global optimum solution of the related objective function. Furthermore, the *run-time* required by a metaheuristic algorithm to reach to global minimizer and the total calculation amount must be at acceptable levels for practical applications. The algorithmic structure of a metaheuristic algorithm is desired to be simple enough to allow for its easy adaptation to different problems. Also, it is desired that the metaheuristic algorithm has no algorithmic control parameters or very few algorithmic control parameters excluding the general ones (i.e., *size of population*, *total number of iterations*, *problem dimension*) of the population based optimization algorithms. If a metaheuristic algorithm has algorithmic control parameters, the related algorithm must not be too dependent on the initial values of the mentioned algorithmic control parameters.

Since a single metaheuristic optimization algorithm which can solve all optimization problems of different types and structures does not exist, the requirement to develop new metaheuristic optimization algorithms increasingly continues (Rashedi et al. 2009).

This paper is organized as follows: In Sect. 2, *Review of Algorithmic Concepts of the CK, PSO, DE and ABC Algorithms*, has been presented. In Sect. 3, *Experiments* have been expressed. Finally, in Sect. 4, *Conclusions* have been given.

2 Review of algorithmic concepts of the CK, PSO, DE and ABC algorithms

The general *random-walk* based system-equation of metaheuristic algorithms is given in Eq. 1 (Mersha and Dempe 2011);

$$v \leftarrow X_i + s\,\delta x \tag{1}$$

In Eq. 1, δx denotes the *step-size* and X subtends to a random solution of the related objective function where $X \in R^n$. $s \in R$ value is the *scaling factor* selected for δx values. To develop the X pattern, the decision rule defined in Eq. 2 is used where f is the objective function (Storn and Price 1997);

$$f(v) < f(X_i) \rightarrow X_i := v \tag{2}$$

Here, $X_i = [x_j | j = 1, 2, 3, \ldots, n]$ vector shows the ith pattern of the pattern matrix, $P = [X_i]$, which includes n dimensional m random solutions where $i = 1, 2, 3, \ldots, a, b, r_1, r_2, r_3, r_4, r_5, \ldots, m$.

The metaheuristic algorithms differ radically in terms of the methods they use to determine s and δx values of Eq. 1 (Karaboga and Akay 2009a; Storn and Price 1997; Clerc and Kennedy 2002; Trelea 2003; Dorigo et al. 1996; Das et al. 2011). The method to be used to determine the mentioned s and δx values directly specifies the development strategy of the x_j attributes. The simplest method followed to determine δx is to use another randomly

selected solution. For the condition of $r_1 \neq r_2 \neq r_3$, let X_{r_1} and X_{r_2} correspond to the random solutions of the related objective function. In this case, if $\delta x_{r_1,r_2}$ is defined as,

$$\delta x_{r_1,r_2} = X_{r_1} - X_{r_2} \tag{3}$$

thus, the generalized system-equation of the *randomized mutation operator* of the *DE/rand/1* algorithm (Storn and Price 1997; Price and Storn 1997) is achieved as,

$$v \leftarrow X_{r_3} + s\,(X_{r_1} - X_{r_2}) \tag{4}$$

The differences in use of the related system-equations is quite important in comparison of metaheuristic optimization algorithms. The DE algorithm uses a considerably effective *adaptive-mutation* strategy together with the related system-equations (Storn and Price 1997; Ferrante and Ville 2010; Price and Storn 1997; Storn 1999; Das and Suganthan 2009). This case is the most significant difference of the DE algorithm from the standard genetic algorithm. Two detailed survey papers on DE based optimization algorithms are given in (Ferrante and Ville 2010; Das and Suganthan 2009).

In the *DE/rand/1* mutation strategy, s scaling factor is shown by F. In this case, the generalized system-equation of the DE algorithm is defined by Eq. 5 where $F^* \in R^n$,

$$v \leftarrow X_{r_3} + F^* \otimes (X_{r_1} - X_{r_2}) \tag{5}$$

In the original ABC algorithm, s scaling factor is shown by ϕ (Karaboga and Akay 2009a). In this case, the generalized system-equation of the ABC algorithm can be defined by Eq. 6 where $\phi^* \in R^n$,

$$v \leftarrow X_{r_1} + \phi^* \otimes (X_{r_1} - X_{r_2}) \tag{6}$$

In Eqs. 5 and 6, $\otimes$ denotes the *Hadamard* multiplication operator. In lots of engineering problems, the *dimension of the problem* is given as $n \gg 1$. On condition that $h \in Z^+$ and $h \in [1\ n]$ is a *random integer number*, if a random solution of any problem is shown by $X = [x_1,\ x_2,\ x_3,\ \ldots, x_h, \ldots,\ x_n]$, Eqs. 5 and 6 can be generalized to the differential solution components as specified below. Thus, the standard system-equation of the related DE strategy (Eq. 7) and the standard system-equation of the ABC algorithm (Eq. 8) are obtained.

$$v \leftarrow X_{r_1,h} + F\,(X_{r_2,h} - X_{r_3,h}) \tag{7}$$

$$v \leftarrow X_{r_1,h} + \phi(X_{r_1,h} - X_{r_2,h}) \tag{8}$$

The researchers have proposed many different strategies to be used in the determination of the value of *scale factor*. Generally, while the real-valued number range $s \in [0\ \ 1]$ is used in the DE algorithm and its derivatives (Ferrante and Ville 2010; Das and Suganthan 2009), the real-valued number range $s \in [-1\ \ 1]$ is used in the ABC algorithm and its variants (Karaboga and Akay 2009a). Different researchers have proposed different methods for the determination of F value in the DE algorithm (Karaboga and Akay 2009a; Das and Suganthan 2009; Price et al. 2005).

A metaheuristic search algorithm may also use a strategy based on using the solution of the pattern which provides the best solution amongst the pattern matrix components while attempting to develop a random solution. In this case, the system-equation of Eq. 9 is obtained by generalizing Eq. 1 as;

$$v_{r_1,h} \leftarrow X_{r_1,h} + \phi_{r_1,h}\delta x_1 + \psi_{r_1,h}\delta x_2 \tag{9}$$

Table 1 The Benchmark functions used to examine the numerical optimization problem solving successes of the CK, PSO, DE and ABC algorithms and features of these functions

FNC	NAME	TYPE	LOW	UP	DIM
F1	FOXHOLES	MS	−65.536	65.536	2
F2	GOLDSTEINPRICE	MN	−2	2	2
F3	PENALIZED	MN	−50	50	30
F4	PENALIZED2	MN	−50	50	30
F5	ACKLEY	MN	−32	32	30
F6	BEALE	UN	−4.5	4.5	5
F7	BOHACHECSKY1	MS	−100	100	2
F8	BOHACHECSKY2	MN	−100	100	2
F9	BOHACHECSKY3	MN	−100	100	2
F10	BOOTH	MS	−10	10	2
F11	BRANIN	MS	−5	10	2
F12	COLVILLE	UN	−10	10	4
F13	DIXONPRICE	UN	−10	10	30
F14	EASOM	UN	−100	100	2
F15	FLETCHER	MN	−3.1416	3.1416	2
F16	FLETCHER	MN	−3.1416	3.1416	5
F17	FLETCHER	MN	−3.1416	3.1416	10
F18	GRIEWANK	MN	−600	600	30
F19	HARTMAN3	MN	0	1	3
F20	HARTMAN6	MN	0	1	6
F21	KOWALIK	MN	−5	5	4
F22	LANGERMANN	MN	0	10	2
F23	LANGERMANN	MN	0	10	5
F24	LANGERMANN	MN	0	10	10
F25	MATYAS	UN	−10	10	2
F26	MICHALEWICS	MS	0	3.1416	2
F27	MICHALEWICS	MS	0	3.1416	5
F28	MICHALEWICS	MS	0	3.1416	10
F29	PERM	MN	−4	4	4
F30	POWELL	UN	−4	5	24
F31	POWERSUM	MN	0	4	4
F32	QUARTIC	US	−1.28	1.28	30
F33	RASTRIGIN	MS	−5.12	5.12	30
F34	ROSENBROCK	UN	−30	30	30
F35	SCHAFFER	MN	−100	100	2
F36	SCHWEFEL	MS	−500	500	30
F37	SCHWEFEL_1_2	UN	−100	100	30
F38	SCHWEFEL_2_22	UN	−10	10	30
F39	SHEKEL10	MN	0	10	4
F40	SHEKEL5	MN	0	10	4

Table 1 continued

FNC	NAME	TYPE	LOW	UP	DIM
F41	SHEKEL7	MN	0	10	4
F42	SHUBERT	MN	−10	10	2
F43	SIXHUMPCAMELBACK	MN	−5	5	2
F44	SPHERE2	US	−100	100	30
F45	STEP2	US	−100	100	30
F46	STEPINT	US	−5.12	5.12	5
F47	SUMSQUARES	US	−10	10	30
F48	TRID	UN	−36	36	6
F49	TRID	UN	−100	100	10
F50	ZAKHAROV	UN	−5	10	10

By rearranging Eq. 9, the standard system-equation of a modified ABC algorithm (Zhua and Kwongb 2010) is obtained as,

$$v_{r_1,h} \leftarrow X_{r_1,h} + \phi_{r_1,h}(X_{r_1,h} - X_{r_2,h}) + \psi_{r_1,h}(Gbest_h - X_{r_1,h}) \tag{10}$$

where $Gbest$ is the best pattern found by the *pattern matrix*.

The ABC algorithm investigates the search space by using a stochastic structured algorithmic decision rule together with its own system-equations. A substantially detailed survey paper regarding the *Bee Search* methods is provided in (Karaboga and Akay 2009b).

Since the metaheuristic algorithms recursively develop the *pattern matrix* components by using the variations of Eqs. 1 and 2, as long as Eq. 2 condition is realized, the X solution continues to search for a better solution within the search space. Consequently, a best solution, $Pbest$, which is achieved at the current iteration step for an X solution, always exists. In this case, if the general system-equations Eqs. 1 and 9 are rearranged, the system-equations of Eq. 11 are obtained.

$$\begin{aligned} v_{r_1+1} &\leftarrow \omega v_{r_1} + C_1\, rand \otimes (Pbest_{r_1} - X_{r_1}) + C_2\, rand \otimes (Gbest - X_{r_1}) \\ X_{r_1+1} &= X_{r_1} + v_{r_1+1} \end{aligned} \tag{11}$$

where the *acceleration constants* $C_1,\ C_2$ and the *inertia weight* ω are predefined by the user and $r_1,\ r_2$ are the uniformly generated random numbers in the range of [0 1]. The $Pbest$ value in Eq. 11 denotes the best solution found by the pattern r_1. The system-equations of Eq. 11 are the most general system-equations of the PSO algorithm (Del Valle et al. 2008; Clerc and Kennedy 2002; Trelea 2003). For ω value used in the velocity system-equation given in Eq. 11, the researchers have suggested the use of different strategies; $\omega = 0.60$ (Karaboga and Akay 2009a), $\omega = rand$ (Del Valle et al. 2008). PSO and its variations can be found in (Del Valle et al. 2008; Clerc and Kennedy 2002; Trelea 2003; Yoshida et al. 2000; Juang 2004; Dorigo et al. 2004; Martinoli et al. 2004).

