Supplementary Material for:

Swagatam Das, Subhodip Biswas, B. K. Panigrahi, Souvik Kundu, and Debabrota Basu, "A Spatially Informative Optic Flow Model of Bee Colony with Saccadic Flight Strategy for Global Optimization", IEEE Transactions on Cybernetics, Accepted, 2014.

Different sections, Tables, and Figures of this document are numbered as A1, A2, ….etc. In what follows, references of the main paper will be called by their actual number as in the main paper and references added in the suppl. material only will be called as R1, R2,…etc.

A1. Related works

For over past one decade or so, researchers have been developing several algorithms based on various intelligent behaviors of the honey bee swarms. Among these, ABC has been vastly accepted by the computational intelligence community and today it is the one of the most widely studied approaches for real parameter optimization.

The first studies on ABC were aimed to evaluate the performance of ABC on the widely used set of numerical benchmark test functions and to predict its niche in the well-known evolutionary computing community which includes well-established members like Particle Swarm Optimization (PSO) [R1], Genetic Algorithms (GAs) [R2], Differential Evolution (DE) [R3], and Ant Colony Optimization (ACO) [R4]. The performance of ABC algorithm was compared with DE, PSO and other EAs for multi-dimensional numeric problems by Karaboga and Basturk [R5]. Mala *et al.* [R6] concluded that ABC based approach has several advantages over ACO by applying it for software test-suite optimization. Ruiz-Vanoye and Daz-Parra [R7] articulated the functional similarities between the science of life and meta-heuristic algorithms including ABC. A performance assessment study of foraging, including ABC and EAs was carried out by El-Abd [R8].

To ameliorate the performance, several EA based concepts have been amalgamated with ABC. Liu and Cai [R9] proposed a new variant, named the artificial bee colony programming, employing randomized distribution, bit hyper-mutation and a novel crossover operator to amend the performance of the traditional algorithm. Motivated by PSO, Zhu and Kwong [R10] inferred that the classical ABC framework facilitates global exploration more in comparison to the exploitation and implemented an ABC algorithm called *gbest*-guided ABC by incorporating the information of global best solution into the solution search equation. Gao and Liu [R11] proposed an improved perturbation scheme for ABC in which the bees search only around the best solution of the former iteration. To enhance the bees update strategy for improving the quality of solutions, Raziuddin *et al.* [R12] introduced the differential ABC. Tuba *et al.* [R13] proposed a novel algorithm called GABC which integrates ABC algorithm with self-adaptive guidance adjusted for engineering optimization problems. Li *et al.* [R14] introduced inertia weight and acceleration coefficients to improve the search procedure of ABC. Banharnsakun *et al.* [R15] brought in the best-so-far selection in ABC and evaluated its performance on two sets of problems: numerical benchmark functions and image registration applications. Kang *et al*. [R16] mentioned the limitations of ABC in handling functions that have a narrow curving valley, a high eccentric ellipse, or are rugged and multimodal in nature. Tsai *et al*. [R17] pointed out that the artificial bee can only move straight to one of the nectar sources by the employed bees. This characteristic can narrow down the search zones for bees to explore.

Akay and Karaboga [R18] recently proposed some modifications in the existing ABC framework for real-parameter optimization. They focused on the importance of controlling frequency of perturbation by a new parameter modification rate MR . The underlying idea behind the modified

ABC (*m*-ABC) was to facilitate better search in complex higher dimensional space as opposed to classical ABC that perturbs a single parameter. The following technique was adopted in m-ABC

$$v_{ij}^{C} = \begin{cases} \theta_{ij}^{C} + \phi_{ij}\left(\theta_{kj}^{C} - \theta_{ij}^{C}\right) & if\ \ rand(0,1) \le MR \\ \theta_{ij}^{C} & otherwise \end{cases}. \quad \text{(S1)}$$

The impact of the control parameters introduced was studied individually and a detailed analysis was made. The modified ABC was able to improve upon the classical ABC for highly multimodal, composite, and rotated problems.

Over past few years, ABC has attracted a good deal of research interest, resulting into a plethora of its improved variants, hybridized versions, and reports on several successful applications. Hybridized ABC algorithms have been proposed to make the algorithm more powerful in which traditional ABC algorithm is combined with different strategies of the established EAs. A hybrid simplex ABC algorithm synergizing the Nelder-Mead simplex method and ABC was proposed by Kang *et al.* [R19]. Duan *et al.* [R20] hybridized ABC with quantum evolutionary algorithm for solving continuous optimization problems. Shi *et al.* [R21] proposed a hybrid swarm intelligent algorithm based on PSO and ABC. Based on the idea of the parallel computation merit of GA, and the speed and self-improvement abilities of ABC, Zhao *et al.* [R22] described a novel hybrid swarm intelligent approach to single-objective function optimization. El-Abd [R23] combined the PSO algorithm with ABC component to improve the personal bests of the particles. To solve course scheduling problem, Oner *et al.* [R24] described a hybrid algorithm composed of a heuristic graph node colouring algorithm and ABC. To generate a novel approximate model for predicting network reliability, Yeh *et al.* [R25] presented a study which combines a bee recurrent neural network optimized by the ABC algorithm with Monte Carlo simulation. Recently Gao *et al*. [R26] proposed an improved variant of ABC by including a modified perturbation strategy aided with the orthogonal learning scheme.

Researchers have been motivated to extend the use of the ABC algorithm to other areas due to the outstanding performance of the algorithm as a single-objective optimizer specifically when dealing with continuous search spaces. For example, Akay and Karaboga [R27] employed ABC to integer programming problems. A combinatorial ABC for travelling salesman problems was introduced by Karaboga and Gorkemli [R28]. Ho and Yang [R29] proposed some novel ways for the employed bees and onlookers to carry out exploiting searches for inverse electromagnetic problem. Mohan and Baskaran [R30] have reconstituted ABC from the initialization phase to the implementation phase and described an energy aware and energy efficient routing protocol for adhoc network by using restructured ABC. Karaboga and Basturk extended ABC algorithm for solving non-linearly constrained problems in [R31]. Very recently, ABC has also been extended to multimodal and dynamic landscapes [R32-R35]. Diversification operators as in [R36-R37] have also been applied.

Recently, the ABC algorithm has been utilized to solve many engineering problems encountered in research and industries. These includes likes of multi-objective design optimization of laminated composite components [R38], solving non-convex economic dispatch problem [R39], parametric optimization of non-traditional machining processes [R40], capacitated vehicle routing problem [R41], forecasting stock markets [R42], robust optimal design and manufacturing [R43], and optimization of the parameters of a least squares support vector machine with applications to data classification problems [R44], circular antenna array synthesis. [R45].

A2. Experimental Results

A.2.1 Results on 30D problems

A close inspection of Table A1 reveals that *Si*PABC_*Sf* obtains the least possible error values on 23 out of 25 possible instances while managing to outperform other competing algorithms on 21 out of 25 cases. In spite of detecting optimum in both functions f_2 and f_9, JADE achieves better accuracy than our method. On f_{21} *Si*PABC_*Sf* gives results at par with CLPSO and HPA, while on function f_{24} similarly mean error values are obtained by SaDE, jDE and CLPSO. In f_7, f_8 and f_{25} HPA comes near to the result obtained by *Si*PABC_*Sf*. On comparing with the different versions of ABC- classical ABC, modified ABC and hybridized HPA, we can infer that the novel *Si*PABC_*Sf* framework manages to improve upon the performances of ABC and m-ABC in every test instance. The recent variant HPA owing to its good local search and multi-parametric perturbation is the best among the ABC variants used. An important observation related to the mean performance can be inferred from the fact that the mean error value obtained by our algorithm is lower by several orders of magnitude (8 for f_1, 11 for f_6 and 6 for f_{13}) from its nearest competitor. The *p*-values obtained by Wilcoxon's rank sum test confirm our observation. Interestingly enough *Si*PABC_*Sf* has successfully located the optima of 6 test problems (f_1, f_2, f_4, f_6, f_7 and f_9), followed by 3 such instances for SaDE (f_1, f_2 and f_4) and JADE (f_1, f_2 and f_9) with accuracy threshold of 10^{-5}.

TABLE A1. MEAN, STANDARD DEVIATION AND *P*-VALUES OF WILCOXON'S RANK SUM TEST FOR BEST OF RUN ERROR VALUES FOR 30*D* FUNCTIONS

