Black-Box Optimization Benchmarking for Noiseless Function Testbed using Artificial Bee Colony Algorithm

Draft version *

Mohammed El-Abd
American University of Kuwait
melabd@auk.edu.kw

ABSTRACT

This paper benchmarks the artificial bee colony (ABC) algorithm using the noise-free BBOB 2010 testbed.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization—*global optimization, unconstrained optimization*; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems

General Terms

Algorithms

Keywords

Benchmarking, Black-box optimization

1. ARTIFICIAL BEE COLONY

The ABC algorithm was first proposed in [7]. The algorithms was inspired by the method adopted of a swarm of honey bees to locate food sources. There are two different honey bee groups that share knowledge in order to successfully locate such sources. First, there are the *employed bees* that are currently exploiting a food source. Second, there are *unemployed bees* that are continually looking for a food source. Unemployed bees are divided into *scout bees* that search around the nest and *onlookers* that wait at the nest and establish communication with the employed bees.

This algorithm was applied to multidimensional and multimodal function optimization in [7, 2, 11]. The swarm is divided into employed bees, scouts and onlookers. S_n solutions to the problem are randomly initialized in the function domain and referred to as food sources. A number of employed bees, set as the number of the food sources and half the colony size, are used to find new food sources using the following equation:

$$v_{ij} = x_{ij} + \phi_{ij} \times (x_{ij} - x_{kj}), \tag{1}$$

for $j \in \{1 \dots d\}$ where d is the number of dimensions, ϕ_{ij} is a random number uniformly distributed in the range [-1,1], and k is the index of a randomly chosen solution. Both $\mathbf{v}_i$ and $\mathbf{x}_i$ are then compared against each other and the employed bee exploits the better food source.

Onlooker bees next choose a random food source according to the probability given in equation 2. Then, each onlooker bee tries to find a better food source around the selected one using equation 1.

$$p_i = \frac{fit_i}{\sum_{j=1}^{SN} fit_j}, \tag{2}$$

where fit_i is the fitness of the i^{th} food source.

If a food source cannot be improved for a predetermined number of cycles, referred to as *limit*, this food source is abandoned. The employed bee that was exploiting this food source becomes a scout that looks for a new food source by randomly searching the problem domain. The ABC algorithm is shown in Algorithm 1.

Algorithm 1 The ABC algorithm

Require: Max_Cycles, ColonySize, S_r, S_e
Initialize the food sources
Evaluate the food sources
Cycle=1
while $Cycle \le Max\_Cycles$ **do**
    Produce new solutions using employed bees
    Evaluate the new solutions
    Apply Greedy selection process
    Calculate normalized P
    Calculate the fitness
    Produce new solutions for onlooker bees
    Apply Greedy selection process for onlooker bees
    Determine abandoned solutions
    Memorize the best solution
    Cycle = Cycle + 1
end while
return best solution

Previous studies performed to assess the performance ABC included the work in [12] showing that the ABC algorithm performs better than PSO, an evolutionary algorithm (EA) and DE on a small suite of classical benchmark functions. Another study was carried in [10] that compared ABC against PSO, a Genetic Algorithm (GA), DE and an Evolutionary

*Submission deadline: March 25th.

GECCO'10, July 7–11, 2010, Portland Oregon, United States of America.

Table 2: ERT loss ratio (see Figure 3) compared to the respective best result from BBOB-2009 for budgets given in the first column. The last row $\mathrm{RL_{US}/D}$ gives the number of function evaluations in unsuccessful runs divided by dimension. Shown are the smallest, 10%-ile, 25%-ile, 50%-ile, 75%-ile and 90%-ile value (smaller values are better).

	f_1–f_{24} in 5-D, maxFE/D=100011					
#FEs/D	best	10%	25%	**med**	75%	90%
2	4.2	5.2	6.9	11	16	27
10	6.4	9.1	10	14	23	1.4e2
100	6.6	15	19	28	40	1.1e2
1e3	4.8	8.2	35	63	1.4e2	2.5e2
1e4	2.4	12	1.0e2	1.7e2	7.3e2	1.1e3
1e5	2.9	13	2.2e2	8.7e2	4.9e3	7.6e3
$\mathrm{RL_{US}/D}$	1e5	1e5	1e5	1e5	1e5	1e5