If the general system-equation given in Eq. 1 is generalized and rearranged, the system-equation of Eq. 12 is obtained.

$$v \leftarrow v + K \cdot (v - X_{best}) \tag{12}$$

Table 2 The *MeanOpt* values of the CK, PSO, DE and ABC algorithms

	CK	PSO	DE	ABC
F1	0.9980038377944496	0.9980038377944496	0.9980038377944496	0.9980038377944496
F2	2.9999999999999201	2.9999999999999205	2.9999999999999210	2.9999999999999223
F3	0.0000000000000000	0.0207338040500992	0.0000000000000000	0.0000000000000003
F4	0.0000000000000000	0.0016481048753826	0.0000000000000000	0.0000000000000003
F5	0.0000000000000044	0.0000000000000080	0.0000000000000044	0.0000000000000300
F6	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000008
F7	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F8	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F9	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000001
F10	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F11	0.3978873577297382	0.3978873577297382	0.3978873577297382	0.3978873577297382
F12	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0216852782176882
F13	0.6666666666666663	32.7000000000000030	0.6666666666666663	0.0000000000000023
F14	−0.3000000000000000	−1.0000000000000000	−1.0000000000000000	−1.0000000000000000
F15	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F16	0.0000000000000000	38.6090558857124510	0.0000000000000000	0.0003010792427653
F17	0.0118567330310422	485.5824526409508600	0.0000000000000000	2.2456643604266411
F18	0.0000000000000000	0.0092269113767599	0.0011086678563541	0.0000000000000000
F19	−3.8627821478207536	−3.8627821478207536	−3.8627821478207536	−3.8627821478207536
F20	−3.3219951715842422	−3.2684932669902809	−3.3041612033862551	−3.3219951715842422
F21	0.0003074859878056	0.0003074859878056	0.0003074860026057	0.0003380996738077
F22	−1.0809384421344377	−1.0809384421344377	−1.0809384421344377	−1.0809384421344377
F23	−1.4999992233524955	−1.4407993010172435	−1.4999992233524955	−1.4999992233524937
F24	−1.4034346917831928	−0.9849435323968848	−1.2806273502926973	−1.0222295172647704
F25	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000

Table 2 continued

	CK	PSO	DE	ABC
F26	−1.8210436836776824	−1.8210436836776824	−1.8210436836776824	−1.8210436836776824
F27	−4.6934684519571146	−4.6769774171754843	−4.6934684519571146	−4.6934684519571137
F28	−9.6601517156413479	−9.5127918086875667	−9.6184137190412695	−9.6601517156413479
F29	0.0000528002823306	0.0013823741086214	0.0051435833427266	0.0204750133573142
F30	0.0000001506204011	0.0000517665237642	0.0000000000000000	0.0004486249518229
F31	0.0000029552286077	0.0001616061328810	0.0001055710319695	0.0022337034076446
F32	0.0021879618606472	0.0047745343217586	0.0004074540652956	0.0129825734203320
F33	1.2828840539586595	28.0080704889407630	15.3906006782917850	0.0000000000000000
F34	0.0000000000000000	2.3638709258458954	0.0000000000000000	0.0609538747914428
F35	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F36	−12,505.4587899889370000	−8,927.9796708997965000	−11,939.3023972921910000	−12,569.4866181730160000
F37	0.0000000000000000	0.0000000007389222	0.0000000000000000	47.4978770095106240
F38	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000004
F39	−10.5364098166920460	−10.5364098166920460	−10.5364098166920460	−10.5364098166920450
F40	−10.1531996790582270	−10.1531996790582270	−10.1531996790582270	−10.1531996790582270
F41	−10.4029405668186680	−10.4029405668186680	−10.4029405668186680	−10.4029405668186680
F42	−186.7309088310239800	−186.7309088235984100	−186.7309088310239500	−186.7309088310239500
F43	−1.0316284534898774	−1.0316284534898774	−1.0316284534898774	−1.0316284534898774
F44	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000004
F45	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F46	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F47	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000004
F48	−50.0000000000001920	−50.0000000000001920	−50.0000000000002130	−49.9999999999999080
F49	−210.0000000000028700	−210.0000000000023300	−210.0000000000036400	−209.9999999999536200
F50	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000488

Table 3 The standard deviation values (*STD*) of the *MeanOpt* values

	CK	PSO	DE	ABC
F1	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F2	0.0000000000000009	0.0000000000000009	0.0000000000000015	0.0000000000000013
F3	0.0000000000000000	0.0425448682235546	0.0000000000000000	0.0000000000000000
F4	0.0000000000000000	0.0040251945388060	0.0000000000000000	0.0000000000000001
F5	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000025
F6	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000002
F7	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F8	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F9	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000001
F10	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F11	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F12	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0145332859252275
F13	0.0000000000000005	98.5965196353833020	0.0000000000000002	0.0000000000000005
F14	0.4701623459816272	0.0000000000000000	0.0000000000000000	0.0000000000000000
F15	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F16	0.0000000000000000	75.6663550155301580	0.0000000000000000	0.0011975521566063
F17	0.0506459600277799	703.1961620442745000	0.0000000000000000	1.4295906487767676
F18	0.0000000000000000	0.0105600444455998	0.0034356399375174	0.0000000000000000
F19	0.0000000000000023	0.0000000000000023	0.0000000000000023	0.0000000000000021
F20	0.0000000000000009	0.0606851636993409	0.0435562035329307	0.0000000000000005
F21	0.0000000000000000	0.0000000000000000	0.0000000000661880	0.0000235986072033
F22	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000002
F23	0.0000000000000007	0.1822135162832727	0.0000000000000007	0.0000000000000008
F24	0.2378187914151629	0.3407357990777687	0.3550649611006146	0.3223365744251494
F25	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F26	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F27	0.0000000000000018	0.0207220031101778	0.0000000000000018	0.0000000000000011
F28	0.0000000000000004	0.1289853405351259	0.0311959408647348	0.0000000000000017
F29	0.0000448197610402	0.0013347446743983	0.0005198266693245	0.0149596386587757
F30	0.0000000545404714	0.0000216470469136	0.0000000000000000	0.0000666092006811
F31	0.0000029262253023	0.0002478983088702	0.0001263865188983	0.0015077135883540
F32	0.0005667706273508	0.0018352361800625	0.0001466487103078	0.0024604768149038
F33	1.0406975037000643	7.8693037959886158	4.9561833894692935	0.0000000000000000
F34	0.0000000000000000	3.3250228546795992	0.0000000000000000	0.0994341785385808
F35	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F36	76.4021080525745330	911.9891017652336100	203.7887007036597100	0.0000000000018662
F37	0.0000000000000000	0.0000000024333530	0.0000000000000000	23.1645463822441510
F38	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000001
F39	0.0000000000000030	0.0000000000000026	0.0000000000000022	0.0000000000000028
F40	0.0000000000000036	0.0000000000000036	0.0000000000000036	0.0000000000000027
F41	0.0000000000000018	0.0000000000000018	0.0000000000000018	0.0000000000000035

Table 3 continued

	CK	PSO	DE	ABC
F42	0.0000000000000000	0.0000000332053694	0.0000000000000332	0.0000000000000160
F43	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F44	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000001
F45	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F46	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F47	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000001
F48	0.0000000000000237	0.0000000000000219	0.0000000000000351	0.0000000000000818
F49	0.0000000000003332	0.0000000000004644	0.0000000000000000	0.0000000000100630
F50	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000430

In this case, the strategy for the determination of K value gains importance in order to efficiently investigate the search space. In the CK algorithm, K value is achieved through a substantially complex random-walk strategy (Yang and Deb 2009, 2010).

2.1 Cuckoo search algorithm

The Cuckoo search (CK) algorithm (Yang and Deb 2009, 2010), which is a population based stochastic global search algorithm, has been proposed by Yang and Deb in (2009). In the CK algorithm, a pattern corresponds to a nest and similarly each individual attribute of the pattern corresponds to a Cuckoo-egg. The general system-equation of the CK algorithm is based on the general system-equation of the random-walk algorithms, which is given in Eq. 13;

$$X_{g+1;i} = X_{g;i} + \alpha \otimes levy(\lambda) \tag{13}$$

where g indicates the number of the current generation ($g = 1, 2, 3, \ldots,$ maxcycle and maxcycle denotes the predetermined maximum generation number). In the CK algorithm, the initial values of the jth attributes of the ith pattern, $P_{g=0;i} = [x_{g=0;j,i}]$, have been determined by using Eq. 14,

$$X_{g=0;j,i} = rand \cdot (up_i - low_i) + low_i \tag{14}$$

where low_i and up_i are the *lower* and *upper* search-space limits of jth attributes, respectively. The CK algorithm controls the boundary conditions in each computation steps. Therefore, when the value of an attribute overflows the allowed search space limits, then the value of the related attribute is updated with the value of the closer limit value to the related attribute. Before starting to iterative search process, the CK algorithm detects the most successful pattern as X_{best} pattern. The iterative evolution phase of the pattern matrix begins with the detection step of the ϕ by using Eq. 15;

$$\phi = \left(\frac{\Gamma(1+\beta) \cdot \sin(\pi \cdot \beta/2)}{\Gamma\left(\left(\frac{1+\beta}{2}\right) \cdot \beta \cdot 2^{\frac{\beta-1}{2}}\right)} \right)^{1/\beta} \tag{15}$$

In the standard software implementation of the CK algorithm (Yang and Deb 2010), using $\beta = 1.50$ has been advised. In Eq. 15, the Γ denotes *gamma function*. The evolution phase

Table 4 The *BestOpt* values of the CK, PSO, DE and ABC algorithms

	CK	PSO	DE	ABC
F1	0.9980038377944496	0.9980038377944496	0.9980038377944496	0.9980038377944496
F2	2.9999999999999192	2.9999999999999192	2.9999999999999183	2.9999999999999210
F3	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000002
F4	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000002
F5	0.0000000000000044	0.0000000000000080	0.0000000000000044	0.0000000000000222
F6	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000004
F7	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F8	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F9	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F10	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F11	0.3978873577297382	0.3978873577297382	0.3978873577297382	0.3978873577297382
F12	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000874954660480
F13	0.6666666666666665	0.6666666666666665	0.6666666666666665	0.0000000000000014
F14	−1.0000000000000000	−1.0000000000000000	−1.0000000000000000	−1.0000000000000000
F15	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F16	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000005
F17	0.0000000000000000	0.0000000000000001	0.0000000000000000	0.4151794271832446
F18	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F19	−3.8627821478207558	−3.8627821478207558	−3.8627821478207558	−3.8627821478207558
F20	−3.3219951715842431	−3.3219951715842431	−3.3219951715842431	−3.3219951715842426
F21	0.0003074859878056	0.0003074859878056	0.0003074859878056	0.0003101288701621
F22	−1.0809384421344377	−1.0809384421344377	−1.0809384421344377	−1.0809384421344377
F23	−1.4999992233524948	−1.4999992233524948	−1.4999992233524948	−1.4999992233524946
F24	−1.5000000000003775	−1.5000000000003775	−1.5000000000003775	−1.4999952788077173
F25	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000