Algo / Func	SaDE	JADE	jDE	HPA	ABC	CLPSO	GSO	m-ABC	SiPABC_*Sf*
	Mean (Std) [*p*-value]	Mean (Std) [*p*-value]	Mean (Std) [*p*-value]	Mean (Std) [p-value]	Mean (Std) [*p*-value]	Mean (Std) [*p*-value]	Mean (Std) [*p*-value]	Mean (Std) [*p*-value]	Mean (Std) [*p*-value]
f_1	6.7843e-45 (2.3879e-50) [2.91e-06]	1.3258e-44 (9.2436e-60) [3.02e-08]	3.8652e-39 (4.5732e-44) [8.27e-10]	3.3023e-28 (4.2689e-28) [5.78e-15]	1.0075e-29 (1.9454e-29) [1.81e-17]	6.8212e-18 (2.5421e-18) [4.28e-20]	6.3267e-08 (7.7487e-09) [2.12e-20]	3.4106e-37 (1.3189e-42) [4.23e-10]	**1.3428e-51 (3.1258e-52) NA**
f_2	9.7191e-26 (4.8596e-26) [2.81e-10]	**3.5246e-32 (7.4269e-50) NA**	7.5064e-26 (7.3804e-28) [4.03e-09]	1.1705e+02 (1.3582e+02) [3.87e-20]	8.5451e-09 (1.7258e-10) [2.14e-08]	2.9339e-11 (2.7952e-11) [3.26e-09]	1.2357e+04 (5.3441e+03) [6.12e-21]	1.2117e-26 (1.7141e-05) [2.62e-15]	8.1159e-32 (1.2568e-37) [3.37e-02]
f_3	2.0521e+06 (1.5754e+05) [3.27e-20]	1.0421e+04 (5.6273e+03) [1.27e-08]	2.2663e+05 (1.6085e+05) [2.16e-09]	4.8422e+06 (5.0049+06) [9.46e-17]	5.0903e+07 (1.3437e+07) [2.18e-20]	3.8454e+06 (1.1124e+06) [4.26e-18]	9.1646e+06 (4.7291e+06) [7.36e-18]	2.3104e+05 (2.4935e+04) [4.87e-19]	**8.4931e+03 (4.7025e+03) NA**
f_4	3.0160e-05 (3.4479e-04) [7.26e-13]	5.1159e+00 (4.0194e+00) [6.27e-18]	2.7305e-04 (2.7305e-03) [2.16e-18]	4.6789e+02 (7.0123e+02) [2.98e-19]	1.2255e+04 (2.0365e+03) [8.25e-20]	6.9531e+02 (3.0987e+02) [2.76e-18]	3.2287e+04 (8.9020e+03) [3.25e-17]	2.2104e+05 (2.5435e+04) [1.36e-21]	**1.6149e-08 (1.0009e-05) NA**
f_5	5.0803e+02 (1.2439e+03) [2.15e-07]	3.2792e+02 (1.8494e+02) [2.16e-07]	7.8908e+03 (2.0238e+02) [5.27e-15]	5.1809e+03 (1.8742e+03) [8.11e-15]	8.0021e+03 (1.1184e+03) [2.78e-10]	3.3004e+03 (5.9445e+02) [4.26e-13]	1.5326e+04 (2.5523e+03) [6.035e-18]	6.6565e+03 (8.0392e+02) [4.28e-15]	**1.7762e+02 (1.2894e+01) NA**
f_6	2.1248e+00 (2.0013e+00) [3.62e-20]	2.7094e+01 (5.9445e+01) [6.27e-20]	3.1196e+01 (3.3987e+01) [1.65e-22]	1.5203e+00 (3.6781e+00) [7.55e-19]	3.7644e+02 (1.5149e+02) [4.26e-24]	4.8573e+01 (3.4278e+01) [1.25e-18]	1.5938e+03 (1.9133e+03) [2.16e-24]	1.4040e+02 (5.8322e+01) [8.27e-19]	**1.6598e-11 (1.0289e-12) NA**
f_7	4.6974e+03 (1.4824e+01) [5.27e-14]	4.6502e+03 (2.4824e+01) [3.26e-16]	4.6972e+03 (3.4824e+00) [5.22e-14]	2.0189e-02 (1.7645e-02) [2.33e-02]	6.1255e+02 (1.8494e+02) [4.27e-12]	4.8384e+03 (2.8499e+01) [8.21e-15]	4.6974e+03 (7.9875e-01) [3.99e-17]	1.6962e+02 (9.0252e+01) [2.36e-14]	**3.2974e-10 (1.2549e-12) NA**
f_8	2.0962e+01 (5.7258e-01) [8.02e-02]	2.0962e+01 (2.6628e-01) [8.02e-02]	2.0963e+01 (2.5067e-01) [8.01e-02]	2.1123e+01 (5.3912e-02) [6.79e-02]	2.0957e+01 (3.5087e-02) [8.89e-02]	2.1082e+01 (7.7543e-02) [6.34e-03]	2.1087e+01 (1.0988e-01) [6.29e-03]	2.1042e+01 (6.3001e-02) [9.61e-03]	**2.0952e+01 (1.2561e-05) NA**
f_9	4.2737e-01 (1.1369e+00) [2.61e-19]	**5.9272e-22 (8.9543e-22) NA**	1.8264e+01 (3.6452e+00) [2.14e-22]	3.7521e-04 (1.5589e-04) [5.39e-15]	1.1542e+02 (1.2343e+01) [1.61e-21]	1.0347e+01 (2.4973e+00) [4.31e-21]	7.4487e+01 (8.1867e+00) [2.88e-24]	6.6306e+01 (5.3969e+01) [7.34e-23]	3.2770e-16 (1.2561e-24) [1.39e-05]
f_{10}	1.0758e+02 (6.0809e+01) [2.72e-10]	7.0313e+01 (9.3551e+00) [8.36e-07]	1.0547e+02 (7.4660e+00) [5.27e-09]	1.1832e+02 (6.9342e+02) [7.23e-13]	2.1940e+02 (1.1478e+01) [7.42e-08]	4.4842e+01 (4.7092e+00) [5.28e-06]	3.2229e+02 (3.1581e+01) [4.25e-06]	2.0056e+02 (1.5651e+01) [8.46e-09]	**2.7670e+01 (1.0265e+00) NA**
f_{11}	3.3562e+01 (1.7275e+00) [6.37e-04]	3.0456e+01 (2.0062e+00) [2.09e-03]	3.3170e+01 (2.3952e+00) [1.11e-04]	3.5512e+01 (7.2092e+00) [8.22e-04]	3.2407e+01 (1.6009e+00) [2.08e-05]	2.9064e+01 (2.5561e+00) [4.77e-04]	3.3743e+01 (1.8491e+00) [2.93e-05]	3.6959e+01 (6.8976e-01) [6.37e-04]	**2.4943e+01 (4.1289e-01) NA**
f_{12}	2.3345e+03 (1.3383e+03) [5.82e-09]	2.9678e+04 (1.4836e+04) [3.27e-14]	1.9876e+03 (1.5974e+03) [6.83e-14]	1.0822e+04 (5.2318e+03) [1.22e-10]	1.5916e+05 (4.3912e+04) [6.35e-13]	2.1265e+04 (1.9608e+04) [7.89e-10]	5.8578e+04 (3.0139e+04) [6.93e-09]	9.9216e+04 (1.6369e+04) [5.83e-10]	**7.5159e+02 (1.0235e+01) NA**
f_{13}	3.7070e+00 (2.3420e+00) [5.27e-12]	3.6285e+00 (4.8739e-01) [5.63e-11]	4.5068e+00 (3.0313e+00) [8.26e-14]	5.4291e-01 (3.0217e-03) [2.42e-04]	5.7836e+01 (1.4627e+01) [8.05e-19]	1.8747e+00 (1.8947e-01) [2.31e-13]	7.9945e+00 (5.3477e+00) [5.37e-16]	3.0216e+00 (1.9967e-01) [2.64e-12]	**1.0348e-07 (1.0231e-02) NA**
f_{14}	1.2960e+01	1.2971e+01	1.3545e+01	1.3428e+01	1.2774e+01	1.3214e+01	1.2996e+01	1.3823e+01	**1.2381e+01**

	(2.5936e-02) [1.33e-01]	(2.2057e-01) [1.06e-01]	(9.9402e-02) [2.61e-02]	(4.9364e-01) [2.45e-02]	(1.8352e-01) [3.27e-01]	(2.1496e-01) [3.51e-02]	(3.1365e-01) [1.06e-01]	(6.1088e-01) [1.55e-02]	**(1.0483e-01)** **NA**
f_{15}	3.7075e+02 (6.6945e+01) [8.69e-04]	2.8884e+02 (9.0503e+01) [3.94e-03]	4.0050e+02 (9.1068e+01) [4.65e-05]	4.2031e+01 (4.1982e+01) [4.54e-04]	4.8678e+02 (7.1429e+01) [2.00e-06]	3.2000e+02 (1.3038e+02) [1.21e-03]	5.6956e+02 (7.6085e+01) [1.84e-08]	2.8299e+01 (3.7039e+01) [4.01e-03]	**6.8882e+00** **(1.0234e-02)** **NA**
f_{16}	1.2854e+02 (4.0730e+01) [2.34e-03]	1.5783e+02 (3.9432e+01) [1.36e-04]	1.2090e+02 (9.7020e+01) [8.96e-03]	2.6762e+02 (8.0932e+01) [7.83e-05]	3.8733e+02 (3.6337e+01) [1.21e-08]	1.4409e+02 (1.4413e+02) [8.14e-04]	4.7346e+02 (6.8079e+01) [1.22e-10]	3.8626e+02 (1.9642e+01) [1.29e-08]	**6.5629e+01** **(6.0235e+00)** **NA**
f_{17}	1.6189e+02 (4.7251e+01) [2.61e-04]	1.8692e+02 (3.5763e+01) [1.22e-06]	2.3590e+02 (6.3630e+02) [2.37e-08]	3.1305e+02 (1.7634+02) [5.34e-09]	3.4402e+02 (7.8310e+01) [2.31e-10]	1.6518e+02 (4.8966e+01) [1.55e-04]	5.7192e+02 (5.9669e+01) [1.33e-13]	3.0070e+02 (2.2203e+01) [6.21e-09]	**1.3404e+02** **(1.2356e+01)** **NA**
f_{18}	9.5440e+02 (3.4380e+01) [1.22e-06]	9.0679e+02 (1.6523e+00) [3.22e-05]	8.7611e+02 (7.8705e+01) [3.22e-04]	9.0833e+02 (4.9832e+00) [7.56e-04]	9.6592e+02 (7.5363e+00) [1.62e-07]	9.1657e+02 (2.7886e+00) [2.11e-05]	1.1052e+03 (2.4850e+01) [2.22e-11]	9.1548e+02 (8.6007e+00) [1.65e-05]	**8.0002e+02** **(1.2356e-10)** **NA**
f_{19}	8.4580e+02 (6.2150e+01) [1.36e-03]	9.0644e+02 (1.5419e+00) [3.55e-05]	8.6801e+02 (3.1790e+01) [2.61e-04]	9.1042e+02 (2.3474+00) [4.12e-05]	9.6456e+02 (6.0133e+00) [1.91e-08]	9.1865e+02 (1.3973e+00) [1.09e-06]	1.0943e+03 (1.7811e+01) [1.62e-12]	8.8899e+02 (8.9187e+01) [9.01e-05]	**8.0000e+02** **(0.0000e+00)** **NA**
f_{20}	2.0400e+03 (8.7680e+02) [1.26e-16]	9.0197e+02 (1.7231e+00) [1.56e-05]	8.7746e+02 (8.5400e+01) [4.51e-04]	9.1279e+02 (2.1287e+00) [2.13e-06]	9.6980e+02 (4.5616e+00) [1.29e-07]	9.1888e+02 (2.0154e+00) [4.55e-06]	1.1492e+03 (5.2501e+01) [2.87e-14]	8.8240e+02 (6.2931e+01) [3.61e-04]	**8.0000e+02** **(0.0000e+00)** **NA**
f_{21}	1.7300e+03 (5.1180e+02) [1.26e-18]	8.5838e+02 (1.1013e+00) [2.94e-09]	8.6002e+02 (1.1361e+00) [1.02e-09]	5.0000e+02 (2.2734e-13) NA	7.2374e+02 (4.7423e+01) [8.51e-07]	**5.0000e+02** **(1.0247e-13)** **NA**	1.2708e+03 (3.1570e+01) [1.34e-16]	6.8604e+02 (1.4671e+02) [2.88e-05]	**5.0000e+02** **(0.0000e+00)** **NA**
f_{22}	9.1340e+02 (1.9115e+02) [6.14e-09]	8.6558e+02 (6.6102e-01) [1.12e-08]	9.0052e+02 (1.2520e+01) [1.24e-09]	9.1823e+02 (2.8722e+01) [7.43E-09]	1.0487e+03 (1.6308e+01) [2.69e-19]	9.2589e+02 (1.3485e+01) [1.11e-10]	1.2695e+03 (1.0444e+02) [1.55e-20]	9.1061e+03 (6.3823e+01) [8.94e-09]	**7.2832e+02** **(5.1253e+01)**
f_{23}	5.5060e+02 (6.4890e+01) [1.61e-04]	5.2762e+02 (1.6550e+02) [5.66e-03]	5.1835e+02 (6.5481e+01) [1.25e-03]	5.3410e+02 (7.3465e-03) [4.96E-05]	7.1178e+02 (3.5533e+01) [8.91e-08]	6.2216e+02 (2.2357e-01) [1.52e-06]	1.2531e+03 (4.3288e+00) [1.67e-18]	8.2504e+02 (4.6358e+01) [1.44e-12]	**5.0002e+02** **(2.3541e-06)** **NA**
f_{24}	**2.0000e+02** **(0.0000e+00)** **NA**	2.6141e+02 (1.5664e+02) [1.32e-04]	**2.0000e+02** **(0.0000e+00)** **NA**	3.4532e+02 (2.4321e+02) NA	5.5351e+02 (3.9123e+01) [5.66e-06]	**2.0000e+02** **(2.0490e-09)** **NA**	1.3234e+03 (1.0896e+01) [1.99e-19]	1.2950e+03 (1.9283e+01) [4.31e-19]	**2.0000e+02** **(0.0000e+00)** **NA**
f_{25}	1.6512e+03 (5.6830e+02) [1.55e-18]	1.6525e+03 (2.5584e+02) [1.51e-18]	1.6761e+03 (2.4090e+02) [1.17e-18]	2.6721e+02 (8.2313e+01) NA	5.7227e+02 (4.2111e+01) [2.61e-08]	1.6709e+03 (1.3981e+01) [1.29e-18]	1.3577e+03 (3.5951e+01) [1.66e-17]	7.2274e+02 (2.9561e+02) [2.68e-10]	**2.0000e+02** **(0.0000e+00)** **NA**