	f_1–f_{24} in 20-D, maxFE/D=100002					
#FEs/D	best	10%	25%	**med**	75%	90%
2	2.7	15	34	84	1.1e2	1.1e2
10	12	19	63	4.5e2	5.4e2	5.4e2
100	2.0	21	29	48	1.3e2	5.4e3
1e3	0.32	7.1	54	1.2e2	2.6e2	1.1e3
1e4	0.54	13	1.6e2	3.3e2	1.1e3	2.2e3
1e5	1.3	13	2.8e2	2.1e3	6.3e3	1.3e4
1e6	1.3	24	7.3e2	1.4e4	4.3e4	1.2e5
$\mathrm{RL_{US}/D}$	1e5	1e5	1e5	1e5	1e5	1e5

Strategy (ES) algorithm on a larger number of functions. It was shown that the performance of ABC is better than or at least similar to those algorithms while having a smaller number of parameters to tune. The work in [9] compared ABC to HS and the Bees Algorithm (BA) proposed in [13]. The comparison was based on a small set of classical functions and the ABC showed superior performance over both algorithms while producing reasonable results for higher dimensions.

2. PARAMETER TUNING

For ABC, the work in [1] indicated that the there is no need to have a huge colony size in order to provide good results. We use 40 bees as our previous experiments conducted on the CEC05 benchmarks [14] and repeated using populations of 20, 40 and 100 bees for different problem sizes showed that this setting provided the best results on average. The recommendations in [10] were followed by setting the *limit* parameters to $S_n \times D$, although recent research [1] indicated that lower values might be needed for more difficult functions.

3. RESULTS

Results from experiments according to [4] on the benchmark functions given in [3, 6] are presented in Figures 1, 2 and 3 and in Tables 1 and 2. Experiments use the ABC code available at [8].

4. CPU TIMING EXPERIMENT

For the timing experiment, ABC was run on f8 and restarted until at least 30 seconds had passed (according to Figure 2 in [5]). The experiments have been conducted with an Intel Core 2 Quad 2.4 GHz under Windows Vista using the