Table 4 continued

	CK	PSO	DE	ABC
F26	−1.8210436836776824	−1.8210436836776824	−1.8210436836776824	−1.8210436836776824
F27	−4.6934684519571128	−4.6934684519571128	−4.6934684519571128	−4.6934684519571128
F28	−9.6601517156413497	−9.6601517156413479	−9.6601517156413497	−9.6601517156413497
F29	0.0000000085822797	0.0000088339092326	0.0047965266340616	0.0011011688503096
F30	0.0000000797695218	0.0000211010308969	0.0000000000000000	0.0002717789268690
F31	0.0000000509883999	0.0000000000427674	0.0000000888051571	0.0003305515663585
F32	0.0013909553459754	0.0022391455486327	0.0001538851948172	0.0059038261927937
F33	0.0003806178377204	13.9294167236679410	7.9596724567463761	0.0000000000000000
F34	0.0000000000000000	0.0000700595196601	0.0000000000000000	0.0002227389995635
F35	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F36	−12,569.4866180428830000	−10,426.9771785279660000	−12,233.9086141680040000	−12,569.4866181730140000
F37	0.0000000000000000	0.0000000000001282	0.0000000000000000	10.7576179839792520
F38	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000003
F39	−10.5364098166920500	−10.5364098166920500	−10.5364098166920500	−10.5364098166920480
F40	−10.1531996790582310	−10.1531996790582310	−10.1531996790582310	−10.1531996790582310
F41	−10.4029405668186660	−10.4029405668186660	−10.4029405668186660	−10.4029405668186660
F42	−186.7309088310239800	−186.7309088310239800	−186.7309088310239800	−186.7309088310239800
F43	−1.0316284534898774	−1.0316284534898774	−1.0316284534898774	−1.0316284534898774
F44	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000003
F45	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F46	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F47	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000003
F48	−50.0000000000002270	−50.0000000000001710	−50.0000000000002270	−50.0000000000000570
F49	−210.0000000000036400	−210.0000000000027300	−210.0000000000036400	−209.9999999999763500
F50	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000083

of the X_i pattern begins by defining the donor vector υ, where $\upsilon = X_i$. After this step, the required $stepsize$ value has been computed by using Eq. 16;

$$stepsize_j = 0.01 \cdot \left(\frac{u_j}{\upsilon_j}\right)^{1/\beta} \cdot (\upsilon - X_{best}) \tag{16}$$

where $u = \phi \cdot randn\,[D]$ and $\upsilon = randn\,[D]$. The $randn[D]$ function generates a uniform integer between [1 D]. In the next step of the CK algorithm, the donor pattern υ is randomly mutated by using Eq. 17;

$$\upsilon := \upsilon + stepsize_j \cdot randn\,[D] \tag{17}$$

The update process of the X_{best} pattern in the CK algorithm is defined by Eq. 18.

$$X_{best} \leftarrow f\,(X_{best}) \leq f\,(X_i) \tag{18}$$

The unfeasible patterns are manipulated by using the crossover operator given in Eq. 19;

$$\upsilon_i := \begin{cases} X_i + rand \cdot (X_{r_1} - X_{r_2}) & rand_i > p_0 \\ X_i & else \end{cases} \tag{19}$$

In the last step of the iterative computations, a simple rule given in Eq. 2 has been used for evolution of the pattern X_i. The algorithmic control parameters of the CK algorithm are the *scale factor* (β) and *mutation probability value* (p_0). In this paper, $\beta = 1.50$ and $p_0 = 0.25$ have been used as in (Yang and Deb 2010).

2.2 Differential evolution algorithm

The Differential evolution (DE) algorithm is a population-based, heuristic evolutionary optimization algorithm developed for the solution of *real-valued* numerical optimization problems (Storn and Price 1997; Price and Storn 1997; Storn 1999; Price et al. 2005). It is a very effective global search algorithm with a quite simple mathematical structure. The DE algorithm uses the *mutation*, *crossover*, and *selection* strategies of the genetic algorithm. The most important difference of the DE algorithm from the standard genetic algorithm is the strong mutation strategies it has (Storn and Price 1997; Price and Storn 1997). The DE algorithm can use two kinds of crossover schemes (i.e., *exponential* and *binomial*) (Storn and Price 1997; Ferrante and Ville 2010; Price and Storn 1997; Storn 1999; Liu and Lampinen 2005; Shahryar et al. 2008). In the standard DE algorithm, five different mutation strategies can be used with one of the two different crossover methods (Shahryar et al. 2008; Das and Suganthan 2009; Kaelo and Ali 2006; Swagatam et al. 2009; Janez et al. 2007; Price et al. 2005; Vesterstrom and Thomsen 2004; Bin et al. 2010). Therefore, the standard DE algorithm has ten different options to define the algorithmic structure to be used. The standard mutation strategies implemented in the original DE algorithm (Storn and Price 1997; Price and Storn 1997; Price et al. 2005; Das and Suganthan 2009; Swagatam et al. 2009) are given in Eq. 20;

$$\begin{aligned}
DE/rand/1 \quad & \upsilon = X_{r_1} + F \cdot (X_{r_2} - X_{r_3}) \\
DE/rand/2 \quad & \upsilon = X_{r_1} + F \cdot (X_{r_2} - X_{r_3}) + F \cdot (X_{r_4} - X_{r_5}) \\
DE/best/1 \quad & \upsilon = X_{best} + F \cdot (X_{r_1} - X_{r_2}) \\
DE/best/2 \quad & \upsilon = X_{best} + F \cdot (X_{r_1} - X_{r_2}) + F \cdot (X_{r_3} - X_{r_4}) \\
DE/target\text{-}to\text{-}best/1 \quad & \upsilon = X_i + F \cdot (X_{best} - X_i) + F \cdot (X_{r_1} - X_{r_2})
\end{aligned} \tag{20}$$

Table 5 Comparison of the performances of the CK, PSO, DE and ABC algorithms for the statistical parameters of *MeanOpt*

	CK	PSO	DE	ABC
SAME	41 (for the functions of: F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F15, F16, F18, F19, F20, F21, F22, F23, F24, F25, F26, F27, F28, F29, F31, F34, F35, F37, F38, F39, F40, F41, F42, F43, F44, F45, F46, F47, F50)	26 (for the functions of: F1, F6, F7, F8, F9, F10, F11, F12, F14, F15, F19, F21, F22, F25, F26, F35, F38, F39, F40, F41, F43, F44, F45, F46, F47, F50)	38 (for the functions of: F1, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F14, F15, F16, F17, F19, F22, F23, F25, F26, F27, F30, F32, F34, F35, F37, F38, F39, F40, F41, F43, F44, F45, F46, F47, F48, F49, F50)	23 (for the functions of: F1, F7, F8, F10, F11, F13, F14, F15, F18, F19, F20, F22, F25, F26, F28, F33, F35, F36, F40, F41, F43, F45, F46)
DIFF.	9 (for the functions of: F13, F14, F17, F30, F32, F33, F36, F48, F49)	24 (for the functions of: F2, F3, F4, F5, F13, F16, F17, F18, F20, F23, F24, F27, F28, F29, F30, F31, F32, F33, F34, F36, F37, F42, F48, F49)	12 (for the functions of: F2, F13, F18, F20, F21, F24, F28, F29, F31, F33, F36, F42)	27 (for the functions of: F2, F3, F4, F5, F6, F9, F12, F16, F17, F21, F23, F24, F27, F29, F30, F31, F32, F34, F37, F38, F39, F42, F44, F47, F48, F49, F50)

SAME The number of benchmark functions that the CK, PSO, DE and ABC algorithms provide equal values for the *minimum MeanOpt* parameter, *DIFF.* The number of benchmark functions that the CK, PSO, DE and ABC algorithms provide different values from the *minimum MeanOpt*

There are many mutation algorithms developed to be used with the DE algorithm. The most common one used in the literature is the *DE/rand/1/bin* algorithm which basically intends to search the global optimum by using three elements randomly selected from the population (Storn and Price 1997; Price and Storn 1997; Price et al. 2005). The weighted difference of two of the three randomly selected solutions is added to the third solution to obtain the donor solution, which is then mutated with a 4th randomly selected solution, and if the objective function value of the obtained mutant solution provides a better solution than the objective function value of the selected original 4th solution, the mutant solution is selected instead of the mentioned 4th solution within the next population. The detailed performed test results have shown that *DE/rand/1/bin* is excessively susceptible to the *size of population*, *total number of iterations*, *crossover value* (CR) and the *weighting value* (F).

The DE algorithm has three algorithmic control parameters; which are the *crossover value* $Cr \in [0\ 1]$, the *scale factor* $F \in [0\ 2]$, and the *size of population* value NP (Storn and Price 1997; Price et al. 2005; Das and Suganthan 2009; Swagatam et al. 2009). Taking the problem size as D, it is suggested that the $3D \leq NP \leq 10D$ value is used for the DE algorithm. In solution of many engineering problems, it has been suggested to use either $F = 0.60$ (Karaboga and Akay 2009a) or $F = 0.5 + (1 - rand)$ (Das and Suganthan 2009). For *DE/rand/1/bin* used in this paper, the values of parameters have been used as follows: $CR = 0.5$, $F = 0.90$ as in (Karaboga and Akay 2009a; Das and Suganthan 2009; Corne et al. 1999).

The problem solving success of DE algorithm is affected directly by the *mutation*, *size of population* and the *crossover* strategies used. The success of the DE algorithm in solving a numeric optimization problem is rather sensitive to the initial values of the Cr, F and NP (Das and Suganthan 2009; Kaelo and Ali 2006; Swagatam et al. 2009; Janez et al. 2007;

Table 6 Comparison of the performances of the CK, PSO, DE and ABC algorithms for the statistical parameters of *STD*

	CK	PSO	DE	ABC
SAME	36 (for the functions of: F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F15, F16, F18, F21, F22, F23, F24, F25, F26, F28, F29, F31, F34, F35, F37, F38, F41, F42, F43, F44, F45, F46, F47, F50)	26 (for the functions of: F1, F2, F5, F6, F7, F8, F9, F10, F11, F12, F14, F15, F21, F22, F25, F26, F35, F38, F41, F43, F44, F45, F46, F47, F48, F50)	35 (for the functions of: F1, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, F22, F23, F25, F26, F30, F32, F34, F35, F37, F38, F39, F41, F43, F44, F45, F46, F47, F49, F50)	21 (for the functions of: F1, F3, F7, F8, F10, F11, F14, F15, F18, F19, F20, F25, F26, F27, F33, F35, F36, F40, F43, F45, F46)
DIFF.	14 (for the functions of: F13, F14, F17, F19, F20, F27, F30, F32, F33, F36, F39, F40, F48, F49)	24 (for the functions of: F3, F4, F13, F16, F17, F18, F19, F20, F23, F24, F27, F28, F29, F30, F31, F32, F33, F34, F36, F37, F39, F40, F42, F49)	15 (for the functions of: F2, F18, F19, F20, F21, F24, F27, F28, F29, F31, F33, F36, F40, F42, F48)	29 (for the functions of: F2, F4, F5, F6, F9, F12, F13, F16, F17, F21, F22, F23, F24, F28, F29, F30, F31, F32, F34, F37, F38, F39, F41, F42, F44, F47, F48, F49, F50)