A.2.2 Results on 100D problems

Considering unit hypercube translation, the volume of search space increases by 1e+70 times in case of 100*D* problems as compared to 30*D* case. This leads to significant deterioration in the performance of most optimizers. This is evident by comparing accuracy of the mean error values reported in Table A1 with that of Table A2. *Si*PABC_*Sf* is less prone to the curse of dimensionality and shows comparatively less deterioration in performance than most algorithms. For functions f_4 and f_6, the error values are high compared to the 30*D* and 50*D* problems. The flatness of landscape makes most of the algorithms wandering in the search space and they remain quite far off from the actual optimum. For Schwefel's problem (f_{12}), most of the algorithms are getting stuck at the second best optimum which accounts for the error value being more or less same for most algorithms. For hybrid benchmarks, *Si*PABC_*Sf* performs slightly poor but keeping in mind the exponential rise in local peak count, we can adjudge its performance to be commendable. Harmonious synergism of biologically observed processes contributes to its elated optimizing power in case of dimensional increase. It obtains the best results on 24 out of 25 functions and is at par JADE on function f_{24}. A close analysis of function f_{24} shows that while four algorithms (namely SaDE, jDE, JADE and CLPSO) managed to display competitive performances to our *Si*PABC_*Sf* algorithm on a 50*D* search space while it decreased to only a single algorithm (JADE) performing at par when the dimension was incremented to 100. The adjusted *P*-values obtained from 4 distinct statistical procedures have been reported in Table A2 and they affirm to our observation.

TABLE A2. MEAN, STANDARD DEVIATION AND *P*-VALUES OF WILCOXON'S RANK SUM TEST FOR BEST OF RUN ERROR VALUES FOR 100*D* FUNCTIONS

Algo Func	SaDE Mean (Std) [*p*-value]	JADE Mean (Std) [*p*-value]	jDE Mean (Std) [*p*-value]	HPA Mean (Std) [p-value]	ABC Mean (Std) [*p*-value]	CLPSO Mean (Std) [*p*-value]	GSO Mean (Std) [*p*-value]	m-ABC Mean (Std) [*p*-value]	SiPABC_*Sf* Mean (Std) [*p*-value]
f_1	8.2615e-08 (5.0445e-09) [7.92e-13]	6.4825e-10 (4.0249e-09) [6.77e-17]	7.4627e-07 (3.9371e-09) [2.94e-14]	4.7645e-10 (2.7664e-09) [3.45e-16]	1.9542e-04 (1.2112e-09) [2.88e-17]	3.0147e-08 (9.5612e-09) [1.72e-13]	1.3251e+05 (3.8741e+03) [2.52e-21]	8.6117e-07 (8.4810e-08) [3.62e-13]	**5.6843e-14** **(1.2356e-12)** **NA**