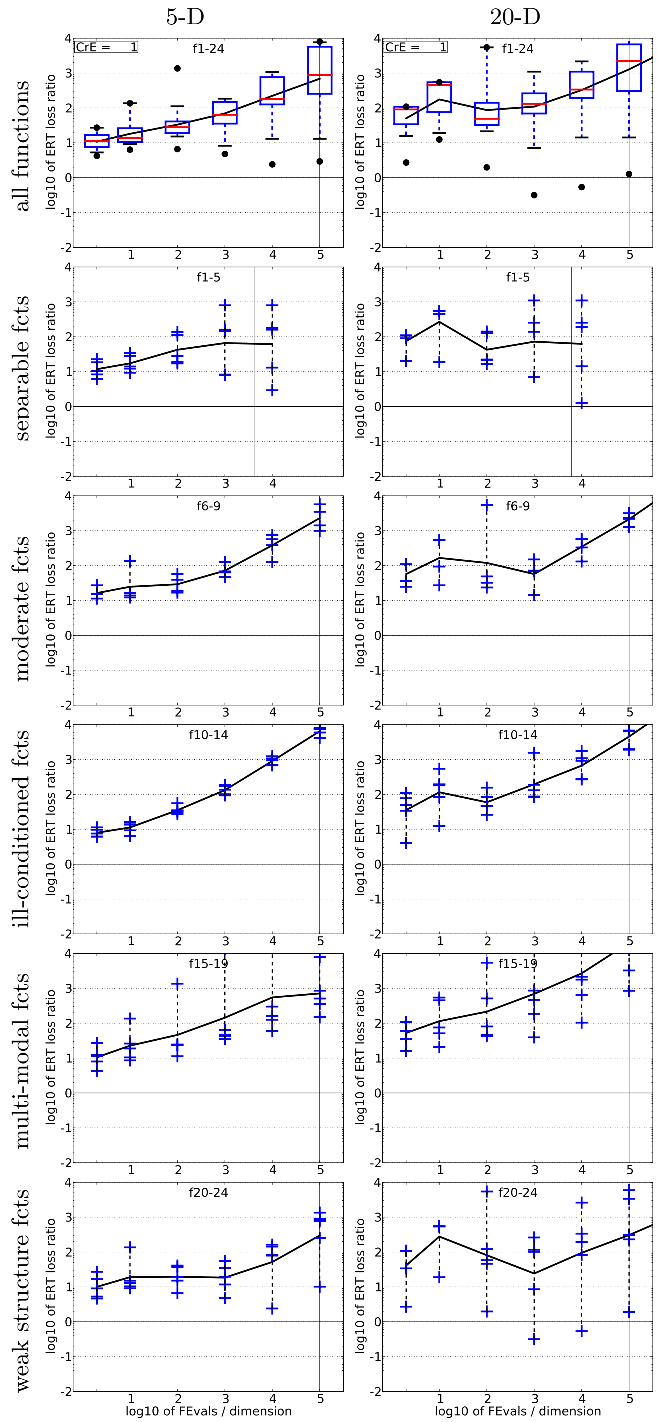

Figure 3: ERT loss ratio versus given budget FEvals. The target value f_t for ERT (see Figure 1) is the smallest (best) recorded function value such that $\mathrm{ERT}(f_t) \leq \mathrm{FEvals}$ for the presented algorithm. Shown is FEvals divided by the respective best $\mathrm{ERT}(f_t)$ from BBOB-2009 for functions f_1–f_{24} in 5-D and 20-D. Each ERT is multiplied by $\exp(\mathrm{CrE})$ correcting for the parameter crafting effort. Line: geometric mean. Box-Whisker error bar: 25-75%-ile with median (box), 10-90%-ile (caps), and minimum and maximum ERT loss ratio (points). The vertical line gives the maximal number of function evaluations in this function subset.