SAME The number of benchmark functions that the CK, PSO, DE and ABC algorithms provide equal values for the *minimum STD* parameter, *DIFF.* The number of benchmark functions that the CK, PSO, DE and ABC algorithms provide different values from the *minimum STD*

Table 7 Comparison of the performances of the CK, PSO, DE and ABC algorithms for the statistical parameters of *BestOpt*

	CK	PSO	DE	ABC
SAME	43 (for the functions of: F1, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F14, F15, F16, F17, F18, F19, F20, F21, F22, F23, F24, F25, F26, F27, F28, F29, F34, F35, F37, F38, F39, F40, F41, F42, F43, F44, F45, F46, F47, F48, F49, F50)	36 (for the functions of: F1, F3, F4, F6, F7, F8, F9, F10, F11, F12, F14, F15, F16, F18, F19, F20, F21, F22, F23, F24, F25, F26, F27, F31, F35, F38, F39, F40, F41, F42, F43, F44, F45, F46, F47, F50)	45 (for the functions of: F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F14, F15, F16, F17, F18, F19, F20, F21, F22, F23, F24, F25, F26, F27, F28, F30, F32, F34, F35, F37, F38, F39, F40, F41, F42, F43, F44, F45, F46, F47, F48, F49, F50)	25 (for the functions of: F1, F7, F8, F9, F10, F11, F13, F14, F15, F18, F19, F22, F25, F26, F27, F28, F33, F35, F36, F40, F41, F42, F43, F45, F46)
DIFF.	7 (for the functions of: F2, F13, F30, F31, F32, F33, F36)	14 (for the functions of: F2, F5, F13, F17, F28, F29, F30, F32, F33, F34, F36, F37, F48, F49)	5 (for the functions of: F13, F29, F31, F33, F36)	25 (for the functions of: F2, F3, F4, F5, F6, F12, F16, F17, F20, F21, F23, F24, F29, F30, F31, F32, F34, F37, F38, F39, F44, F47, F48, F49, F50)

SAME The number of benchmark functions that the CK, PSO, DE and ABC algorithms provide equal values for the *minimum BestOpt* parameter, *DIFF.* The number of benchmark functions that the CK, PSO, DE and ABC algorithms provide different values from the *minimum BestOpt*

Price et al. 2005; Vesterstrom and Thomsen 2004; Bin et al. 2010). Moreover, the process of determining the optimum mutation and crossover strategies for the problem structure in the DE algorithm is time-consuming (Das and Suganthan 2009).

Readers who wish to obtain more detailed information about the DE algorithm are recommended to examine (Ferrante and Ville 2010; Kaelo and Ali 2006; Das and Suganthan 2009).

2.3 Particle swarm optimization algorithm

The particle swarm optimization (PSO) algorithm is a population-based, stochastic and multi-agent parallel global-search technique (Del Valle et al. 2008; Clerc and Kennedy 2002; Trelea 2003; Yoshida et al. 2000; Juang 2004; Sousa et al. 2004). Unlike the genetic algorithm and the DE algorithm, the PSO algorithm has no crossover and mutation operators. The PSO algorithm is based on the mathematical modeling of various collective behaviors of the living creatures that display complex social behaviors. In the PSO algorithm, while a pattern (i.e., particle) is developing a new situation, both the cognitive component of the relative particle and the social component generated by the swarm are used. This situation enables the PSO algorithm to effectively develop the local solutions into global optimum solutions. However, the PSO algorithm is significantly affected by the initial values of the parameters used in the weighting of the cognitive and social components and the weighting strategy of the velocity vector. The Lbest and Gbest topologies are the mostly used topologies in the standard PSO algorithm. In this paper a modernized implementation of the PSO algorithm with Lbest topology, which is known as PSO-2007, has been used (Bin et al. 2010).

The success of the PSO algorithm in finding the global optimum depends extremely on the initial values of the control parameters (c_1, c_2, ω) of the PSO algorithm, the *size of swarm* value, and the maximum iteration number. In the tests made in this paper, control parameter values of the PSO algorithm have been selected as the same with the values given in (Karaboga and Akay 2009a); $c_{1_{initial}} = 1.80$, $c_{2_{initial}} = 1.80$, and $\omega = \frac{1+rand}{2}$ as in (Eberhart and Shi 2001).

For more detailed information on the PSO algorithm, please refer to the study given in (Del Valle et al. 2008; Clerc and Kennedy 2002; Trelea 2003; Ratnaweera et al. 2004; Eberhart and Shi 2001).

2.4 Artificial bee colony algorithm

The Artificial Bee Colony (ABC) algorithm is a population-based numeric optimization algorithm (Karaboga and Akay 2009a; Yang 2005; Zhang et al. 2007; Karaboga and Basturk 2007a,b; Karaboga 2009; Fei et al. 2009; Karaboga and Akay 2009b). It is based on the simplified mathematical models of the food searching behaviors of the bee-swarms. In the ABC algorithm, any random solution of the problem corresponds to a *source of nectar*. There is one *employed bee* assigned to each nectar source. The number of the employed bees equals to the total number of food sources (i.e. the *size of population* value). The employed bee of a nectar source that has run out of nectar turns into a scout bee again. The amount of nectar in a nectar source is expressed with the objective function value of the related nectar source. Therefore, the ABC algorithm targets to locate the nectar source that has the maximum amount of nectar.

In the first step of the ABC algorithm, initial-nectar sources are randomly generated by using the Eq. 14. While starting the calculations, value of the *failure* variable in which the number of failures to develop a nectar source is hidden made as $failure_i = 0$ (Karaboga

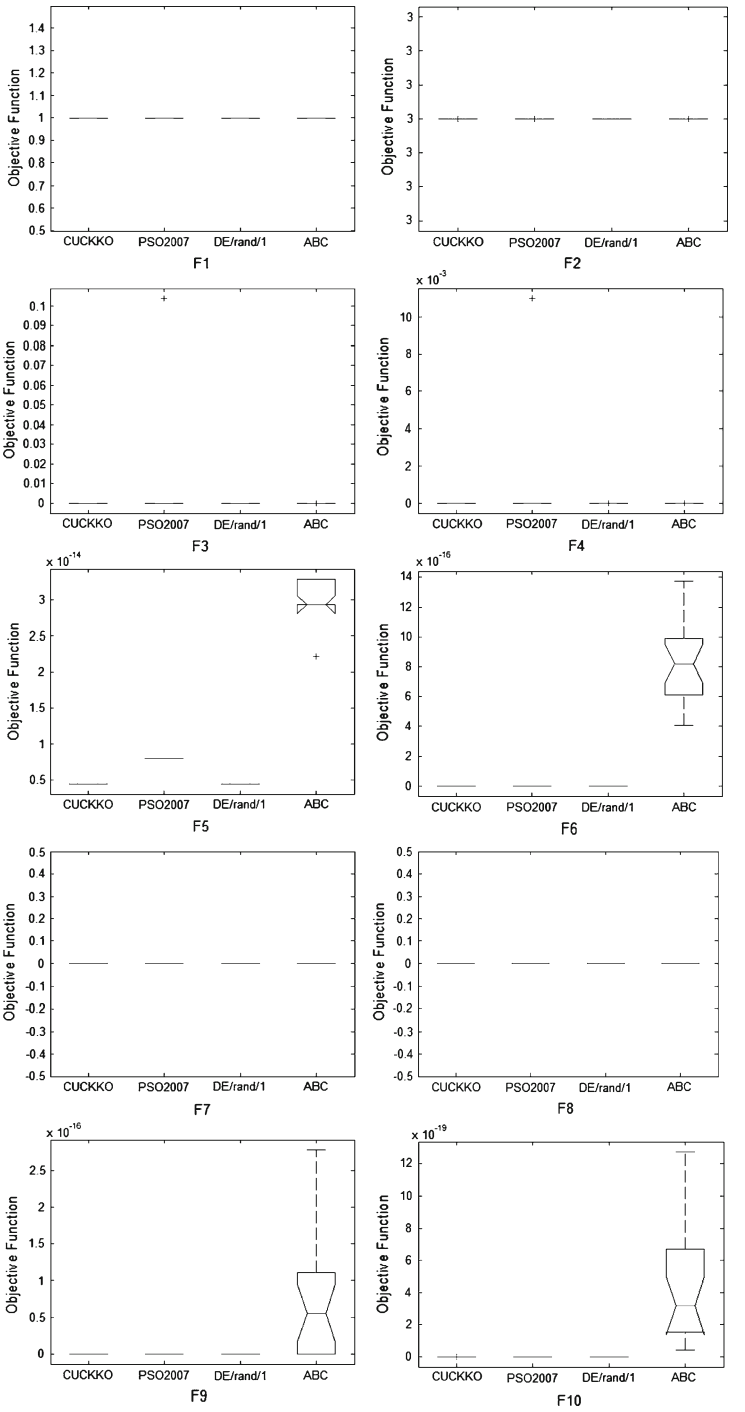

Fig. 1 Anova tests of the global minimum values, which are computed by using the CK, PSO, DE and ABC algorithms (F1–F10)