f_2	2.9471e+04 (7.7352e-03) [1.78e-08]	3.4923e+03 (8.4725e-06) [5.42e-14]	5.9371e+03 (8.4625e-07) [3.9e-13]	6.4574e+04 (3.8636e+04) [8.45e-11]	1.4161e+05 (9.2385e+03) [3.6e-11]	6.0345e+04 (5.3218e+03) [1.82e-12]	2.3912e+05 (3.0861e+04) [2.36e-16]	2.0484e+05 (1.8989e+04) [8.63e-15]	**1.7455e-03** **(1.3259e-06)** **NA**
f_3	7.9171e+06 (8.0936e+02) [2.34e-05]	2.9371e+06 (7.4728e+04) [8.76e-04]	8.9372e+06 (3.0927e+04) [4.32e-07]	5.0234e+08 (2.2645e+08) [1.03e-05]	4.1917e+09 (3.6152e+08) [3.56e-05]	2.6768e+09 (2.2696e+08) [5.55e-08]	3.1497e+10 (2.4891e+08) [7.31e-12]	4.9101e+09 (1.3447e+09) [3.18e-11]	**1.0365e+06** **(2.1562e+04)** **NA**
f_4	6.0283e+04 (6.9573e+03) [4.76e-04]	5.0342e+04 (6.9785e+03) [6.22e-03]	7.9261e+04 (2.7456e+03) [7.82e-05]	2.6477e+05 (6.4633e+04) [4.23e-10]	1.8595e+05 (6.0857e+03) [3.91e-07]	1.0985e+05 (8.9805e+03) [5.87e-06]	3.1489e+05 (2.1564e+03) [6.14e-07]	5.8332e+05 (2.0144e+04) [4.82e-08]	**9.5611e+03** **(1.2365e+03)** **NA**
f_5	9.7364e+05 (2.0963e+03) [4.35e-07]	7.5251e+05 (3.9464e+03) [8.02e-06]	2.9361e+05 (6.4536e+03) [2.43e-05]	2.0745e+04 (5.4722e+03) [2.34e-05]	3.8839e+04 (7.5372e+02) [7.92e-05]	1.3942e+04 (1.0412e+03) [5.17e-05]	9.4561e+05 (2.3217e+03) [4.88e-07]	6.0007e+04 (2.0265e+03) [1.61e-06]	**1.0501e+04** **(1.0265e+03)** **NA**
f_6	2.0172e+04 (8.2514e+01) [1.43e-07]	7.5329e+04 (7.3725e+01) [5.23e-09]	8.1876e+04 (8.4752e+00) [2.83e-06]	2.4532e+02 (7.8736e+01) [8.47e-16]	2.2410e+09 (2.6316e+08) [3.05e-16]	2.9224e+02 (6.0144e+01) [8.33e-03]	1.3657e+08 (2.4594e+07) [3.96e-17]	1.0268e+03 (2.6074e+01) [6.16e-05]	**1.3138e+02** **(1.2356e-01)** **NA**
f_7	9.8463e+04 (8.4623e+02) [3.64e-15]	9.0417e+04 (8.4735e+02) [1.23e-16]	1.1652e+05 (7.4637e+02) [2.74e-18]	2.8643e+04 (7.9544e+03) [1.09e-18]	1.9150e+04 (7.9732e+02) [2.78e-16]	1.8648e+04 (1.9702e+02) [5.92e-15]	1.5892e+05 (2.3654e+02) [5.63e-17]	2.2943e+04 (8.4794e+02) [8.30e-14]	**3.1055e+00** **(1.2598e+00)** **NA**
f_8	2.2075e+01 (9.3736e+00) [1.75e-02]	2.1964e+01 (9.4673e+01) [1.33e-02]	2.2197e+01 (7.4627e+00) [2.73e-03]	2.2087e+01 (3.7632e+00) [2.89e-03]	2.3035e+01 (1.2563e+00) [1.52e-04]	2.2355e+01 (6.2312e-01) [4.34e-04]	2.2295e+01 (6.1456e+00) [5.48e-03]	2.1308e+01 (3.6231e-02) [1.73e-02]	**2.1306e+01** **(2.8086e-02)** **NA**
f_9	9.2737e+02 (5.1369e+01) [3.52e-19]	8.9272e+02 (4.5043e+01) [5.47e-17]	9.3264e+02 (7.3645e+01) [2.51e-18]	1.0374e+00 (3.2315e-01) [7.94e-09]	7.1406e+02 (7.5038e+01) [3.26e-18]	1.9718e+01 (1.9450e+00) [1.62e-16]	4.3245e+03 (4.1259e+01) [1.62e-21]	1.0441e+00 (1.4764e+00) [8.72e-09]	**1.2220e-02** **(1.2356e-03)** **NA**
f_{10}	8.3426e+02 (1.6378e+01) [2.53e-03]	8.9350e+02 (8.8760e+01) [4.85e-05]	8.9597e+02 (7.7653e+01) [5.26e-03]	9.0386e+02 (6.7433+01) [8.48e-05]	1.2050e+03 (3.3705e+01) [3.49e-06]	8.6463e+02 (2.0677e+01) [2.47e-03]	1.3578e+03 (2.4592e+01) [1.96e-07]	2.7074e+03 (5.7948e+01) [8.96e-08]	**7.1601e+02** **(1.3562e+01)** **NA**
f_{11}	8.4684e+01 (3.4850e+00) [3.61e-05]	9.1208e+01 (4.7744e+00) [9.32e-06]	7.3630e+01 (9.4008e+00) [4.22e-04]	1.0279e+02 (4.7403e+01) [2.34e-08]	1.5873e+02 (2.2230e+00) [6.28e-12]	1.6102e+02 (1.3191e+00) [8.41e-15]	8.1256e+01 (5.1456e+00) [7.26e-04]	1.6013e+02 (2.0970e+00) [2.87e-14]	**5.5698e+01** **(6.1256e+00)** **NA**
f_{12}	6.7781e+05 (5.9092e+03) [3.77e-07]	6.7680e+05 (2.1050e+04) [2.89e-07]	7.4730e+05 (7.9280e+05) [8.49e-08]	1.4682e+05 (4.9604e+04) [2.42e-08]	6.7371e+06 (6.8148e+05) [2.48e-08]	2.6960e+05 (1.1081e+05) [2.78e-05]	4.2561e+05 (5.1245e+04) [9.14e-04]	7.8826e+05 (4.3300e+04) [1.05e-09]	**4.9013e+04** **(1.3256e+03)** **NA**
f_{13}	3.5713e+00 (5.1120e+00) [3.72e-03]	3.4141e+00 (4.0834e+00) [4.58e-03]	3.8650e+01 (4.3220e-01) [4.86e-06]	3.1645e+01 (6.2968e+00) [2.42e-04]	3.9376e+03 (8.7901e+02) [5.47e-10]	2.9377e+01 (3.9937e+00) [4.18e-05]	3.2564e+01 (4.1289e-02) [4.78e-03]	1.7436e+01 (3.7824e+00) [8.05e-03]	**1.1352e+00** **(1.0123e+00)** **NA**
f_{14}	4.2874e+01 (2.5343e-01) [6.82e-03]	4.2684e+01 (3.8163e-01) [5.63e-03]	4.3309e+01 (2.7163e-01) [2.95e-03]	4.4859e+01 (8.0349e-01) [4.54e-04]	4.7632e+01 (1.7446e-01) [1.26e-04]	4.7669e+01 (3.1133e-02) [5.38e-04]	4.71256e+01 (3.4586e-01) [8.56e-03]	4.7754e+01 (4.4378e-02) [3.25e-04]	**2.1112e+01** **(1.0236e-02)** **NA**
f_{15}	4.0691e+02 (3.4999e+00) [3.82e-03]	3.8929e+02 (4.1158e+00) [3.21e-03]	4.0000e+02 (0.0000e+00) [3.54e-03]	3.8776e+02 (9.8302e+00) [1.24e-04]	5.4729e+02 (2.0197e+01) [6.42e-03]	4.0000e+02 (5.0468e-04) [3.42e-03]	4.0000e+02 (1.4589e-10) [3.57e-03]	4.7152e+02 (4.4541e+00) [2.17e-05]	**2.0000e+02** **(1.0269e-14)** **NA**
f_{16}	2.4890e+02 (5.7687e-01) [1.53e-05]	2.3897e+02 (8.0553e+00) [9.62e-04]	2.5452e+02 (3.9367e+00) [3.11e-05]	9.9464e+02 (3.7865e+01) [8.24e-06]	3.4452e+02 (1.2041e+01) [4.26e-05]	2.6040e+02 (1.1838e+01) [1.63e-05]	2.5491e+02 (1.6542e+01) [8.56e-04]	1.0951e+03 (1.7844e+01) [4.85e-07]	**2.1936e+02** **(1.6523e-01)** **NA**
f_{17}	2.6752e+02 (4.2637e+00) [5.37e-03]	2.9210e+02 (1.9007e+00) [3.55e-03]	2.9182e+02 (2.0643e+00) [5.82e-03]	4.3815e+02 (2.4577e+01) [4.67e-05]	4.1417e+02 (1.4826e+01) [7.41e-04]	5.8247e+02 (4.6137e+00) [3.57e-05]	3.2458e+02 (5.3245e+01) [2.14e-05]	4.7561e+02 (5.1753e+01) [1.86e-05]	**2.4826e+02** **(1.9863e+00)** **NA**
f_{18}	9.1600e+02 (1.1368e-13) [1.22e-02]	1.1059e+03 (5.6975e+01) [1.06e-08]	1.1810e+03 (1.2932e+01) [8.40e-05]	1.092e+03 (5.6832e+01) [6.21e-05]	1.2827e+03 (4.0573e+01) [3.26e-06]	1.2275e+03 (7.1542e+01) [8.31e-05]	1.1573e+03 (1.6245e+01) [1.28e-06]	1.0024e+03 (1.0273e+01) [9.82e-07]	**8.6542e+02** **(1.2635e+01)** **NA**
f_{19}	9.8956e+02 (1.7685e+02) [5.31e-02]	1.0463e+03 (1.4694e+02) [4.72e-04]	1.1963e+03 (1.2824e+01) [2.83e-04]	1.2703e+03 (1.9535e+02) [2.04e-04]	1.3714e+03 (1.8801e+01) [2.75e-04]	1.2165e+03 (1.0948e+01) [3.94e-04]	1.2412e+03 (2.1254e+01) [1.23e-04]	1.0345e+03 (2.4927e+01) [2.07e-05]	**9.1196e+02** **(1.0356e+02)** **NA**
f_{20}	1.2596e+03 (1.1368e+02) [2.88e-05]	1.10529e+03 (6.0171e+01) [1.58e-05]	1.2048e+03 (5.5194e+01) [3.04e-05]	1.1840e+03 (5.3945e+01) [2.36e-05]	1.2905e+03 (8.8008e+01) [4.73e-05]	1.2211e+03 (1.2417e+02) [5.83e-05]	1.4121e+03 (1.1124e+02) [2.15e-06]	9.3542e+02 (1.2789e+01) [7.12e-04]	**9.0340e+02** **(6.2563e+01)** **NA**
f_{21}	9.1465e+02 (4.5579e-08) [1.39e-04]	9.2514e+02 (3.6311e+00) [2.11e-04]	8.5410e+02 (3.5774e-07) [2.74e-05]	8.8542e+02 (3.7523+02) [1.09e-06]	1.0235e+03 (1.3259e+03) [3.88e-07]	9.7942e+02 (1.5232e-02) [2.81e-05]	8.1795e+03 (5.3249e+02) [9.32e-12]	9.3167e+02 (1.5982e-06) [3.54e-03]	**5.0000e+02** **(1.0236e-12)** **NA**
f_{22}	1.0304e+03 (1.9518e+01) [5.82e-04]	9.5235e+02 (1.6132e+01) [4.65e-05]	9.9255e+02 (2.6481e+01) [2.88e-03]	1.1284e+03 (4.0328e+02) [6.93e-08]	1.3259e+03 (2.3651e+02) [1.90e-09]	1.0487e+03 (2.4866e+01) [2.31e-06]	1.6859e+03 (7.3245e+02) [4.85e-10]	1.6008e+03 (8.3245e+02) [3.54e-11]	**8.8112e+02** **(1.2563e+02)** **NA**
f_{23}	8.2956e+02 (7.4166e-01) [9.53e-05]	8.3237e+02 (3.7111e+00) [7.56e-05]	8.2936e+02 (3.2134e-01) [4.78e-05]	9.8784e+02 (1.4923e+02) [4.23e-07]	1.3256e+03 (5.2189e+01) [3.67e-07]	7.2970e+02 (7.7549e-02) [2.11e-04]	1.2761e+03 (6.2456e+01) [8.96e-06]	1.5621e+03 (6.3891e+01) [2.86e-07]	**5.3915e+02** **(1.2658e+01)** **NA**
f_{24}	1.2704e+03 (4.9727e+00) [2.69e-03]	2.0023e+02 (3.5623e-01) [8.92e-02]	5.6091e+02 (1.1040e+02) [2.05e-02]	5.3561e+02 (7.3866+01) [5.91e-02]	9.6912e+02 (1.2358e+02) [4.62e-04]	1.2230e+03 (1.3273e+01) [8.33e-07]	1.2734e+03 (6.2173e+00) [2.14e-08]	1.5403e+03 (3.7778e+00) [7.19e-09]	**2.0000e+02** **(0.0000e+00)** **NA**
f_{25}	1.2611e+03 (7.3691e+00) [1.78e-12]	1.7507e+03 (1.6658e+00) [6.51e-14]	1.5601e+03 (4.0194e-03) [4.62e-16]	7.4395e+02 (3.7240e+02) [4.50e-09]	1.9573e+02 (1.2567e+01) [6.39e-14]	1.6572e+03 (5.9688e+00) [4.69e-15]	1.2953e+03 (3.6233e+00) [3.25e-13]	1.5450e+03 (2.9570e+01) [2.85e-13]	**2.0000e+02** **(0.0000e+00)** **NA**

A.3 Convergence analysis

For analysis of convergence, we conduct experiments A- to determine the speed of convergence and quality measure, and experiment B- to obtain the trapping rate in recently proposed deceptive function class.

Problem definitions used in Experiment A

TABLE A. GLOBAL OPTIMUM, SEARCH RANGES, AND FORMULAE OF THE TEST FUNCTIONS OF C-SET [R1]

Function	Global optimum		Search range	Formula
	$x*$	$f(x*)$		
C-f_1: Sphere function	o	0	$[-100,100]$	$C-f_1(x) \coloneqq \sum_{i=1}^{D} x_i^2$
C-f_2: Schwefel's problem 2.22	o	0	$[-10,10]$	$C-f_2(x) \coloneqq \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_j \rvert$
C-f_3: Schwefel's problem 1.2	o	0	$[-100,100]$	$C-f_3(x) \coloneqq \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2$
C-f_4: Schwefel's problem 2.21	o	0	$[-100,100]$	$C-f_4(x) \coloneqq \max_i \{ \lvert x_i \rvert, 1 \le i \le D \}$
C-f_5: Generalized Rosenbrock's Function	$(1,1,.....1,1)$	0	$[-30,30]$	$C-f_5(x) \coloneqq \sum_{i=1}^{D-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right]$
C-f_6: Step function	o	0	$[-100,100]$	$C-f_6(x) \coloneqq \sum_{i=1}^{D} \left(\lfloor x_i + 0.5 \rfloor \right)^2$
C-f_7: Quartic Function with Noise	o	0	$[-1.28,1.28]$	$C-f_6(x) \coloneqq \sum_{i=1}^{D} i x_i^4 + rand(0,1)$
C-f_8: Generalized Schwefel's problem	$(420.9687,.....420.9687)$	−12569.5	$[-500,500]$	$C-f_8(x) \coloneqq -\sum_{i=1}^{D} \left(x_i \sin \sqrt{\lvert x_i \rvert} \right)$
C-f_9: Generalized Rastrigin's Function	o	0	$[-5.12,5.12]$	$C-f_8(x) \coloneqq \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10 \right)$
C-f_{10}: Ackley's function	o	0	$[-32,32]$	$C-f_{10}(x) \coloneqq -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2} \right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi x_i) \right) + 20 + e$
C-f_{11}: Griewank's Function	o	0	$[-600,600]$	$C-f_{11}(x) \coloneqq \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$
General Penalized Functions C-f_{12} and C-f_{13}	$(1,1,.....1,1)$	0	$[-50,50]$	$C-f_{12}(x) \coloneqq \sum_{i=1}^{D} u(x_i,10,100,4) + \frac{\pi}{30}\{10\sin^2(\pi y_1) (y_D - 1)^2 + \sum_{i=1}^{D-1} (y_i - 1)^2 [1 + 10\sin^2(\pi y_1)]\}$ $C-f_{13}(x) \coloneqq \sum_{i=1}^{D} u(x_i,5,100,4) + 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{D-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] + (x_D - 1)^2 [1 + \sin^2(2\pi x_D)]\}$ where $u(x_i,a,k,m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a \le x_i \le a \\ k(-x_i - a)^m & x_i < -a \end{cases}$ $y_i = 1 + \frac{1}{4}(x_i + 1)$

A.3.1 Experiment A

The convergence nature of the peer algorithms are analyzed by measuring their Convergence Speed (C_s) and Quality Measure (Q_m) using a subset of the problems in the benchmark set [R46]. These problems can be broadly classified as: Unimodal Separable $(C-f_1, C-f_2, C-f_4, C-f_6, C-f_7)$, unimodal Non-separable $(C-f_3)$, multi-modal Separable functions $(C-f_8, C-f_9)$, and multi-modal Non-separable functions $(C-f_5, C-f_{10} - C-f_{13})$.