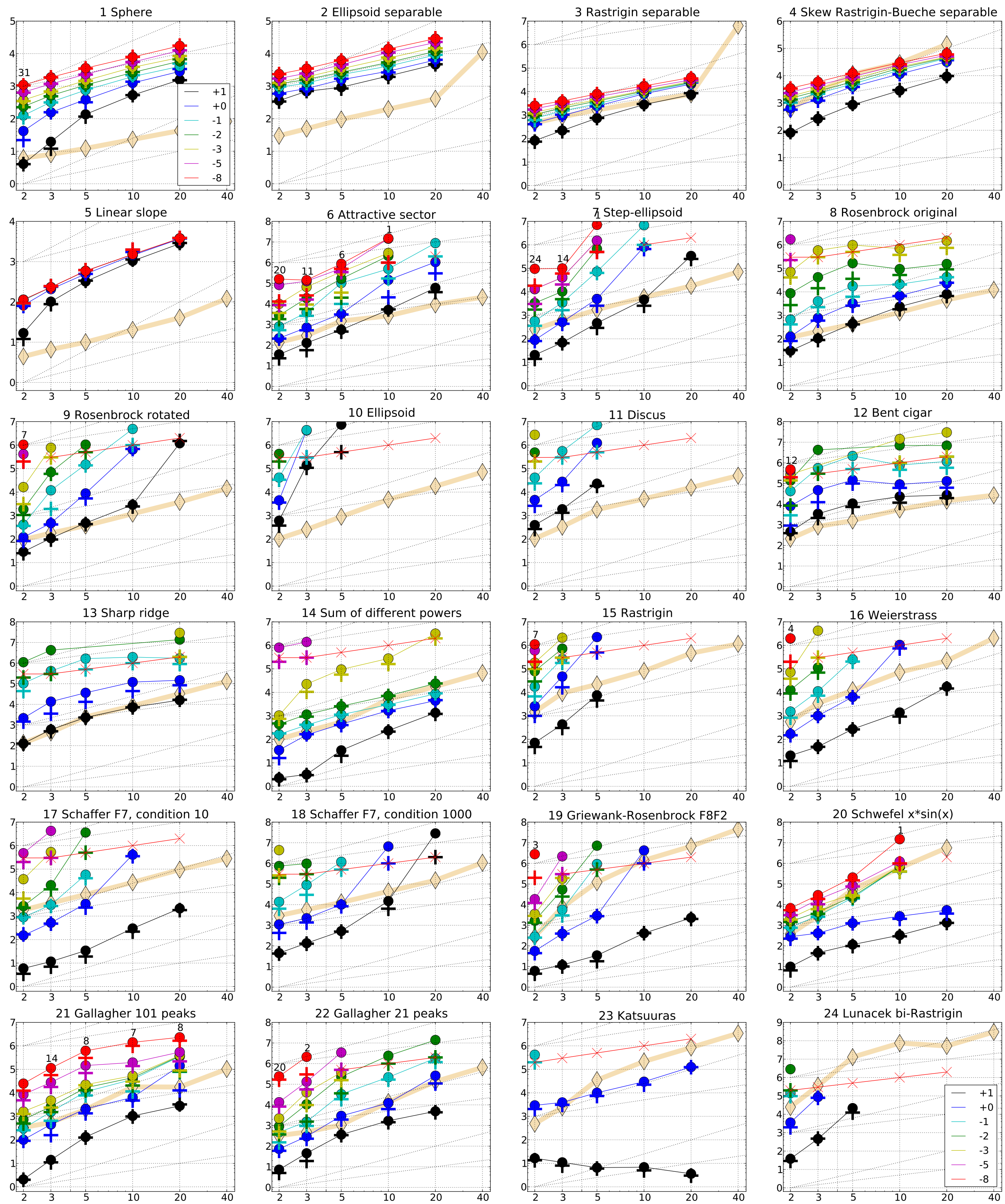

Figure 1: Expected Running Time (ERT, ●) to reach $f_{\mathrm{opt}} + \Delta f$ and median number of f-evaluations from successful trials (+), for $\Delta f = 10^{\{+1,0,-1,-2,-3,-5,-8\}}$ (the exponent is given in the legend of f_1 and f_{24}) versus dimension in log-log presentation. For each function and dimension, $\mathrm{ERT}(\Delta f)$ equals to $\#\mathrm{FEs}(\Delta f)$ divided by the number of successful trials, where a trial is successful if $f_{\mathrm{opt}} + \Delta f$ was surpassed. The $\#\mathrm{FEs}(\Delta f)$ are the total number (sum) of f-evaluations while $f_{\mathrm{opt}} + \Delta f$ was not surpassed in the trial, from all (successful and unsuccessful) trials, and f_{opt} is the optimal function value. Crosses (×) indicate the total number of f-evaluations, $\#\mathrm{FEs}(-\infty)$, divided by the number of trials. Numbers above ERT-symbols indicate the number of successful trials. Y-axis annotations are decimal logarithms. The thick light line with diamonds shows the single best results from BBOB-2009 for $\Delta f = 10^{-8}$. Additional grid lines show linear and quadratic scaling.

Δf	**f_1 in 5-D**, N=15, mFE=3980 #	ERT	10%	90%	$\mathrm{RT_{succ}}$	**f_1 in 20-D**, N=15, mFE=18380 #	ERT	10%	90%	$\mathrm{RT_{succ}}$
10	15	1.4e2	9.0e0	3.1e2	1.4e2	15	1.6e3	6.5e2	2.5e3	1.6e3
1	15	3.9e2	1.7e2	7.5e2	3.9e2	15	2.8e3	1.1e3	4.6e3	2.8e3
1e−1	15	7.6e2	4.9e2	1.1e3	7.6e2	15	4.1e3	1.4e3	6.2e3	4.1e3
1e−3	15	1.5e3	1.2e3	1.8e3	1.5e3	15	7.6e3	3.8e3	9.7e3	7.6e3
1e−5	15	2.3e3	2.2e3	2.6e3	2.3e3	15	1.3e4	1.2e4	1.3e4	1.3e4
1e−8	15	3.6e3	3.4e3	3.9e3	3.6e3	15	1.7e4	1.6e4	1.8e4	1.7e4

Δf	**f_2 in 5-D**, N=15, mFE=7022 #	ERT	10%	90%	$\mathrm{RT_{succ}}$	**f_2 in 20-D**, N=15, mFE=30621 #	ERT	10%	90%	$\mathrm{RT_{succ}}$
10	15	9.3e2	5.7e2	1.5e3	9.3e2	15	4.6e3	2.8e3	5.5e3	4.6e3
1	15	1.6e3	7.4e2	2.3e3	1.6e3	15	6.4e3	5.2e3	8.2e3	6.4e3
1e−1	15	2.3e3	1.6e3	3.4e3	2.3e3	15	9.1e3	7.0e3	1.1e4	9.1e3
1e−3	15	3.4e3	2.0e3	4.4e3	3.4e3	15	1.6e4	1.3e4	1.9e4	1.6e4
1e−5	15	4.6e3	3.7e3	5.5e3	4.6e3	15	2.2e4	2.0e4	2.4e4	2.2e4
1e−8	15	6.3e3	5.8e3	6.8e3	6.3e3	15	2.9e4	2.7e4	3.1e4	2.9e4