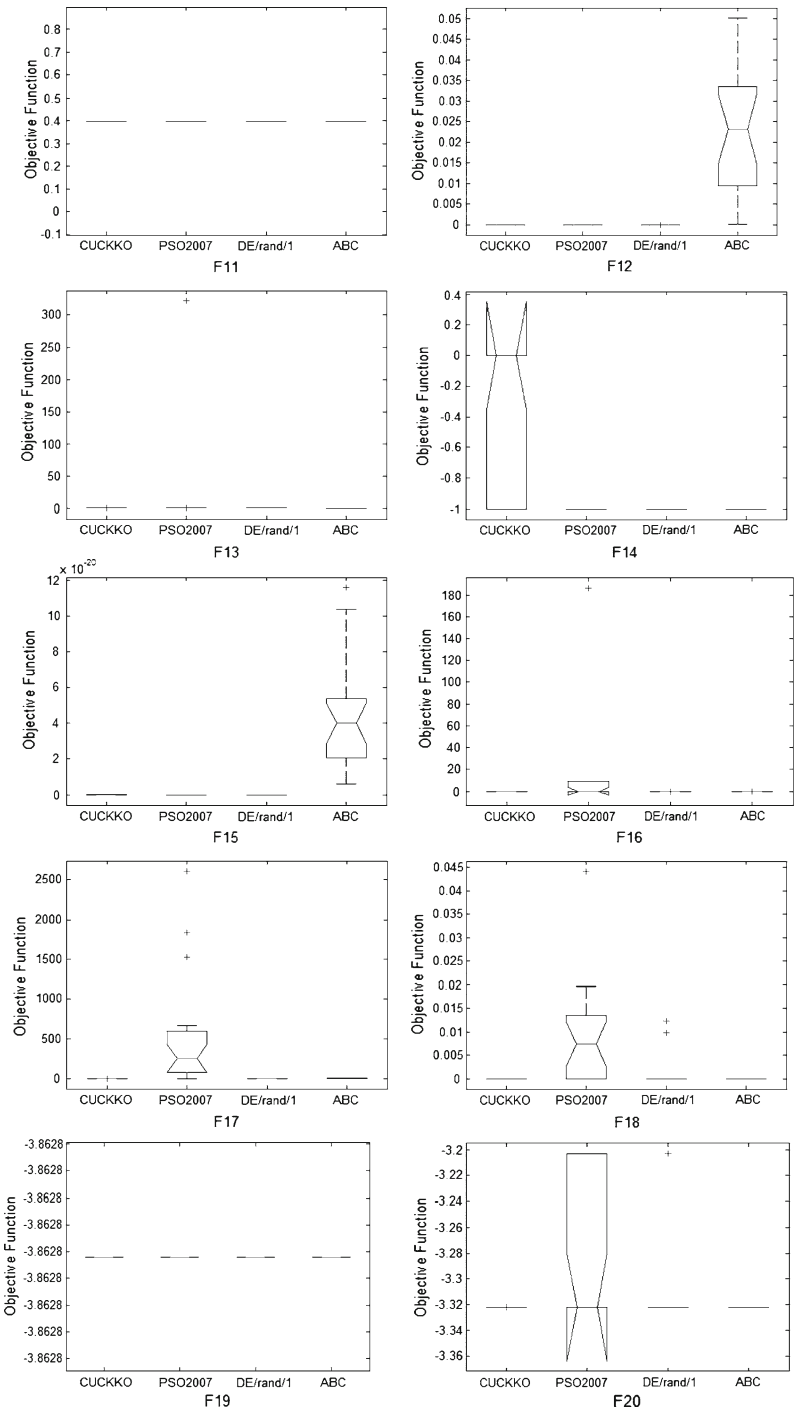

Fig. 2 Anova tests of the global minimum values, which are computed by using the CK, PSO, DE and ABC algorithms (F11–F20)

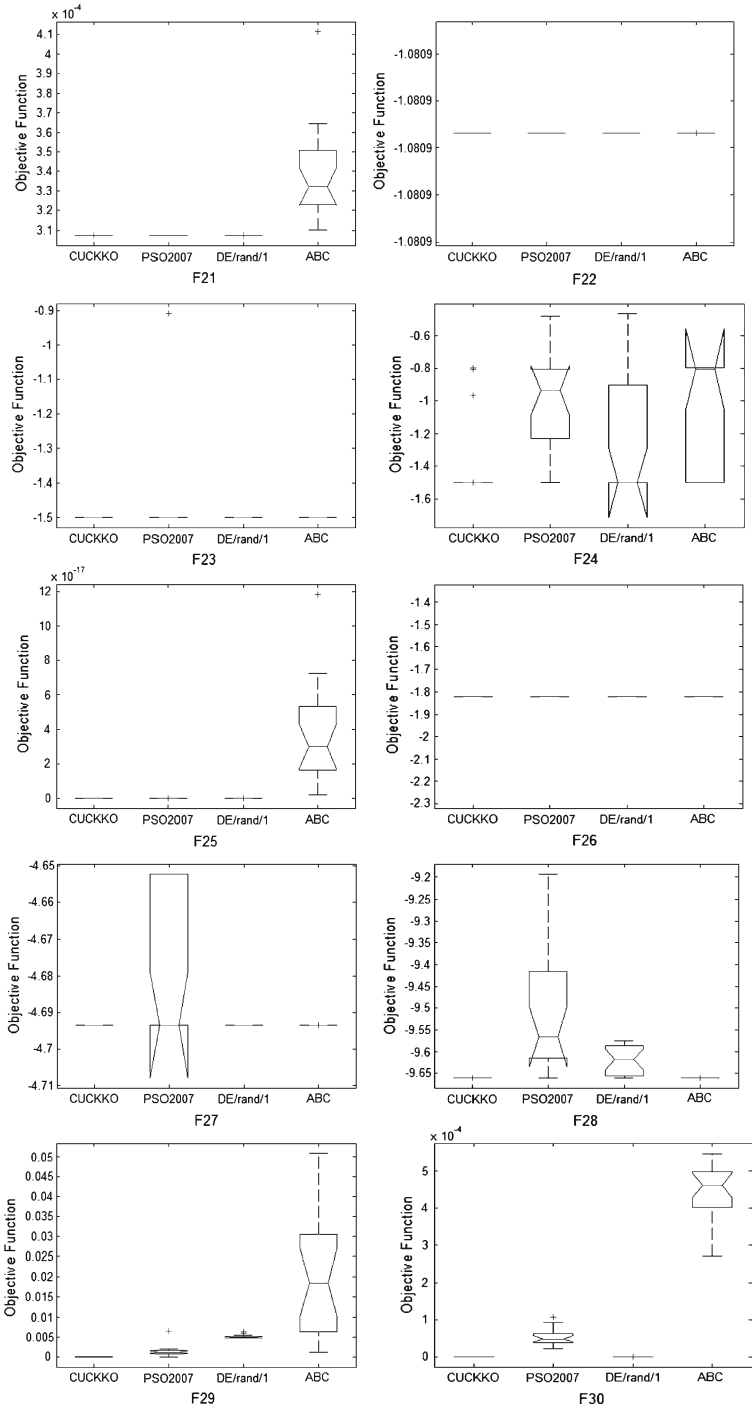

Fig. 3 Anova tests of the global minimum values, which are computed by using the CK, PSO, DE and ABC algorithms (F21–F30)

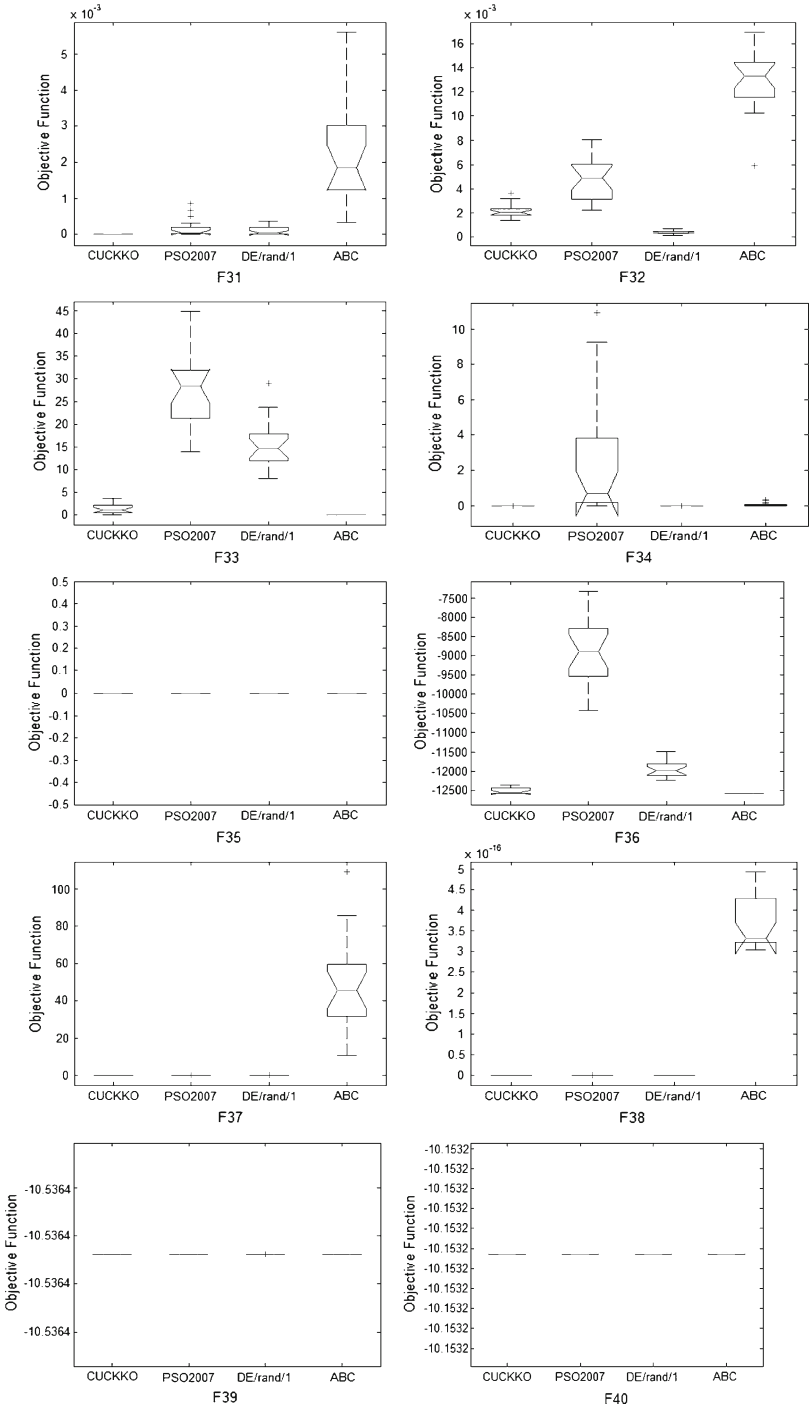

Fig. 4 Anova tests of the global minimum values, which are computed by using the CK, PSO, DE and ABC algorithms (F31–F40)

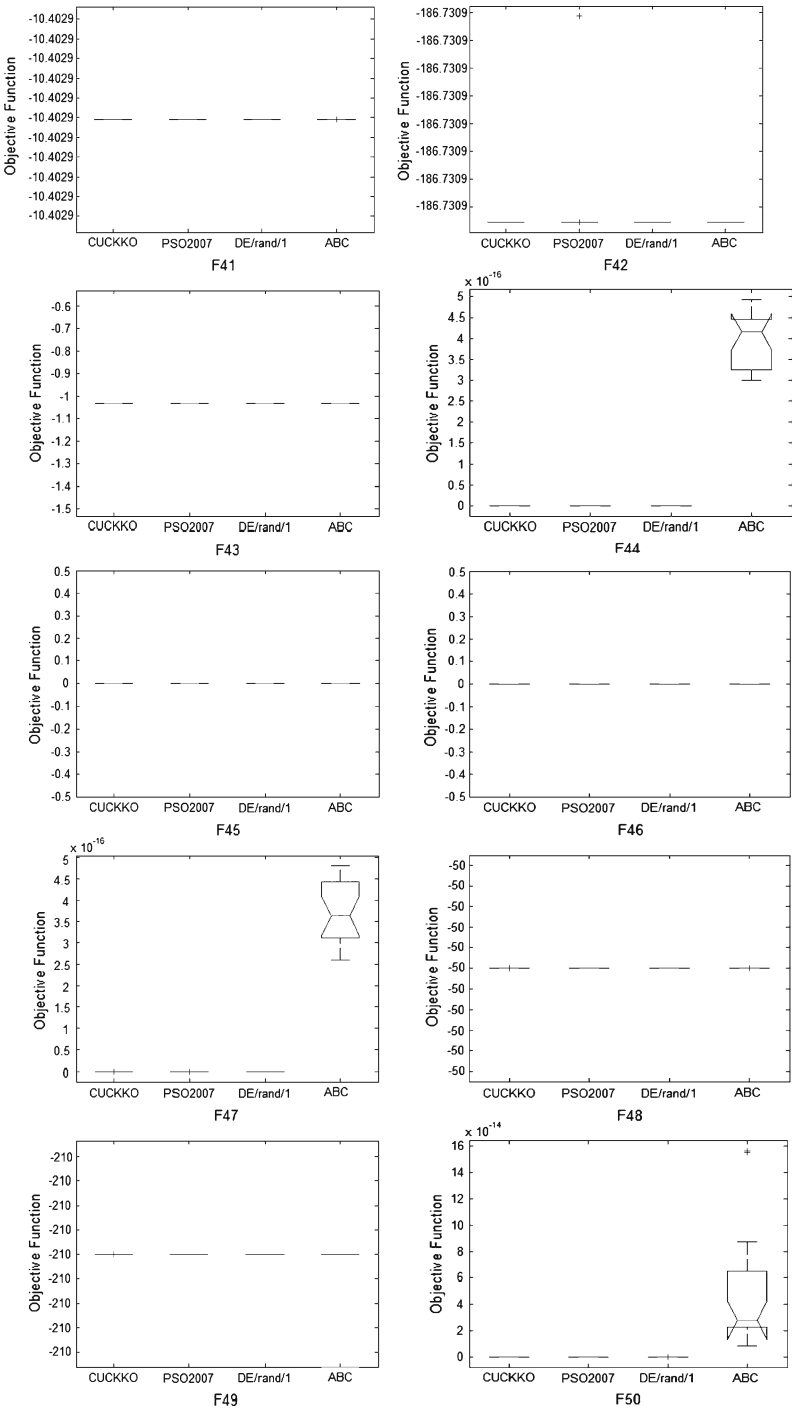

Fig. 5 Anova tests of the global minimum values, which are computed by using the CK, PSO, DE and ABC algorithms (F41–F50)