As EAs are stochastic in nature, 100 independent runs were performed per algorithm per test function (by initializing the population for every run with uniform random initialization within the search space). The algorithm convergence, usually mean the convergence of the objective function that we minimize. The rate of convergence is generally described by $f(iter)$, where $iter$ is the current iteration. We use Q_m to study the convergence nature of the competing algorithms. Quality measure is an empirical measure of the algorithm's convergence [R47]. It serves to compare the objective function convergence of different evolutionary methods and is used in our study.

The formula of Q *-Measure* is $Q_m = C / P_C$, where P_C is the probability of convergence and C is the convergence measure. The value of C is calculated as $C = SumE_j / n_c . P_C$ equals $(n_C / n_T) \times 100$ where n_c, n_T and $SumE_j$ are the number of successful runs, total number of runs and the total number of FEs taken for all the successful runs respectively for an algorithm for all the functions considered under each category (separable unimodal, separable multimodal, non-separable unimodal, non-separable multimodal).

Similarly, the convergence speed (C_s) is the mean percentage of total number of FEs required by each of the algorithm, for 100 runs, to reach their best objective function value. This measure is used to detect which algorithms are more competitive, in each set of function classes. The variants which are good in convergence will have less mean percentage i.e., they will reach their optimum value with less number of FEs. The benchmark set used here shall be referred to as C-set (C stands for convergence which is analyzed here).

Now we perform the test for convergence by setting problem dimensionality *D* to 30 and the maximum FEs allotted for each algorithm is kept at 180,000. The values of convergence speed have been reported in Table A3 for all the functions using low FEs budget.

The results reveal that the speed of converges of our algorithm is the best for 10 out of 13 cases. JADE and JDE obtains better performance on $C-f_2$ and $C-f_7$ respectively. For functions $C-f_3$ to $C-f_5$, significant difference in the performance of *SiP*ABC_*Sf* as compared to other algorithms. The overall consistency in the performance in achieving higher convergence speed over multi-modal $(C-f_5, C-f_8 - C-f_{13})$ and unimodal functions $(C-f_1 \; to \; C-f_4, C-f_6, C-f_7)$ with or without variable linkages indicates the versatility of the search method. We have reported the corresponding *Q*-measures in Tables A4.

TABLE A3. CONVERGENCE SPEED MEASURED FOR THE PEER ALGORITHMS BASED ON 100 SAMPLE RUNS. THE LOWEST PERCENTAGE FOR A FUNCTION IS MARKED WITH "*" INDICATING THE BEST ALGORITHM.

Algo / Problems	SADE	JADE	jDE	HPA	ABC	CLPSO	GSO	m-ABC	*Si*PABC_*Sf*
C-f_1	16.16	10.47	19.23	34.64	75.59	24.29	100.00	64.29	10.91*
C-f_2	25.71	14.95*	27.11	30.45	100.00	41.76	100.00	96.57	23.99
C-f_3	87.34	61.32	100.00	100.00	100.00	98.71	100.00	100.00	40.97*
C-f_4	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	51.57*
C-f_5	96.32	92.28	100.00	96.03	100.00	92.65	100.00	100.00	60.31*
C-f_6	13.83	69.83	6.59	17.82	25.10	8.52	48.47	21.40	3.56*
C-f_7	9.86	22.54	9.61*	100.00	36.48	16.17	75.67	32.90	14.87
C-f_8	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
C-f_9	43.57	100.00	54.77	100.00	100.00	48.04	100.00	100.00	34.46*
C-f_{10}	32.44	80.18	30.44	28.66	100.00	39.08	100.00	100.00	17.42*
C-f_{11}	30.37	71.31	23.85	63.99	81.59	36.77	100.00	69.66	22.89*
C-f_{12}	18.60	36.75	17.74	24.93	67.73	26.34	100.00	58.44	10.86*
C-f_{13}	16.43	26.62	19.13	27.92	71.60	27.43	100.00	61.97	11.15*

TABLE A4. *Q*-MEASURE ACHIEVED BY THE PEER ALGORITHMS FOR THE FUNCTIONS OF C-SET

Functions	Unimodal functions		Multi-modal functions	
Algorithms	Separable	Non-separable	Separable	Non-separable
SADE	453.268	1103.78	1545.57	470.059
JADE	503.21	976.32	-	838.105
jDE	451.75	-	1240.72	500.21
HPA	606.698	-	-	757.507
ABC	1371.7	-	-	2228.28
CLPSO	510.4	13384.7	1729.36	767.67
GSO	2605.97	-	-	2495.61
m-ABC	1193.6	1103.78	-	1900.77
*Si*PABC_*Sf*	377.624	637.4	944.56	362.171

The quality measures indicate that all algorithms reach optimum for unimodal separable functions, *Si*PABC_*Sf* has the least value of Q_m due to high probability of detecting optima. Then comes the adaptive DE variants, CLPSO and HPA. However the classical SI approaches- GSO and ABC, have high values due to their explorative tendency which hampers their chances of detecting optima with the given accuracy under low FEs budget. When variable linkage appears, the problem landscape becomes more challenging. High values of Q_m reflect it. For multi-modal separable functions similar trend is observed due to difficulty in locating optima. But for non-separable multimodal problem better Q_m have been reported. Interestingly enough, the low FEs budget is a serious limitation to the real-parameter optimizers and in cases where limited FEs is available our algorithm has potential for use.

A.3.2 Experiment B

Earlier we could not predict the tendency of our algorithm getting trapped in deceptive problem landscapes. To do so, we perform this experiment using a simple class of functions–HappyCat where well-known search algorithms fail. It was recently proposed by Beyer and Finck [R48]. More details are provided in Supplementary attachment.

Problem definitions used in Experiment A

Happy Cat function [R48]: The function is of the following form

$$f_{HC}(x) := \left[\left(\|x\|^2 - N\right)^2\right]^{\alpha} + \frac{1}{N}\left(\frac{1}{2}\|x\|^2 + \sum_{i=1}^{N} x_i\right) + \frac{1}{2}. \tag{S2}$$

The case $N = 2$ is displayed in the above figure. As one can see, the α-part in the above equation produces an attracting groove for path-oriented search strategies. If $\alpha = 1/2$ the groove obtained is V-shaped. For $\alpha < 1/2$ the groove shape resembles the geometry of a black hole. Actually, it turns out that getting closer to the groove results in an increasing descent gradient towards the bottom of the groove. Its absolute value goes to infinity. That is why it is difficult to escape from this "black groove". Since the shape of the groove is tuneable by the exponent α, one can continuously control the problem hardness.

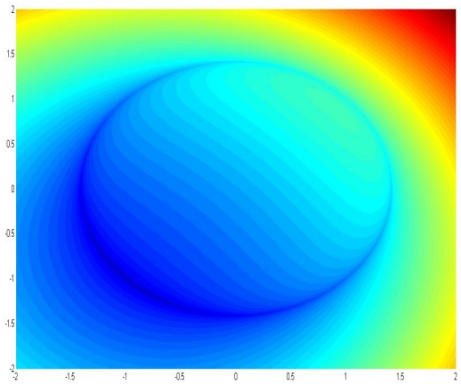

Fig A. The contour plot of Happy Cat function in two dimensions $(\alpha < 1/2)$ which gives rise to the name of this function.

Here we have used the competing algorithms and judged their dynamic behavior on the HappyCat function with the following configuration

- Dimension $N = 10$.
- Exponent $\alpha = 1/8$.
- Maximum FEs $N \times 10^4$.
- Accuracy threshold $\varepsilon = 0.1$.

We run every algorithm with their default parametric configuration 100 times and report the number of times that the competing algorithms have achieved the accuracy threshold ε within *MaxFEs* and also the rate of getting trapped at local optimal solution set is presented in Table A5. Additionally we have also reported the average distance accuracy of the solutions to the optimizer (-1, 1, ….1, 1). The results show that the accuracy with which the peaks have been located by *SiP*ABC_*Sf* is better than other competing algorithms by 2 orders of magnitude. ABC and GSO are highly susceptible to premature trapping and their local search fails in the cumulative adaptation of the benchmark function steps. The adaptive DE variants perform at par with each other. The empirical tests reveal that the *Si*PABC_*Sf* performs fairly well in comparison to state-of-the-art EAs in terms of speed of convergence and is less prone to falling into deceptive traps. There is a good balance of search moves that contributes to the versatility of our algorithm and helps it to perform well in a wide variety of problems.

TABLE A5. THE SUCCESS RATE AND TRAP RATE (MEASURED IN %) OF COMPETING ALGORITHMS ON THE HAPPY-CAT FUNCTION.

Algo / Metrics	SADE	JADE	jDE	HPA	ABC	CLPSO	GSO	m-ABC	*SiP*ABC_*Sf*
Success rate	29	34	21	11	4	33	0	8	83
Trap rate	71	66	79	89	96	67	100	92	17
Distance accuracy	9.853e-1 (2.238e-1)	9.134e-1 (1.954e-1)	9.599e-1 (2.029e-1)	1.103e+0 (4.737e-1)	1.344e+0 (2.722e-1)	1.189e+0 (2.447e-1)	1.803e+0 (5.285e-1)	1.202e+0 (2.685e-1)	3.463e-03 (8.462e-05)

A.4 Empirical Analysis of Time Complexity

TABLE A6. COMPUTATIONAL EFFORT: MEAN AND STANDARD DEVIATION OF RATIO R FOR $30D$ AND $50D$ PROBLEMS BASED ON 50 INDEPENDENT RUNS.