Δf	**f_3 in 5-D**, N=15, mFE=9620 #	ERT	10%	90%	$\mathrm{RT_{succ}}$	**f_3 in 20-D**, N=15, mFE=73315 #	ERT	10%	90%	$\mathrm{RT_{succ}}$
10	15	7.4e2	4.1e2	1.2e3	7.4e2	15	7.4e3	3.9e3	1.3e4	7.4e3
1	15	2.4e3	1.1e3	3.3e3	2.4e3	15	2.1e4	7.8e3	2.8e4	2.1e4
1e−1	15	3.0e3	2.0e3	4.2e3	3.0e3	15	2.3e4	9.4e3	3.1e4	2.3e4
1e−3	15	4.5e3	3.7e3	5.3e3	4.5e3	15	2.7e4	1.3e4	3.7e4	2.7e4
1e−5	15	6.0e3	5.3e3	6.9e3	6.0e3	15	3.2e4	2.2e4	3.9e4	3.2e4
1e−8	15	8.0e3	7.1e3	9.4e3	8.0e3	15	4.0e4	3.5e4	4.5e4	4.0e4

Δf	**f_4 in 5-D**, N=15, mFE=21552 #	ERT	10%	90%	$\mathrm{RT_{succ}}$	**f_4 in 20-D**, N=15, mFE=121198 #	ERT	10%	90%	$\mathrm{RT_{succ}}$
10	15	8.6e2	5.8e2	1.3e3	8.6e2	15	9.3e3	3.8e3	1.5e4	9.3e3
1	15	4.0e3	2.2e3	6.0e3	4.0e3	15	3.2e4	1.5e4	4.3e4	3.2e4
1e−1	15	5.0e3	3.2e3	6.5e3	5.0e3	15	4.3e4	1.4e4	4.9e4	4.3e4
1e−3	15	7.8e3	5.8e3	1.3e4	7.8e3	15	4.9e4	1.7e4	8.4e4	4.9e4
1e−5	15	9.2e3	6.9e3	1.4e4	9.2e3	15	5.7e4	3.9e4	1.0e5	5.7e4
1e−8	15	1.2e4	9.5e3	1.6e4	1.2e4	15	6.6e4	5.3e4	1.1e5	6.6e4

Δf	**f_5 in 5-D**, N=15, mFE=980 #	ERT	10%	90%	$\mathrm{RT_{succ}}$	**f_5 in 20-D**, N=15, mFE=5940 #	ERT	10%	90%	$\mathrm{RT_{succ}}$
10	15	3.2e2	1.2e2	4.5e2	3.2e2	15	2.8e3	1.8e3	3.7e3	2.8e3
1	15	4.9e2	2.1e2	7.6e2	4.9e2	15	3.7e3	2.0e3	5.0e3	3.7e3
1e−1	15	5.8e2	2.1e2	8.9e2	5.8e2	15	3.8e3	2.0e3	4.7e3	3.8e3
1e−3	15	5.9e2	2.1e2	9.1e2	5.9e2	15	3.8e3	2.3e3	5.2e3	3.8e3
1e−5	15	5.9e2	2.5e2	9.1e2	5.9e2	15	3.8e3	2.3e3	5.2e3	3.8e3
1e−8	15	5.9e2	2.1e2	9.1e2	5.9e2	15	3.8e3	2.3e3	5.2e3	3.8e3

Δf	**f_6 in 5-D**, N=15, mFE=500058 #	ERT	10%	90%	$\mathrm{RT_{succ}}$	**f_6 in 20-D**, N=15, mFE=2.00e6 #	ERT	10%	90%	$\mathrm{RT_{succ}}$
10	15	5.6e2	2.1e2	9.1e2	5.6e2	15	6.0e4	9.7e3	1.5e5	6.0e4
1	15	3.3e3	1.4e3	6.5e3	3.3e3	11	1.1e6	1.1e5	3.0e6	3.3e5
1e−1	13	1.0e5	5.6e3	5.1e5	2.6e4	3	8.8e6	1.2e6	1.7e7	8.3e5
1e−3	9	3.6e5	1.3e4	1.0e6	2.6e4	0	*54e−2*	*76e−3*	*14e−1*	1.1e6
1e−5	8	5.2e5	2.1e4	1.1e6	7.9e4	.	.	.	.	.
1e−8	6	8.8e5	5.9e4	2.8e6	1.3e5	.	.	.	.	.