Table 8 The significantly different algorithms according to Anova test

Fnc.	*P*-value	Algorithms			
		CK	PSO	DE	ABC
F1	–	*	*	*	*
F2	2.3527E-15	DE, ABC	ABC	CK, ABC	CK, PSO, DE
F3	4.3336E-03	PSO	CK, DE, ABC	PSO	PSO
F4	–	*	*	*	*
F5	4.6356E-72	PSO, ABC	CK, PSO, ABC	PSO, ABC	CK, PSO, DE
F6	4.8111E-37	ABC	ABC	ABC	CK, PSO, DE
F7	–	*	*	*	*
F8	–	*	*	*	*
F9	1.1381E-09	ABC	ABC	ABC	CK, PSO, DE
F10	3.4875E-13	ABC	ABC	ABC	CK, PSO, DE
F11	–	*	*	*	*
F12	1.0263E-16	ABC	ABC	ABC	CK, PSO, DE
F13	–	*	*	*	*
F14	1.1401E-16	PSO, DE, ABC	CK	CK	CK
F15	4.2246E-16	ABC	ABC	ABC	CK, PSO, DE
F16	2.5259E-03	ABC	ABC	ABC	CK, PSO, DE
F17	2.0879E-05	ABC	ABC	ABC	CK, PSO, DE
F18	6.7357E-07	ABC	ABC	ABC	CK, PSO, DE
F19	–	*	*	*	*
F20	3.1494E-05	PSO	CK, DE, ABC	PSO	PSO
F21	6.0150E-14	ABC	ABC	ABC	CK, PSO, DE
F22	–	*	*	*	*
F23	–	*	*	*	*
F24	9.1529E-05	PSO, ABC	CK, DE	PSO	CK
F25	2.1747E-13	ABC	ABC	ABC	CK, PSO, DE
F26	–	*	*	*	*
F27	8.4588E-07	PSO	CK, DE, ABC	PSO	PSO
F28	2.3781E-10	PSO	CK, DE, ABC	PSO	PSO
F29	2.8953E-13	ABC	ABC	ABC	CK, PSO, DE
F30	9.9297E-57	PSO, ABC	CK, DE, ABC	PSO, ABC	CK, PSO, DE
F31	1.9907E-15	ABC	ABC	ABC	CK, PSO, DE
F32	1.8096E-39	PSO, DE, ABC	CK, DE, ABC	CK, PSO, ABC	CK, PSO, DE
F33	1.0385E-32	PSO, DE	CK, DE, ABC	CK, PSO, ABC	PSO, DE
F34	1.3360E-05	ABC	ABC	ABC	CK, PSO, DE
F35	–	*	*	*	*
F36	1.5576E-40	PSO, DE	CK, DE, ABC	CK, PSO, ABC	PSO, DE
F37	4.4219E-24	ABC	ABC	ABC	CK, PSO, DE
F38	2.9078E-54	ABC	ABC	ABC	CK, PSO, DE
F39	–	*	*	*	*
F40	–	*	*	*	*

Table 8 continued

Fnc.	*P*-value	Algorithms			
		CK	PSO	DE	ABC
F41	–	*	*	*	*
F42	–	*	*	*	*
F43	–	*	*	*	*
F44	3.1873E-55	ABC	ABC	ABC	CK, PSO, DE
F45	–	*	*	*	*
F46	–	*	*	*	*
F47	1.2363E-51	ABC	ABC	ABC	CK, PSO, DE
F48	–	*	*	*	*
F49	–	*	*	*	*
F50	1.3159E-11	ABC	ABC	ABC	CK, PSO, DE

* No algorithm has significantly different for this benchmark function

and Akay 2009a). Following the generation of initial nectar resources, the ABC algorithm starts to search for the solution of the numeric optimization problem using the *employed bee*, *onlooker bee*, and *scout-bee* tools. The employed bee tries to develop the nectar source to which it is assigned using the other nectar sources as well. If the employed bee finds a better nectar source, it memorizes the new nectar source to use it instead of the old one. This process is modeled in Eq. 21;

$$v_{r_1,j} = X_{r_1,j} + \phi_{r_1,j} \cdot (X_{r_1,j} - X_{r_2,j}) \tag{21}$$

where $\phi_{i,j}$ is a random number generated in the range of $[-1\ 1]$. $X_{r_1,j}$ and $X_{r_2,j}$ indicate the jth parameters of the r_1th and r_2th patterns (i.e., *nectar sources* in the ABC algorithm) respectively.

If the v_{r_1} value has a better objective function value than the X_{r_1} value, the X_{r_1} value is updated as $X_{r_1} := v_{r_1}$ and the *failure* variable becomes $failure_{r_1} = 0$. If the v_{r_1} value does not have a better objective function value than X_{r_1}, the employed bee continues to go to the X_{r_1} source, and since the X_{r_1} solution cannot be developed, the $failure_{r_1}$ value that is the development meter related to the nectar source X_{r_1} increases by one unit.

Using the objective function value of all nectar sources, the probability values, p_i, to be used by the onlooker bees are obtained by using Eq. 22;

$$p_i = \frac{fitness_i}{\sum_{i=1}^{SN} fitness_i} \tag{22}$$

where

$$fitness_i = \begin{cases} \frac{1}{1+f_i} & f_i \geq 0 \\ 1 + |f_i| & f_i < 0 \end{cases} \tag{23}$$

As the $fitness_i$ value given in Eq. 23 increases, the number of employed bees that will select this region of nectar source will increase. The ABC algorithm selects the nectar sources to be visited by the bees using the *roulette selection* technique used in the genetic algorithms; a random number within the range of [0 1] is generated for each nectar source, if the p_i value is higher than the generated random number, the onlooker bees search for new nectar sources

Table 9 The *MeanFncEvol* values of the CK, PSO, DE and ABC algorithms

	CK	PSO	DE	ABC
F1	6,3780.000	16,075.000	6,435.850	5,7078.150
F2	762,760.000	945,452.500	25,821.550	858,664.100
F3	1,189,595.000	284,395.000	84,940.200	1,222,120.000
F4	510,685.000	306,765.000	88,159.050	1,005,375.700
F5	747,320.000	287,335.000	109,007.050	676,613.600
F6	64,115.000	21,550.000	8,223.300	1,209,937.000
F7	38,740.000	8,027.500	4,674.100	3,900.000
F8	43,420.000	8,505.000	4,874.300	10,288.750
F9	40,860.000	11,332.500	5,012.600	841,851.800
F10	46,005.000	16,192.500	7,804.750	1,068,879.250
F11	42,735.000	7,277.500	4,829.100	6,492.000
F12	199,765.000	109,8730.000	22,549.900	1,098,706.450
F13	1,037,135.000	293,197.500	81,170.950	1,143,382.950
F14	17,545.000	7,817.500	5,114.700	68,610.800
F15	100,585.000	14,725.000	8,844.500	774,403.300
F16	496,845.000	408,265.000	24,767.800	1,218,613.050
F17	1,987,760.000	1,274,342.500	65,553.900	829,452.950
F18	450,290.000	212,505.000	68,883.200	43,0470.800
F19	51,450.000	13,525.000	6,947.200	572,035.850
F20	696,830.000	184,805.000	289,642.400	69,249.850
F21	1,362,165.000	1,023,625.000	142,949.200	972,436.700
F22	82,990.000	42,910.000	14,966.100	212,638.600
F23	245,280.000	23,057.500	13,396.350	724,480.350
F24	1,299,345.000	291,832.500	70,887.500	990,701.250
F25	41,055.000	24,962.500	24,957.050	1,079,579.850
F26	33,565.000	8,127.500	4,456.950	73,930.250
F27	623,450.000	133,180.000	344,507.500	766,727.100
F28	1,147,535.000	645,212.500	359,735.950	555,586.400
F29	1,954,575.000	12,51,597.500	1,322,338.800	886,792.150
F30	1,957,870.000	19,99,887.500	271,687.250	1,999,517.500
F31	1,879,165.000	1,395,240.000	1,999,987.950	1,036,051.250
F32	1,713,795.000	1,882,150.000	1,666,733.800	1,080,681.000
F33	1,993,640.000	712,470.000	192,018.000	102,052.350
F34	1,497,570.000	1,999,950.000	735,727.450	965,998.000
F35	260,375.000	24,682.500	10,016.150	51,913.500
F36	1,994,785.000	318,157.500	207,993.500	191,307.150
F37	1,914,020.000	1,999,105.000	226,282.150	1,999,902.500
F38	568,315.000	191,775.000	79,724.600	1,298,752.250
F39	514,440.000	317,287.500	23,287.050	593,977.200
F40	124,835.000	63,632.500	14,276.150	268,448.650
F41	171,940.000	135,647.500	25,890.850	184,214.600
F42	955,790.000	640,845.000	28,081.450	504,695.750

Table 9 continued

	CK	PSO	DE	ABC
F43	40,345.000	10,640.000	5,436.250	6,922.200
F44	408,885.000	205,107.500	85,477.800	989,217.850
F45	88,350.000	56,950.000	15,570.150	11,070.000
F46	2,160.000	282.500	4,399.650	977.500
F47	393,375.000	197,022.500	81,768.550	1,004,375.150
F48	228,320.000	170,175.000	174,747.700	920,208.350
F49	343,490.000	749,015.000	334,781.400	1102,751.300
F50	221,570.000	123,570.000	37,208.000	1,945,944.400

to develop the nectar source X_i using the Eqs. 22, 23. If an X_i source has a $failure_i$ value higher than a certain threshold value, that X_i source is left, and the employed bee assigned hereto goes to a random nectar source generated newly. The success of the ABC algorithm in finding the global optimum is sensitive to the control parameters of the algorithm (i.e, the *limit* value, the number of the *employed-bees*, *the population size*, and *the maximum cycle value*). The ABC algorithm's control parameter of *limit* used in the tests conducted in this paper are the same with the value used in (Karaboga and Akay 2009a); (limit $= SN \cdot D$ as in Eq. 9 of (Karaboga and Akay 2009a).