Algo		SaDE	JADE	jDE	HPA	ABC	CLPSO	GSO	m-ABC	SiPABC_*Sf*
Func.	D	Mean (Std)	Mean (Std)	Mean (Std)	Mean (Std)	Mean (Std)	Mean (Std)	Mean (Std)	Mean (Std)	Mean (Std)
f_1	30	49.711 (1.605)	2.102 (0.012)	4.704 (0.005)	11.403 (0.039)	4.380 (0.020)	39.606 (0.324)	4.359 (0.052)	4.574 (0.002)	16.401 (0.490)
	50	65.897 (0.506)	2.728 (0.017)	6.320 (0.007)	17.438 (0.023)	6.281 (0.006)	63.817 (1.977)	6.240 (0.028)	6.576 (0.077)	21.700 (0.164)
f_2	30	33.769 (0.059)	1.923 (0.004)	4.021 (0.004)	9.592 (0.094)	3.928 (0.017)	31.901 (0.080)	3.907 (0.012)	4.123 (0.007)	11.501 (0.021)
	50	33.796 (0.668)	2.263 (0.010)	4.541 (0.017)	11.167 (0.034)	4.627 (0.006)	31.035 (0.086)	4.398 (0.004)	4.605 (0.011)	11.747 (0.208)
f_3	30	32.203 (0.144)	1.828 (0.002)	3.613 (0.007)	8.796 (0.013)	3.571 (0.013)	30.597 (0.209)	3.420 (0.164)	3.659 (0.002)	10.980 (0.045)
	50	42.900 (1.003)	2.249 (0.022)	4.556 (0.013)	11.034 (0.025)	4.600 (0.004)	40.017 (1.070)	4.371 (0.087)	4.576 (0.009)	14.497 (0.317)
f_4	30	32.506 (0.153)	1.922 (0.018)	3.870 (0.013)	9.690 (0.013)	3.904 (0.007)	31.013 (0.223)	3.825 (0.012)	3.872 (0.009)	11.159 (0.059)
	50	33.247 (0.040)	2.186 (0.005)	4.460 (0.012)	10.798 (0.016)	4.514 (0.004)	31.267 (0.092)	4.298 (0.011)	4.472 (0.005)	11.567 (0.015)
f_5	30	19.761 (0.103)	1.772 (0.020)	3.363 (0.005)	7.759 (0.062)	3.246 (0.007)	18.616 (0.142)	3.227 (0.009)	3.347 (0.054)	7.221 (0.045)
	50	22.896 (0.013)	2.040 (0.001)	3.992 (0.002)	9.631 (0.017)	4.020 (0.014)	21.451 (0.098)	3.909 (0.005)	4.074 (0.006)	8.357 (0.005)
f_6	30	41.906 (1.594)	2.077 (0.010)	4.562 (0.047)	11.918 (0.486)	4.398 (0.022)	38.124 (0.237)	4.358 (0.044)	4.663 (0.024)	14.185 (0.492)
	50	54.460 (0.055)	2.639 (0.006)	5.925 (0.015)	17.172 (1.170)	5.808 (0.012)	51.044 (0.052)	5.746 (0.004)	6.067 (0.005)	18.393 (0.021)
f_7	30	25.635 (1.311)	1.816 (0.001)	3.387 (0.003)	8.841 (0.052)	3.289 (0.018)	24.681 (0.436)	3.261 (0.059)	3.419 (0.027)	9.091 (0.401)
	50	28.035 (0.302)	2.144 (0.002)	3.940 (0.022)	10.012 (0.086)	3.886 (0.006)	26.696 (0.680)	3.854 (0.004)	4.002 (0.035)	10.051 (0.094)
f_8	30	28.242 (0.215)	1.843 (0.008)	3.502 (0.013)	8.554 (0.222)	3.416 (0.042)	26.008 (0.187)	3.412 (0.019)	3.530 (0.010)	9.953 (0.072)
	50	36.029 (1.121)	2.212 (0.014)	4.190 (0.034)	9.924 (0.530)	4.093 (0.016)	32.248 (0.199)	4.051 (0.038)	4.225 (0.014)	12.602 (0.354)
f_9	30	37.483 (0.662)	2.083 (0.002)	4.429 (0.031)	11.608 (0.099)	4.202 (0.003)	34.485 (0.013)	4.265 (0.052)	4.470 (0.022)	13.040 (0.206)
	50	47.249 (0.087)	2.559 (0.014)	5.558 (0.012)	14.748 (0.064)	5.114 (0.009)	43.932 (0.057)	5.285 (0.022)	5.571 (0.063)	16.392 (0.037)
f_{10}	30	27.855 (0.353)	1.783 (0.001)	3.451 (0.002)	8.317 (0.035)	3.338 (0.008)	26.459 (0.073)	3.369 (0.009)	3.521 (0.003)	9.929 (0.111)
	50	35.690 (0.578)	2.052 (0.005)	4.009 (0.083)	9.786 (0.099)	3.966 (0.007)	33.076 (0.569)	4.036 (0.042)	4.188 (0.040)	12.562 (0.184)
f_{11}	30	1.721 (0.019)	1.050 (0.001)	1.149 (0.001)	1.428 (0.001)	1.141 (0.001)	1.690 (0.026)	1.139 (0.001)	1.150 (0.001)	1.262 (0.006)
	50	1.510 (0.001)	1.044 (0.001)	1.118 (0.001)	1.321 (0.001)	1.115 (0.001)	1.477 (0.001)	1.114 (0.001)	1.118 (0.001)	1.191 (0.001)
f_{12}	30	3.011 (0.015)	1.150 (0.001)	1.420 (0.002)	2.290 (0.013)	1.393 (0.003)	2.872 (0.005)	1.397 (0.005)	1.455 (0.005)	1.747 (0.005)
	50	2.002 (0.003)	1.090 (0.001)	1.234 (0.001)	1.700 (0.010)	1.219 (0.001)	1.933 (0.002)	1.222 (0.002)	1.252 (0.002)	1.383 (0.001)
f_{13}	30	26.481 (0.405)	1.839 (0.003)	3.470 (0.044)	8.746 (0.025)	3.534 (0.008)	25.640 (0.093)	3.620 (0.024)	3.676 (0.064)	9.895 (0.134)
	50	28.112 (0.035)	2.038 (0.002)	4.030 (0.002)	9.733 (0.014)	3.938 (0.006)	26.355 (0.036)	4.056 (0.011)	4.244 (0.032)	10.561 (0.013)
f_{14}	30	21.573 (0.216)	1.617 (0.013)	2.932 (0.055)	6.557 (0.064)	2.893 (0.024)	19.978 (0.269)	2.854 (0.045)	2.953 (0.038)	8.293 (0.081)
	50	21.305 (0.049)	1.787 (0.010)	3.291 (0.005)	7.232 (0.021)	3.250 (0.010)	20.250 (0.157)	3.259 (0.017)	3.304 (0.013)	8.316 (0.023)
f_{15}	30	1.344 (0.003)	1.022 (0.001)	1.063 (0.001)	1.196 (0.001)	1.063 (0.001)	1.320 (0.003)	1.063 (0.001)	1.067 (0.001)	1.133 (0.001)
	50	1.250 (0.001)	1.019 (0.001)	1.049 (0.001)	1.146 (0.003)	1.051 (0.001)	1.234 (0.001)	1.050 (0.001)	1.051 (0.001)	1.098 (0.001)
f_{16}	30	1.297 (0.008)	1.017 (0.001)	1.052 (0.006)	1.141 (0.014)	1.050 (0.003)	1.235 (0.017)	1.054 (0.003)	1.054 (0.001)	1.117 (0.003)
	50	1.253 (0.001)	1.018 (0.001)	1.051 (0.001)	1.145 (0.001)	1.050 (0.001)	1.237 (0.001)	1.050 (0.001)	1.053 (0.001)	1.102 (0.006)
f_{17}	30	1.282 (0.005)	1.021 (0.001)	1.067 (0.001)	1.157 (0.003)	1.053 (0.001)	1.265 (0.008)	1.051 (0.001)	1.067 (0.001)	1.118 (0.002)
	50	1.241 (0.002)	1.017 (0.001)	1.051 (0.001)	1.142 (0.001)	1.045 (0.001)	1.220 (0.005)	1.050 (0.001)	1.052 (0.001)	1.101 (0.001)
f_{18}	30	1.248 (0.002)	1.021 (0.001)	1.061 (0.001)	1.191 (0.001)	1.062 (0.004)	1.239 (0.002)	1.063 (0.002)	1.067 (0.002)	1.110 (0.001)
	50	1.219 (0.001)	1.017 (0.001)	1.049 (0.001)	1.135 (0.001)	1.042 (0.001)	1.203 (0.001)	1.047 (0.001)	1.050 (0.001)	1.096 (0.007)
f_{19}	30	1.268 (0.004)	1.021 (0.001)	1.065 (0.001)	1.198 (0.003)	1.064 (0.003)	1.229 (0.015)	1.063 (0.002)	1.066 (0.002)	1.124 (0.002)
	50	1.219 (0.001)	1.017 (0.001)	1.049 (0.001)	1.134 (0.001)	1.042 (0.001)	1.204 (0.001)	1.047 (0.001)	1.050 (0.001)	1.101 (0.001)
f_{20}	30	1.340 (0.001)	1.021 (0.001)	1.064 (0.001)	1.194 (0.004)	1.062 (0.002)	1.318 (0.003)	1.061 (0.001)	1.064 (0.002)	1.165 (0.001)
	50	1.220 (0.001)	1.017 (0.001)	1.049 (0.001)	1.135 (0.001)	1.042 (0.001)	1.205 (0.001)	1.047 (0.001)	1.050 (0.001)	1.108 (0.002)
f_{21}	30	1.337 (0.002)	1.019 (0.001)	1.061 (0.001)	1.203 (0.001)	1.059 (0.001)	1.313 (0.003)	1.064 (0.001)	1.061 (0.001)	1.175 (0.001)
	50	1.232 (0.001)	1.016 (0.001)	1.049 (0.001)	1.149 (0.001)	1.043 (0.001)	1.219 (0.002)	1.049 (0.001)	1.050 (0.001)	1.122 (0.002)
f_{22}	30	1.228 (0.056)	1.016 (0.001)	1.028 (0.002)	1.154 (0.002)	1.049 (0.001)	1.241 (0.003)	1.051 (0.001)	1.051 (0.001)	1.131 (0.030)
	50	1.177 (0.001)	1.014 (0.001)	1.039 (0.001)	1.108 (0.001)	1.033 (0.001)	1.164 (0.001)	1.038 (0.001)	1.039 (0.001)	1.102 (0.001)
f_{23}	30	1.330 (0.006)	1.015 (0.003)	1.060 (0.001)	1.189 (0.002)	1.059 (0.001)	1.305 (0.005)	1.063 (0.001)	1.058 (0.003)	1.206 (0.005)
	50	1.231 (0.001)	1.017 (0.001)	1.049 (0.001)	1.140 (0.001)	1.043 (0.001)	1.214 (0.001)	1.048 (0.001)	1.050 (0.001)	1.147 (0.001)
f_{24}	30	1.348 (0.006)	1.018 (0.002)	1.066 (0.001)	1.203 (0.005)	1.063 (0.002)	1.337 (0.007)	1.063 (0.001)	1.072 (0.013)	1.247 (0.005)
	50	1.371 (0.001)	1.025 (0.001)	1.077 (0.001)	1.227 (0.001)	1.063 (0.001)	1.343 (0.002)	1.072 (0.001)	1.076 (0.002)	1.265 (0.001)
f_{25}	30	1.333 (0.002)	1.022 (0.001)	1.062 (0.003)	1.223 (0.007)	1.070 (0.003)	1.355 (0.016)	1.072 (0.004)	1.069 (0.003)	1.271 (0.002)
	50	1.318 (0.009)	1.020 (0.001)	1.062 (0.003)	1.202 (0.005)	1.064 (0.002)	1.296 (0.007)	1.058 (0.002)	1.066 (0.001)	1.258 (0.008)

The computation time taken by the competing algorithms is shown in Table A6 for 30*D* and 50*D* problems. Note that the computation time is platform dependent and varies with operating system, machine processor, memory, CPU speed, programming language, compiler, condition of hardware etc. To account for this non-uniformity and homogenize the condition, we have reported ratio values calculated as

$$R = \frac{T_{total}}{T_{bench}} = \frac{T_{algo} + T_{bench}}{T_{bench}} = 1 + \frac{T_{algo}}{T_{bench}} \ , \tag{S3}$$

where T_{total} is the sum of the time taken to run the algorithm T_{algo} and T_{bench} benchmark separately.