Δf	**f_7 in 5-D**, N=15, mFE=500055 #	ERT	10%	90%	$\mathrm{RT_{succ}}$	**f_7 in 20-D**, N=15, mFE=2.00e6 #	ERT	10%	90%	$\mathrm{RT_{succ}}$
10	15	4.7e2	2.3e1	1.4e3	4.7e2	15	3.4e5	5.3e4	9.4e5	3.4e5
1	15	5.0e3	1.1e3	1.0e4	5.0e3	0	*60e−1*	*37e−1*	*83e−1*	5.8e5
1e−1	15	7.3e4	2.7e3	1.5e5	7.3e4	.	.	.	.	.
1e−3	4	1.5e6	2.5e4	4.6e6	1.3e5	.	.	.	.	.
1e−5	4	1.5e6	2.5e4	3.7e6	1.3e5	.	.	.	.	.
1e−8	1	7.1e6	5.9e5	2.0e7	9.0e4	.	.	.	.	.

Δf	**f_8 in 5-D**, N=15, mFE=500057 #	ERT	10%	90%	$\mathrm{RT_{succ}}$	**f_8 in 20-D**, N=15, mFE=2.00e6 #	ERT	10%	90%	$\mathrm{RT_{succ}}$
10	15	4.4e2	1.8e2	8.5e2	4.4e2	15	8.0e3	4.6e3	1.2e4	8.0e3
1	15	3.2e3	9.7e2	6.6e3	3.2e3	15	2.3e4	1.3e4	3.2e4	2.3e4
1e−1	15	1.8e4	3.6e3	1.5e4	1.8e4	15	4.2e4	3.3e4	7.9e4	4.2e4
1e−3	6	9.8e5	4.7e4	2.5e6	2.3e5	11	1.5e6	1.6e5	3.3e6	7.6e5
1e−5	0	*10e−4*	*25e−5*	*18e−3*	1.2e5	0	*24e−5*	*47e−6*	*29e−4*	8.0e5
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f_9 in 5-D**, N=15, mFE=500056 #	ERT	10%	90%	$\mathrm{RT_{succ}}$	**f_9 in 20-D**, N=15, mFE=2.00e6 #	ERT	10%	90%	$\mathrm{RT_{succ}}$
10	15	4.8e2	2.1e2	8.1e2	4.8e2	13	1.2e6	9.5e4	1.9e6	8.9e5
1	15	8.8e3	1.8e3	3.1e4	8.8e3	0	*92e−1*	*67e−1*	*10e+0*	2.0e6
1e−1	15	1.5e5	8.0e3	3.5e5	1.5e5	.	.	.	.	.
1e−3	0	*13e−3*	*18e−4*	*49e−3*	3.1e5	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f_{10} in 5-D**, N=15, mFE=500059 #	ERT	10%	90%	$\mathrm{RT_{succ}}$	**f_{10} in 20-D**, N=15, mFE=2.00e6 #	ERT	10%	90%	$\mathrm{RT_{succ}}$
10	1	7.4e6	9.2e5	1.6e7	4.2e5	0	*50e+2*	*34e+2*	*79e+2*	1.3e6
1	0	*55e+0*	*15e+0*	*17e+1*	2.6e5	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f_{11} in 5-D**, N=15, mFE=500059 #	ERT	10%	90%	$\mathrm{RT_{succ}}$	**f_{11} in 20-D**, N=15, mFE=2.00e6 #	ERT	10%	90%	$\mathrm{RT_{succ}}$
10	15	2.3e4	7.4e2	6.9e4	2.3e4	0	*92e+0*	*65e+0*	*10e+1*	5.2e5
1	5	1.2e6	1.8e5	2.6e6	2.3e5	.	.	.	.	.
1e−1	1	7.1e6	6.2e5	2.2e7	1.2e5	.	.	.	.	.
1e−3	0	*15e−1*	*18e−2*	*28e−1*	1.8e5	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f_{12} in 5-D**, N=15, mFE=500059 #	ERT	10%	90%	$\mathrm{RT_{succ}}$	**f_{12} in 20-D**, N=15, mFE=2.00e6 #	ERT	10%	90%	$\mathrm{RT_{succ}}$