The local-search ability of the ABC algorithm is sufficiently strong for various problem types. The ABC algorithm selects the pattern used in defining the search-direction using the probability values and the roulette-selection rule used in the genetic-algorithm (see, Eqs. 6, 7 of Karaboga and Akay 2009a). In the ABC algorithm, the probability values are calculated using the fitness values (see, Eq. 6 of Karaboga and Akay 2009a, Eq. 3 of Akay and Karaboga 2010 and Eq. 2.1 of Karaboga and Basturk 2007b). The mathematical model used in the standard ABC algorithm while calculating the fitness values causes it to produce equal probability values for the local solutions that have equal absolute values but different signs (see, Eq. 6 of Karaboga and Akay 2009a). This decreases the probability of a pattern that provides a relatively better solution being selected to define the search-direction.

The modified-ABC algorithm calculates the fitness values for different local-solutions by using different methods (see, Eq. 3 of Akay and Karaboga 2010). Therefore, it is appropriate to sort the patterns in the probability values of pattern matrix acquired in the end just nonlinearly according to the quality of the solution they acquire. Thus, the strategies used to calculate the probability values in the ABC algorithm produce the pseudo-probability value that is suitable to grade the patterns in the pattern matrix just nonlinearly. This affects the problem solving success of the ABC algorithm significantly, and decreases its local-search ability (Karaboga and Akay 2009a; Karaboga and Basturk 2007b; Akay and Karaboga 2010).

For more detailed information on the ABC algorithm, please refer to the study given in (Karaboga and Akay 2009a; Karaboga and Basturk 2007a,b; Karaboga 2009; Fei et al. 2009; Karaboga and Akay 2009b).

3 Experiments

In this paper, various features of the benchmark functions used for testing the successes of the CK, PSO, DE and ABC algorithms are given in Table 1. Mathematical descriptions of

Table 10 The *MinRuntime* values of the CK, PSO, DE and ABC algorithms

	CK	PSO	DE	ABC
F1	17.346	4.051	1.768	11.766
F2	38.111	29.233	0.675	25.802
F3	304.774	65.306	22.152	232.415
F4	161.562	87.265	28.759	221.884
F5	51.291	11.952	4.262	24.964
F6	4.125	0.812	0.285	46.955
F7	2.296	0.273	0.133	0.132
F8	2.544	0.295	0.134	0.323
F9	2.410	0.385	0.140	27.665
F10	2.688	0.529	0.213	34.433
F11	2.267	0.223	0.115	0.191
F12	12.230	40.636	0.722	37.565
F13	91.050	17.662	4.931	60.084
F14	0.957	0.222	0.123	2.078
F15	11.159	1.316	0.786	55.841
F16	55.956	35.935	2.263	96.074
F17	229.316	119.123	6.191	65.035
F18	41.236	13.469	4.355	24.613
F19	2.838	0.431	0.186	17.409
F20	37.047	6.028	7.941	2.173
F21	69.023	30.559	3.693	26.273
F22	10.718	4.501	1.619	18.607
F23	31.793	2.466	1.466	59.122
F24	172.384	32.778	8.005	91.876
F25	2.294	0.830	0.660	34.101
F26	2.312	0.378	0.179	3.406
F27	49.757	8.007	20.300	44.706
F28	121.003	55.008	31.341	38.011
F29	355.641	202.511	232.793	101.448
F30	228.237	189.205	27.143	152.155
F31	226.846	140.658	215.596	87.648
F32	149.040	121.339	106.970	60.014
F33	118.470	26.071	6.830	3.400
F34	96.516	76.901	28.318	31.648
F35	13.974	0.802	0.265	1.593
F36	121.250	11.845	7.441	6.699
F37	112.177	66.368	7.323	60.890
F38	35.193	6.815	2.741	43.919
F39	27.938	10.269	0.667	18.884
F40	6.706	1.944	0.383	7.541
F41	9.245	4.337	0.709	5.978
F42	52.095	21.321	0.756	15.322

Table 10 continued

	CK	PSO	DE	ABC
F43	2.149	0.313	0.143	0.205
F44	24.075	6.704	2.692	27.930
F45	5.438	1.933	0.512	0.355
F46	0.123	0.005	0.112	0.027
F47	29.290	9.603	3.896	42.597
F48	13.428	6.269	5.473	32.045
F49	20.678	28.576	11.263	37.974
F50	14.276	5.547	1.508	77.222

the benchmark functions can be found in (Karaboga and Akay 2009a; Karaboga and Basturk 2007a,b). The benchmark functions used in the tests consist of Unimodal (U), Multimodal (M), Separable (S), and Non-Separable (N) functions of different features. The problem dimensions of the benchmark functions used vary between 2 and 30, as in (Karaboga and Akay 2009a; Ferrante and Ville 2010; Zhang and Sanderson 2009; Vesterstrom and Thomsen 2004).

In all experiments in this paper, the values of the common control parameters of the mentioned algorithms such as the *size of pattern matrix* and the *maximum function evaluation number* were chosen to be the same. The size of the *pattern matrix* has been fixed as 50 and the maximum *function evaluation number* was set to 2,000,000. In order to make comparison coherently, the global minimum values below 10^{-16} are assumed to be 0 in all experiments.

The setting values of algorithmic control parameters of the mentioned algorithms are given below:

- *CK Settings*: $\beta = 1.50$ and $p_0 = 0.25$ have been used as recommended in (Yang and Deb 2010).
- *DE Settings*: In DE, the $DE/rand/1$ mutation strategy with $binomial$ crossover operator has been used. The algorithmic control parameters of $DE/rand/1$ has been used as $F = 0.50$ and $Cr = 0.90$ as recommended in (Ferrante and Ville 2010; Das and Suganthan 2009; Karaboga and Akay 2009a).
- *PSO Settings*: $C_1 = C_2 = 1.80$ and $\omega = 0.60$ have been used as recommended in (Karaboga and Akay 2009a).
- *ABC Settings*: $limit = 50D$ has been used as recommended in (Karaboga and Akay 2009a).

The global minimum values of each of the benchmark functions used in this paper have been solved 20 times by the mentioned algorithms using a different initial population at every turn. The *run-time* and the minimum *function evaluation number* of the best solution, and the final *global minimum value* have been recorded during experiments for further statistical analysis. Subsequently, the mean-value of the global minimum values (*MeanOpt*), standard deviation value of *MeanOpt* (*STD*) and best solution (*BestOpt*) values have been computed by analyzing the recorded *global minimum values* during experiments. The *MeanOpt*, *STD* and *BestOpt* values are given in Tables 2, 3, and 4. The multiple comparison results for the *minimum-MeanOpt* values are given in Table 5, where the *minimum-MeanOpt* value denotes the *minimum* value of the *MeanOpt* values computed by the CK, PSO, DE and ABC algorithms for a certain benchmark function. As it is seen from Table 5, the performances of the

Table 11 Comparison of the performances of the CK, PSO, DE and ABC algorithms for the statistical parameters of *MeanFncEvol*

	CK	PSO	DE	ABC
SAME	0	3 (for the functions of: F27, F46, F48)	39 (for the functions of: F1, F2, F3, F4, F5, F6, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18, F19, F21, F22, F23, F24, F25, F26, F28, F30, F34, F35, F37, F38, F39, F40, F41, F42, F43, F44, F47, F49, F50)	8 (for the functions of: F7, F20, F29, F31, F32, F33, F36, F45)
DIFF.	50 (for the functions of: F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18, F19, F20, F21, F22, F23, F24, F25, F26, F27, F28, F29, F30, F31, F32, F33, F34, F35, F36, F37, F38, F39, F40, F41, F42, F43, F44, F45, F46, F47, F48, F49, F50)	47 (for the functions of: F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18, F19, F20, F21, F22, F23, F24, F25, F26, F28, F29, F30, F31, F32, F33, F34, F35, F36, F37, F38, F39, F40, F41, F42, F43, F44, F45, F47, F49, F50)	11 (for the functions of: F7, F20, F27, F29, F31, F32, F33, F36, F45, F46, F48)	42 (for the functions of: F1, F2, F3, F4, F5, F6, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18, F19, F21, F22, F23, F24, F25, F26, F27, F28, F30, F34, F35, F37, F38, F39, F40, F41, F42, F43, F44, F46, F47, F48, F49, F50)

SAME The number of benchmark functions that the CK, PSO, DE and ABC algorithms provide equal values for the *MeanFncEvol* parameter, *DIFF.* The number of benchmark functions that the CK, PSO, DE and ABC algorithms provide different values from the *MeanFncEvol*

CK and DE algorithms are better than the performances of PSO and ABC. The CK and DE algorithms find out the *minimum MeanOpt* values in 41 and 38 functions, respectively. The multiple comparison results for the *minimum-STD* values are given in Table 6, where the *minimum-STD* value denotes the *minimum* value of the *STD* values computed by the CK, PSO, DE and ABC algorithms for a certain benchmark function. The multiple comparison results for the *BestOpt* values are given in Table 7, where the *BestOpt* value denotes the *best* solution of the global minimizer values computed by the CK, PSO, DE and ABC algorithms for a certain benchmark function. As it is seen from Table 7, the performances of the CK, DE and PSO algorithms are better than the performance of the ABC aspect of the *BestOpt* values.

In addition to the basic statistical analyzes given above (i.e., *MeanOpt*, *STD* and *BestOpt*), analysis of variance (Anova) test has also been carried out for multiple comparison of the performances of the CK, PSO, DE and ABC algorithms. The null hypothesis is determined as '*there is no difference in the minimum-MeanOpt performances of the CK, PSO, DE and ABC algorithms*' and $\alpha = 0.05$ (for a 95% confidence) has been used at Anova test. The graphical analysis results of the Anova test are illustrated in Figs. 1, 2, 3, 4, and 5 and *significantly* different algorithms for the benchmark functions are tabulated in Table 8. *Mean-function evaluation numbers* (*MeanFncEvol*) have also been analyzed, where *MeanFncEvol* denotes the mean of the best function evaluation numbers of an experiment (an experiment involves 20 trials as mentioned above). The *MeanFncEvol* values have been tabulated in Table 9.