We notice that as the benchmark problems become more complex i.e. the hybrid composition problems (resembling real-world engineering problems), the ratio R confines to the range $1 < R < 2$. This is because when objective functions are calculated, each parameter from 1 to *D* are considered for evaluation and may we add, in a far complex manner. Thus with increasing dimensionality, the value of *R* approaches 1 indicating that the ratio is only T_{algo} / T_{bench} marginal. On the other hand, we must be aware of the fact that the unimodal and classic multimodal test functions ($f_1 - f_9$) are not computationally expensive by themselves. One fitness evaluation of these functions is usually less than 10^{-4} s. However, in many real-world problems, the computation of fitness can be very time-consuming. In aerodynamic design optimization, for example, it takes over several hours to perform one fitness evaluation when it involves a 3-*D* computational fluid dynamics (CFD) simulation [R49]. This indicates that the complexity $\Theta(SN^2)$ does not significantly affect the efficiency of the algorithm in terms of computation time. Thus in the tradeoff between time and precision, we can say that offline optimization problems that require a highly accurate performance, our *SiPABC_Sf* can come in handy.

A.5 Results on real world optimization problems

Population-based metaheuristics find applications in real world scenarios for the development of technology. The authors have selected a finite set of problems, with bound constraints, that find widespread applications in the technical world we know. The test suite chosen here contains a subset of problems that appeared in CEC 2011 competition on real world problems. The problems used here are specified below (the problem number according to the Technical Report [R50] specified in parenthesis). We refer to the benchmark set as *R*-set (*R* stands for real-world)

TABLE A7. MEAN, STANDARD DEVIATION AND *P*-VALUES FROM WILCOXON'S RANK SUM TEST FOR BEST-OF-RUN-ERROR OBTAINED FOR REAL WORLD PROBLEMS

Algorithm / Problems	SADE	JADE	jDE	HPA	ABC	CLPSO	GSO	m-ABC	*Si*PABC_*Sf*	Fitness Evaluations
	Mean (Std) *p*-values	Mean (Std) *p*-values	Mean (Std) *p*-values	Mean (Std) *p*-values	Mean (Std) *p*-values	Mean (Std) *p*-values	Mean (Std) *p*-values	Mean (Std) *p*-values	Mean (Std) *p*-values	
*R-f*1: FM Waves	3.342e+00 (2.212e-01) 4.178e-09	5.641e-02 (7.712e-03) 4.165e-01	9.858e+00 (3.917e+00) 2.349e-11	3.456e+01 (2.324e+00) 2.681e-24	2.535e+01 (2.202e+00) 9.639e-21	2.325e+01 (2.273e+00) 1.613e-20	2.387e+01 (2.914e+01) 8.645e-23	1.328e+01 (1.524e+00) 6.664e-18	**1.297e-02 (3.316e-03) NA**	5.0e+4
	2.213e+00 (1.733e-02) 3.563e-12	3.30e-03 (4.102e-03) 1.351e-02	2.134e+00 (1.613e+00) 1.614e-12	2.606e+01 (2.162e+00) 4.668e-22	2.104e+01 (1.549e+00) 6.346e-19	2.150e+01 (2.093e+00) 3.231e-20	2.034e+01 (1.322e+01) 2.619e-18	1.1023e+01 (7.233e-01) 5.517e-16	**2.314e-04 (1.124e-06) NA**	1.0e+5
	7.893e-01 (2.235e-02) 6.296e-10	6.972e-05 (3.826e-05) 3.651e-06	9.742e-01 (1.297e-01) 2.116e-09	1.541e+01 (2.062e+00) 4.291e-21	2.072e+01 (1.497e+00) 2.823e-18	2.150e+01 (2.031e-01) 4.552e-19	1.805e+01 (3.994e+00) 1.624e-17	1.098e+01 (7.011e-01) 6.551e-15	**5.509e-09 (3.342e-08) NA**	1.5e+5
*R-f*2: Spread Spectrum Polly phase	1.674e+00 (2.232e+02) 1.325e-01	1.241e+00 (6.402e-01) 6.145e-01	1.603e+00 (7.177e-02) 1.962e-01	4.023e+00 (2.232e+00) 3.618e-09	1.998e+00 (1.311e-01) 2.622e-02	2.122e+00 (1.510e-01) 5.113e-03	2.723e+00 (2.003e+00) 6.318e-05	1.480e+00 (9.953e-02) 4.005e-01	**1.227e+00 (3.372e-01) NA**	5.0e+4
	1.434e+00 (1.052e+00) 3.453e-01	1.174e+00 (3.221e-01) 4.061e-01	1.420e+00 (1.351e-02) 2.891e-01	3.023e+00 (1.121e+00) 6.384e-04	1.826e+00 (5.191e-02) 9.256e-02	1.128e+00 (1.152e-01) 4.125e-01	1.856e+00 (1.431e+00) 1.031e-03	1.393e+00 (8.122e-02) 3.119e-01	**1.073e+00 (1.023e-02) NA**	1.0e+5
	1.335e+00 (3.432e-01) 4.709e-02	1.085e+00 (1.151e-01) 6.335e-02	1.268e+00 (5.701e-03) 4.831e-02	3.011e+00 (1.032e+00) 2.306e-06	1.759e+00 (5.813e-02) 5.346e-03	1.109e+00 (1.621e-02) 2.825e-01	1.338e+00 (1.131e-01) 3.197e-02	1.369e+00 (6.052e-02) 1.694e-02	**9.657e-01 (3.341e-03) NA**	1.5e+5
*R-f*3: Messenger	1.652e+01 (3.93e-02) 1.972e-01	1.543e+01 (3.231e-02) 2.116e-01	1.689e+01 (3.03e-02) 1.582e-01	1.678e+01 (3.01e-02) 1.118e-01	5.835e+01 (9.135e+00) 7.418e-03	5.277e+01 (4.163e+00) 3.633e-03	3.543e+01 (3.32e+00) 1.318e-03	1.946e+01 (2.582e+00) 9.894e-02	**1.342e+01 (2.652e-02) NA**	5.0e+4
	1.548e+01 (2.359e-02) 2.341e-01	**1.396e+01 (3.07e-02) NA**	1.597e+03 (2.323e-02) 2.289e-01	1.535e+01 (2.14e-02) 2.521e-01	3.750e+01 (3.802e+00) 4.765e-02	1.670e+01 (6.312e-01) 2.416e-01	2.234e+01 (2.243e+00) 1.061e-02	1.856e+01 (3.442e+00) 1.019e-01	1.424e+01 (2.142e-02) 6.175e-01	1.0e+5
	1.338e+01 (1.914e-02) 3.672e-01	1.326e+01 (2.03e-02) 3.984e-01	1.193e+01 (2.273e-02) 2.314e-01	1.328e+01 (1.913e-02) 3.672e-01	3.710e+01 (3.591e+00) 2.523e-01	1.620e+01 (4.232e-02) 3.571e-01	1.747e+01 (5.114e-01) 6.161e-02	1.849e+01 (3.091e+00) 4.367e-02	**1.043e+01 (2.934e-02) NA**	1.5e+5
*R-f*4: Cassini2	1.988e+01 (4.964e-02) 3.426e-01	1.946e+01 (4.828e-02) 3.851e-01	2.048e+01 (5.018e-02 3.325e-01	2.091e+01 (5.392e-02) 2.150e-01	4.500e+02 (2.645e+00) 7.884e-06	3.829e+01 (2.025e+00) 4.481e-03	4.232e+01 (2.341e+00) 5.671e-05	2.583e+01 (3.452e+01) 2.913e-02	**1.576e+01 (3.346e-02) NA**	5.0e+4
	1.909e+01 (4.378e-02) 2.134e-01	1.817e+01 (4.002e-02) 2.989e-01	1.939e+01 (4.324e-02) 2.196e-01	1.949e+01 (4.743e-02) 2.068e-01	3.045e+01 (2.531e+00) 7.631e-02	1.859e+01 (3.076e+00) 3.095e-01	3.122e+01 (1.132e+00) 1.006e-03	2.2807e+01 (2.792e+00) 4.319e-02	**1.513e+01 (2.261e-02) NA**	1.0e+5
	1.753e+01 (3.529e-02) 2.425e-02	1.683e+01 (3.925e-02) 3.643e-01	1.794e+01 (4.289e-02) 2.397e-02	1.783e+01 (4.549e-02) 2.401e-02	2.983e+01 (2.163e+00) 8.473e-02	1.839e+01 (3.172e-01) 2.042e-02	2.766e+01 (1.152e-01) 3.182e-03	1.867e+01 (4.576e-01) 2.161e-02	**1.501e+01 (1.145e-02) NA**	1.5e+5

a. *R-f*1: Parameter Estimation for FM Sound Waves (1)
b. *R-f*2: Spread Spectrum Radar Poly-phase Code Design (7)
c. *R-f*3 and *R-f*4: Spacecraft Trajectory Optimization Problems- Messenger Full (12) and Cassini 2 (13).

The problems discussed above include various challenging features like numerous local peaks, interdependence, nonlinearity and bound constraints. The data is assimilated based on 25 test runs. The mean of functional values and their standard deviation at 5.0e+4, 1.0e+5 and 1.5e+5 FEs have been recorded in Table A7 and the *P*-values calculated from Wilcoxon's rank-sum test conducted on the test results (at 5% significance level) is tabulated alongside. The test results indicate that our proposed *Si*PABC_*Sf* algorithm manages to outperform the eight other competing algorithms on functions *R-f*1, *R-f*2 and *R-f*4 at every run instance recorded. Only in case of *R-f*3 it is bettered by jDE at 1.0e+5 FEs but again regains its top position before reaching the termination criteria.

SUPPLEMENTARY REFERENCES

[R1] M. Clerc, *Particle Swarm Optimization*, ISTE Publications, 2008.

[R2] L. J. Fogel, A. J. Owens and M. J. Walsh. (1966) *Artificial Intelligence through Simulated Evolution*, John Wiley.

[R3] S. Das, and P. N. Suganthan," Differential evolution: a survey of the state-of-the-art", *IEEE Transactions on Evol. Comput.*, Vol. 15, No. 1, pp. 4-31, February 2011.