10	15	1.1e4	3.1e3	2.0e4	1.1e4	15	2.8e4	1.5e4	2.6e4	2.8e4
1	14	1.5e5	2.5e4	2.8e5	1.1e5	15	1.3e5	2.6e4	1.7e5	1.3e5
1e−1	3	2.2e6	1.7e5	5.4e6	1.7e5	12	1.2e6	2.3e5	2.5e6	6.8e5
1e−3	0	*29e−2*	*54e−3*	*49e−2*	2.8e5	1	3.0e7	6.0e6	7.4e7	2.0e6
1e−5	.	.	.	.	.	0	*18e−3*	*40e−4*	*23e−2*	1.0e6
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f_{13} in 5-D**, N=15, mFE=500059 #	ERT	10%	90%	$\mathrm{RT_{succ}}$	**f_{13} in 20-D**, N=15, mFE=2.00e6 #	ERT	10%	90%	$\mathrm{RT_{succ}}$
10	15	2.3e3	4.1e2	4.4e3	2.3e3	15	1.6e4	8.6e3	2.3e4	1.6e4
1	15	3.6e4	4.2e3	2.2e5	3.6e4	15	1.5e5	4.5e4	3.7e5	1.5e5
1e−1	4	1.7e6	1.2e5	3.2e6	2.8e5	10	1.7e6	4.5e5	4.5e6	7.2e5
1e−3	0	*15e−2*	*71e−3*	*62e−2*	1.7e5	1	2.9e7	2.2e6	6.9e7	1.2e6
1e−5	.	.	.	.	.	0	*60e−3*	*93e−4*	*15e−2*	1.3e6
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f_{14} in 5-D**, N=15, mFE=500057 #	ERT	10%	90%	$\mathrm{RT_{succ}}$	**f_{14} in 20-D**, N=15, mFE=2.00e6 #	ERT	10%	90%	$\mathrm{RT_{succ}}$
10	15	3.4e1	1.1e1	7.3e1	3.4e1	15	1.3e3	5.0e2	2.8e3	1.3e3
1	15	4.5e2	1.7e2	8.3e2	4.5e2	15	4.4e3	1.5e3	6.9e3	4.4e3
1e−1	15	1.1e3	7.1e2	1.5e3	1.1e3	15	8.6e3	6.6e3	1.1e4	8.6e3
1e−3	15	9.4e4	8.0e3	2.3e5	9.4e4	8	3.1e6	1.0e6	6.0e6	1.4e6
1e−5	0	*27e−5*	*11e−5*	*89e−5*	1.8e5	0	*10e−4*	*82e−5*	*13e−4*	1.8e6
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f_{15} in 5-D**, N=15, mFE=500055 #	ERT	10%	90%	$\mathrm{RT_{succ}}$	**f_{15} in 20-D**, N=15, mFE=2.00e6 #	ERT	10%	90%	$\mathrm{RT_{succ}}$
10	15	7.5e3	1.4e3	7.4e3	7.5e3	0	*84e+0*	*69e+0*	*99e+0*	1.1e6
1	3	2.3e6	1.6e5	4.4e6	2.6e5	.	.	.	.	.
1e−1	0	*11e−1*	*15e−2*	*20e−1*	2.7e5	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f_{16} in 5-D**, N=15, mFE=500057 #	ERT	10%	90%	$\mathrm{RT_{succ}}$	**f_{16} in 20-D**, N=15, mFE=2.00e6 #	ERT	10%	90%	$\mathrm{RT_{succ}}$
10	15	2.7e2	1.7e2	5.0e2	2.7e2	15	1.7e4	1.5e3	4.1e4	1.7e4
1	15	6.4e3	2.8e3	1.0e4	6.4e3	0	*39e−1*	*28e−1*	*45e−1*	1.2e6
1e−1	13	2.5e5	4.3e4	6.9e5	1.8e5	.	.	.	.	.