Table 12 Comparison of the performances of the CK, PSO, DE and ABC algorithms for the statistical parameters of *MinRuntime*

	CK	PSO	DE	ABC
SAME	0	2 (for the functions of: F27, F46)	40 (for the functions of: F1, F2, F3, F4, F5, F6, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18, F19, F21, F22, F23, F24, F25, F26, F28, F30, F34, F35, F37, F38, F39, F40, F41, F42, F43, F44, F47, F48, F49, F50)	8 (for the functions of: F7, F20, F29, F31, F32, F33, F36, F45)
DIFF.	50 (for the functions of: F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18, F19, F20, F21, F22, F23, F24, F25, F26, F27, F28, F29, F30, F31, F32, F33, F34, F35, F36, F37, F38, F39, F40, F41, F42, F43, F44, F45, F46, F47, F48, F49, F50)	48 (for the functions of: F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18, F19, F20, F21, F22, F23, F24, F25, F26, F28, F29, F30, F31, F32, F33, F34, F35, F36, F37, F38, F39, F40, F41, F42, F43, F44, F45, F47, F48, F49, F50)	10 (for the functions of: F7, F20, F27, F29, F31, F32, F33, F36, F45, F46)	42 (for the functions of: F1, F2, F3, F4, F5, F6, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18, F19, F21, F22, F23, F24, F25, F26, F27, F28, F30, F34, F35, F37, F38, F39, F40, F41, F42, F43, F44, F46, F47, F48, F49, F50)

SAME The number of benchmark functions that the CK, PSO, DE and ABC algorithms provide equal values for the *MinRuntime* parameter, *DIFF.* The number of benchmark functions that the CK, PSO, DE and ABC algorithms provide different values from the *MinRuntime*

The *run-time* complexities are given as seconds in Table 10 where *MinRuntime* denotes the run-time of the best function evaluation number of an experiment. The multiple comparison results for the *MeanFncEvol* and *MinRuntime* have been given in Tables 11, 12.

4 Conclusion

In this paper, the numerical optimization problem solving successes of the CK, PSO, DE and ABC algorithms have been compared statistically. Statistical analysis relieved that the problem solving success of the CK and DE algorithms are quite better than the PSO (i.e., PSO2007) and ABC algorithms. Although there are several improved versions of PSO in the literature, the PSO-2007 implementation has been preferred in the tests due to its high performance.

The PSO algorithm is successful in the solution of many benchmark functions, but its well-known stability problem restricts the success rate of this algorithm against the CK and DE algorithms. Since the PSO algorithm maintains its stochastic behavior capacity better than the ABC algorithm while searching for the global optimum value, it provides more successful results than the ABC algorithm. The ABC algorithm basically has a search strategy, which is considerably similar to the standard DE algorithm (i.e., DE/rand/1). However, the ABC algorithm has a very successful decision mechanism that decides which areas within

the search space require to be surveyed in more detail. The strategy of the ABC algorithm used to discover new nectar sources within the ABC algorithm and manage the capacity of the nectar sources discovered is also substantially powerful.

Acknowledgments The authors would like to thank to the referees who have contributed to enhancement of the technical contents of this paper. The studies in this paper have been supported within the scope of the scientific research projects of 110Y309 supported by TUBITAK and FBA-9-1131 supported by Erciyes University.

References

Akay B, Karaboga D (2010) A modified artificial bee colony algorithm for real-parameter optimization. Inf Sci (in press, online version)

Ali MM, Torn A (2004) Population set-based global optimization algorithms: some modifications and numerical studies. Comput Oper Res 31(10):1703–1725

Bin X, Jie C, Zhi-Hong P, Feng P (2010) An adaptive hybrid optimizer based on particle swarm and differential evolution for global optimization. Sci China Inf Sci 53(5):980–989

Chaoshun L, Jianzhong Z (2011) Parameters identification of hydraulic turbine governing system using improved gravitational search algorithm. Energy Convers Manag 52(1):374–381

Clerc M, Kennedy J (2002) The particle swarm—explosion, stability, and convergence in a multidimensional complex space. IEEE Trans Evol Comput 6(1):58–73

Corne D, Dorigo M, Glover F (1999) New ideas in optimization. McGraw-Hill, USA

Das S, Suganthan P (2009) Differential evolution: a survey of the state-of-the-art. IEEE Trans Evol Comput 15(1):4–31

Das S, Mukhopadhyay A, Roy A, Abraham A, Panigrahi BK (2011) Exploratory power of the Harmony search algorithm: analysis and improvements for global numerical optimization. IEEE Trans Syst Man Cybern Part B Cybern 4(1):89–106

Deb K, Pratap A, Agarwal S et al (2002) A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Trans Evol Comput 6(2):182–197

Del Valle Y, Venayagamoorthy GK, Mohagheghi S, Hernandez JC, Harley RG (2008) Particle swarm optimization: basic concepts, variants and applications in power systems. IEEE Trans Evol Comput 12(2):171–195

Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern Part B Cybern 26(1):29–41

Dorigo M, Bonabeau E, Theraulaz G (2000) Ant algorithms and stigmergy. Future Gener Comput Syst 16(8):851–871

Dorigo M, Trianni V, Sahin E et al (2004) Evolving self-organizing behaviors for a swarm-bot. Auton Robots 17(2–3):223–245

Duman S, Guvenc U, Yorukeren N (2010) Gravitational search algorithm for economic dispatch with valve-point effects. Int Rev Electr Eng Iree 5(6):2890–2895

Eberhart RC, Shi Y (2001) Tracking and optimizing dynamic systems with particle swarms. In: Proceedings of IEEE congress on evolutionary computation vol 1, pp 94–100

Esmat R, Hossein NP, Saeid S (2010) Bgsa: binary gravitational search algorithm. Nat Comput 9(3):727–745

Esmat R, Hossien NP, Saeid S (2011) Filter modeling using gravitational search algorithm. Eng Appl Artif Intell 24(1):117–122

Fei K, Junjie L, Qing X (2009) Structural inverse analysis by hybrid simplex artificial bee colony algorithms. Comput Struct 87(13–14):861–870

Ferrante N, Ville T (2010) Recent advances in differential evolution: a survey and experimental analysis. Artif Intell Rev 33(1–2):61–106

Geem ZW, Kim JH, Loganathan G (2001) A new heuristic optimization algorithm: Harmony search. Simulation 76(2):60–68

Haupt R (1995) Comparison between genetic and gradient-based optimization algorithms for solving electromagnetics problems. IEEE Trans Magn 31(3):1932–1935

Horst R, Pardalos PM, Thoai NV (2000) Introduction to global optimization. Kluwer Academic Publishers, Dordrecht, The Netherland

Janez B, Borko B, Saso G et al (2007) Performance comparison of self-adaptive and adaptive differential evolution algorithms. Soft Comput 11(7):617–629

Juang C (2004) A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. IEEE Trans Syst Man Cybern Part B Cybern 34(2):997–1006
Kaelo P, Ali MM (2006) A numerical study of some modified differential evolution algorithms. Eur J Oper Res 169(3):1176–1184
Karaboga D, Akay B (2009a) A comparative study of artificial bee colony algorithm. Appl Math Comput 214(12):108–132
Karaboga D, Akay B (2009b) A survey: algorithms simulating bee swarm intelligence. Artif Intell Rev 31(1–4):61–85
Karaboga D, Basturk B (2007a) Artificial bee colony (abc) optimization algorithm for solving constrained optimization problems. Lecture Notes Comput Sci 4529:789–798
Karaboga D, Basturk B (2007b) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. J Glob Optim 39(3):459–471
Karaboga N (2009) A new design method based on artificial bee colony algorithm for digital iir filters. J Frankl Inst Eng Appl Math 346(4):328–348
Lee K, Geem ZW (2004) A new structural optimization method based on the Harmony search algorithm. Comput Struct 82(9–10):781–798
Lee K, Geem ZW (2005) A new meta-heuristic algorithm for continuous engineering optimization: Harmony search theory and practice. Comput Methods Appl Mech Eng 194(36–38):3902–3933
Liu J, Lampinen J (2005) A fuzzy adaptive differential evolution algorithm. Soft Comput 9(6):448–462
Mahamed GO, Mehrdad M (2008) Global-best Harmony search. Appl Math Comput 198(2):643–656
Mahdavi M, Fesanghary M, Damangir E (2007) An improved Harmony search algorithm for solving optimization problems. Appl Math Comput 188(2):1567–1579
Martinoli A, Easton K, Agassounon W (2004) Modeling swarm robotic systems: a case study in collaborative distributed manipulation. Int J Robot Res 23(4-5):415–436
Mersha AG, Dempe S (2011) Direct search algorithm for bilevel programming problems. Comput Optim Appl 49(1):1–15
Nowak W, Cirpka OA (2004) A modified levenberg-marquardt algorithm for quasi-linear geostatistical inversing. Adv Water Resour 27(7):737–750
Ong YS, Lim MH, Zhu N et al (2006) Classification of adaptive memetic algorithms: a comparative study. IEEE Trans Syst Man Cybern Part B Cybern 36(1):141–152
Price K, Storn R (1997) Differential evolution. Dr Dobbs J 22(4):18–24
Price K, Storn R, Lampinen J (2005) Differential evolution: a practical approach to global optimization. Springer, Berlin, Germany
Rashedi E, Nezamabadi-pour H, Saryazdi S (2009) Gsa: a gravitational search algorithm. Inf Sci 179(13):2232–2248
Ratnaweera A, Halgamuge SK, Watson HC (2004) Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Trans Evol Comput 8(3):240–255
Shahryar R, Hamid RT, Magdy MAS (2008) Opposition-based differential evolution. IEEE Trans Evol Comput 12(1):64–79
Sousa T, Silva A, Neves A (2004) Particle swarm based data mining algorithms for classification tasks. Comput Optim Appl 30(5–6):767–783
Storn R (1999) System design by constraint adaptation and differential evolution. IEEE Trans Evol Comput 3(1):22–34
Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359
Swagatam D, Ajith A, Uday KC et al (2009) Differential evolution using a neighborhood-based mutation operator. IEEE Trans Evol Comput 13(3):526–553
Tahk MJ, Park MS, Woo HW, Kim HJ (2009) Hessian approximation algorithms for hybrid optimization methods. Eng Optim 41(7):609–633
Trelea IC (2003) The particle swarm optimization algorithm: convergence analysis and parameter selection. Inf Process Lett 85(6):317–325
Vesterstrom J, Thomsen R (2004) A comparative study of differential evolution particle swarm optimization and evolutionary algorithms on numerical benchmark problems. Congr Evol Comput, CEC2004 2:1980–1987
Yang X (2005) Engineering optimizations via nature-inspired virtual bee algorithms. Lecture Notes Comput Sci 3562:317–323
Yang X, Deb S (2009) Cuckoo search via levey flights. World congress on nature and biologically inspired computing'NABIC-2009, vol 4. Coimbatore, pp 210–214
Yang XS (2009) Firefly algorithms for multimodal optimization. Lecture Notes Comput Sci 5792:169–178

Yang XS, Deb S (2010) Engineering optimisation by Cuckoo search. Int J Math Modell Numer Optim 1(4):330–343

Yoshida H, Kawata K, Fukuyama Y et al (2000) A particle swarm optimization for reactive power and voltage control considering voltage security assessment. IEEE Trans Power Syst 15(4):1232–1239

Zhang J, Sanderson A (2009) Tracking and optimizing dynamic systems with particle swarms. IEEE Trans Evol Comput 13(5):945–958

Zhang J, Chung H, Lo W (2007) Engineering optimizations via nature-inspired virtual bee algorithms. IEEE Trans Evol Comput 11(3):326–335

Zhua G, Kwongb S (2010) Gbest-guided artificial bee colony algorithm for numerical function optimization. Appl Math Comput 217(7):3166–3173