[R4] M. Dorigo and T. Stützle, *Ant Colony Optimization*, MIT Press, Cambridge, MA, 2004.

[R5] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm", *Applied Soft Computing*, vol. 8(1), pp. 687-697, 2008.

[R6] D. J. Mala, M. Kamalapriya, R. Shobana, and V. Mohan," A non-pheromone based intelligent swarm optimization technique in software test suite optimization. In: IAMA: 2009 international conference on intelligent agent and multi-agent systems, IEEE Madras Section; IEEE Computer Society, Madras Chapter; (2009), pp. 188–192.

[R7] J. Ruiz-Vanoye, and O. Daz-Parra," Similarities between meta-heuristics algorithms and the science of life," *Central European Journal on Operational Research*, vol. 19, pp. 445–466, 2011.

[R8] M. El-Abd, "Performance assessment of foraging algorithms vs evolutionary algorithms." *Information Sciences* 182(1): (2012) 243–263.

[R9] X. Liu, and Z. Cai, "Artificial bee colony programming made faster." In: 2009. ICNC '09. *Fifth international conference on natural computation*, 4, pp. 154–158, 2009.

[R10] G. Zhu, and S. Kwong, "Gbest-guided artificial bee colony algorithm for numerical function optimization", *Applied Mathematics and Computation, ivol*. 217, pp. 3166–3173, 2010.

[R11] W. Gao, and S. Liu,"Improved artificial bee colony algorithm for global optimization", *Inf Process Lett*, 2011, 111 (17): 871–882.

[R12] S. Raziuddin, S. A. Sattar, R. Lakshmi, and M. Parvez," Differential artificial bee colony for dynamic environment. In: *Advances in computer science and information technology, communications in computer and information science*, vol. 131. Springer, Berlin, pp 59–69, 2011.

[R13] M. Tuba, N. Bacanin, and N. Stanarevic," Guided artificial bee colony algorithm," In: *Proceedings of the European computing conference* (ECC'11), 2011, pp 398–403.

[R14] G. Li, P. Niu, and X. Xiao, "Development and investigation of efficient artificial bee colony algorithm for numerical function optimization." *Applied Soft Computing*, Vol. 12 (1), pp. 320-332, 2012.

[R15] A. Barnharnsakun, T. Achalakul, and B. Sirinaovakul,"The best-so-far selection in artificial bee colony algorithm," *Applied Soft Computing* Vol. 11 pp. 2888–2901, 2011.

[R16] F. Kang, J. Li, and Z. Ma, "Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions", *Information Sciences*, Vol. 181, No. 16, pp. 3508–3531, Aug., 2011.

[R17] P.-W. Tsai, J.-S. Pan, B.-Y. Liao and S.-C. Chu, "Enhanced bee colony optimization", *International Journal of Innovative Computing, Information and Control,* Vol. 5, No. 12, Dec. 2009.

[R18] B. Akay and D. Karaboga, "A modified artificial bee colony algorithm for real-parameter optimization", *Information Sciences*, Vol. 192, pp. 120–142, June 2012.

[R19] F. Kang, J. Li, Q. Xu,"Hybrid simplex artificial bee colony algorithm and its application in material dynamic parameter back analysis of concrete dams". *J Hydraul Eng* Vol. 40 Issue 6, pp.736–742, 2009.

[R20] H. B. Duan, C. F. Xu, and Z. H. Xing, " A hybrid artificial bee colony optimization and quantum evolutionary algorithm for continuous optimization problems," *Int J Neural Syst*.,Vol. 20, Issue 1, pp.39–50, 2010.

[R21] X. Shi, Y. Li, H. Li, R. Guan, L. Wang, and Y. Liang," An integrated algorithm based on artificial bee colony and particle swarm optimization," In: *2010 sixth international conference on natural computation (ICNC)*, vol 5, pp 2586–2590, 2010.

[R22] H. Zhao, Z. Pei, J. Jiang, R. Guan, C. Wang, and X. Shi," A hybrid swarm intelligent method based on genetic algorithm and artificial bee colony," *In: Advances in swarm intelligence, pt 1, proceedings, Lecture notes in computer science*, vol 6145, pp 558–565, 2010.

[R23] M. El-Abd,"A hybrid abc-spso algorithm for continuous function optimization," In: *2011 IEEE SSCI*, pp 1–6, 2011.

[R24] A. Oner, S. Ozcan, and D. Dengi,"Optimization of university course scheduling problem with a hybrid artificial bee colony algorithm." *In: 2011 IEEE congress on evolutionary computation (CEC)*, pp 339–346, 2011.

[R25] W. C. Yeh, J. C. P. Su, T. J. Hsieh, M. Chih, and S. L. Liu," Approximate reliability function based on wavelet latin hypercube sampling and bee recurrent neural network," *IEEE Trans Reliab*,Vol. 60, Issue 2, pp. 404–414, 2011.

[R26] W. Gao, S. Liu, and L. Huang, "A novel artificial bee colony algorithm based on modified search equation and orthogonal learning", *IEEE Transactions on Cybernetics*, 2013, DOI: https://10.1109/TSMCB.2012.2222373.

[R27] B. Akay, and D. Karaboga, "Solving integer programming problems by using artificial bee colony algorithm. In: *AI (ASTERISK) IA 2009: emergent perspectives in artificial intelligence. Italian Association of Artificial Intelligence*; University Modena Reggio Emilia, Lecture notes in artificial intelligence, vol 5883, pp 355–364.

[R28] D. Karaboga, and B. Gorkemli, "A combinatorial artificial bee colony algorithm for traveling salesman problem," In: *2011 international symposium on innovations in intelligent systems and applications* (INISTA), pp 50–53, 2011.

[R29] S. L. Ho, and S. Yang," An artificial bee colony algorithm for inverse problems," *Int J Appl. Electromagn. Mech.* Vol. 31, No. 3, pp.181–192, 2009).

[R30] B. C. Mohan, and R. Baskaran," Energy aware and energy efficient routing protocol for adhoc network using restructured artificial bee colony system," In: *High performance architecture and grid computing, communications in computer and information science*, vol 169. Springer, Berlin, pp 473–484, 2011.

[R31] D. Karaboga and B. Basturk, A modified artificial bee colony (ABC) algorithm for constrained optimization problems, *Applied Soft Computing* Vol. 11, No. 3, pp. 3021–3031, 2011.

[R32] S. Biswas, S. Kundu, S. Das, A. V. Vasilakos," Information sharing in bee colony for detecting multiple niches in non-stationary environments", In *Proceeding of the fifteenth annual conference companion on Genetic and evolutionary computation*

conference companion, Christian Blum (Ed.). ACM, New York, NY, USA, 1-2. DOI: http://doi.acm.org/10.1145/2464576.2464588.

[R33] S. Biswas, S. Das, S. Kundu, and G. R. Patra. Utilizing time-linkage property in DOPs: An information sharing based Artificial Bee Colony algorithm for tracking multiple optima in uncertain environments. Soft Computing, November, 2013. DOI: http://dx.doi.org/10.1007/s00500-013-1138-z.

[R34] D. Bose, S. Biswas, S. Kundu and S. Das: A Strategy Pool Adaptive Artificial Bee Colony Algorithm for Dynamic Environment through Multi-population Approach. SEMCCO 2012: 611-619. DOI: http://dx.doi.org/10.1007/978-3-642-35380-2_71.

[R35] S. Biswas, D. Bose, S. Kundu: A Clustering Particle Based Artificial Bee Colony Algorithm for Dynamic Environment. SEMCCO 2012: 151-159. DOI: http://dx.doi.org/10.1007/978-3-642-35380-2_19.

[R36] S. Biswas, S. Das, S. Debchoudhury and S. Kundu, "Co-evolving Bee Colonies by Forager Migration: A Multi-swarm based Artificial Bee Colony Algorithm for Global Optimization", *Applied Mathematics and Computation*. In Press.

[R37] S. Biswas, S. Kundu, D. Bose, S. Das, P. N. Suganthan, and B. K. Panigrahi. Migrating forager population in a multi-population Artificial Bee Colony algorithm with modified perturbation schemes. In: Swarm Intelligence Symposium under Proceedings of IEEE Symposium Series on Computational Intelligence, 16-19 April 2013, Singapore, 248 - 255. DOI: http://dx.doi.org/10.1109/SIS.2013.6615186.

[R38] S. N. Omkar, J. Senthilnath, R. Khandelwal, G.N. Naik, and S. Gopalakrishman, "Artificial bee colony (ABC) for multi-objective design optimization of composite structures," *Applied Soft Computing* vol. 11 pp. 489–499, 2011.

[R39] D. Aydın and S. Özyön," Solution to non-convex economic dispatch problem with valve point effects by incremental artificial bee colony with local search," *Applied Soft Computing*, Vol. 13, Issue 5, May 2013, pp. 2456–2466.

[R40] 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.

[R41] W.Y. Szeto, Y. Wu, and S.C. Ho," An artificial bee colony algorithm for the capacitated vehicle routing problem," *European Journal of Operational Research* Vol. 215, pp. 126–135, 2011.

[R42] T-J. Hsieh, H-F. Hsiao and W-C. Yeh," Forecasting stock markets using wavelet transforms and recurrent neural networks: an integrated system based on artificial bee colony algorithm," *Applied Soft Computing* Vol. 11, Issue 2, March 2011, pp. 2510–2525.

[R43] A. R. Yildiz," A new hybrid artificial bee colony algorithm for robust optimal design and manufacturing," Applied Soft Computing. Vol. 13, Issue 5, May 2013, pp. 2906–2912.

[R44] T.-J. Hsieh and W.-C. Yeh, "Knowledge discovery employing grid scheme least squares support vector machines based on orthogonal design bee colony algorithm", *IEEE Trans. on SMC, Part - B*, Vol. 41, No. 5, pp. 1198 – 1212, Oct. 2011.

[R45] D. Bose, S. Kundu, S. Biswas and S. Das: Circular Antenna Array Design Using Novel Perturbation Based Artificial Bee Colony Algorithm. SEMCCO 2012: 459-466. DOI: http://dx.doi.org/10.1007/978-3-642-35380-2_71.

[R46] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. Evolut. Comput.*, vol. 3, no. 2, pp. 82–102, Jul. 1999.

[R47] V. Feoktistov, Differential Evolution in Search of Solutions, Springer, 2006.

[R48] H-G. Beyer and S. Finck, "HappyCat – A simple function class where well-Known direct search algorithms do fail", Parallel Problem Solving from Nature-PPSN XII, *Lecture Notes in Computer Science 7492*, pp. 367-376, 2012.

[R49] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation", *Soft Computing*, Vol. 9, pp. 3–12, 2005.

[R50] S. Das and P. N. Suganthan, "Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems," Jadavpur University & Nanyang Technological University, Dec. 2010.