1e−3	0	*48e−3*	*17e−3*	*11e−2*	2.9e5	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f_{17} in 5-D**, N=15, mFE=500053 #	ERT	10%	90%	$\mathrm{RT_{succ}}$	**f_{17} in 20-D**, N=15, mFE=2.00e6 #	ERT	10%	90%	$\mathrm{RT_{succ}}$
10	15	3.4e1	6.0e0	7.5e1	3.4e1	15	2.1e3	5.6e2	3.6e3	2.1e3
1	15	3.3e3	9.9e2	1.5e4	3.3e3	0	*40e−1*	*27e−1*	*44e−1*	1.2e6
1e−1	15	5.8e4	2.1e4	1.1e5	5.8e4	.	.	.	.	.
1e−3	0	*31e−3*	*10e−3*	*54e−3*	2.6e5	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f_{18} in 5-D**, N=15, mFE=500059 #	ERT	10%	90%	$\mathrm{RT_{succ}}$	**f_{18} in 20-D**, N=15, mFE=2.00e6 #	ERT	10%	90%	$\mathrm{RT_{succ}}$
10	15	5.2e2	1.6e2	1.1e3	5.2e2	1	2.8e7	2.4e6	5.6e7	4.2e5
1	15	1.0e4	2.7e3	1.6e4	1.0e4	0	*13e+0*	*10e+0*	*15e+0*	8.9e5
1e−1	5	1.2e6	1.0e5	2.7e6	1.9e5	.	.	.	.	.
1e−3	0	*14e−2*	*53e−3*	*24e−2*	2.6e5	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f_{19} in 5-D**, N=15, mFE=500057 #	ERT	10%	90%	$\mathrm{RT_{succ}}$	**f_{19} in 20-D**, N=15, mFE=2.00e6 #	ERT	10%	90%	$\mathrm{RT_{succ}}$
10	15	3.4e1	7.0e0	1.1e2	3.4e1	15	2.3e3	1.2e3	3.0e3	2.3e3
1	15	2.9e3	9.5e2	5.0e3	2.9e3	0	*37e−1*	*34e−1*	*40e−1*	1.5e6
1e−1	6	9.3e5	7.5e4	2.5e6	1.8e5	.	.	.	.	.
1e−3	0	*12e−2*	*34e−3*	*20e−2*	3.1e5	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f_{20} in 5-D**, N=15, mFE=466095 #	ERT	10%	90%	$\mathrm{RT_{succ}}$	**f_{20} in 20-D**, N=15, mFE=2.00e6 #	ERT	10%	90%	$\mathrm{RT_{succ}}$
10	15	1.2e2	5.7e1	1.8e2	1.2e2	15	1.3e3	1.0e3	1.8e3	1.3e3
1	15	1.3e3	4.6e2	2.3e3	1.3e3	15	5.4e3	2.4e3	1.0e4	5.4e3
1e−1	15	2.1e4	4.1e3	4.4e4	2.1e4	0	*23e−2*	*18e−2*	*28e−2*	1.7e6
1e−3	15	3.2e4	1.1e4	5.4e4	3.2e4	.	.	.	.	.
1e−5	15	8.4e4	2.3e4	1.6e5	8.4e4	.	.	.	.	.
1e−8	15	2.1e5	4.6e4	4.5e5	2.1e5	.	.	.	.	.

Δf	**f_{21} in 5-D**, N=15, mFE=500057 #	ERT	10%	90%	$\mathrm{RT_{succ}}$	**f_{21} in 20-D**, N=15, mFE=2.00e6 #	ERT	10%	90%	$\mathrm{RT_{succ}}$
10	15	1.3e2	6.3e1	2.2e2	1.3e2	15	2.8e3	1.6e3	4.4e3	2.8e3
1	15	2.1e3	5.8e2	5.5e3	2.1e3	15	1.4e5	4.7e3	6.6e5	1.4e5
1e−1	15	1.1e4	7.3e2	2.7e4	1.1e4	15	3.5e5	1.6e4	8.5e5	3.5e5
1e−3	15	2.2e4	2.7e3	4.8e4	2.2e4	15	3.9e5	4.4e4	9.1e5	3.9e5
1e−5	14	1.5e5	1.5e4	3.8e5	1.1e5	15	5.4e5	1.1e5	1.6e6	5.4e5
1e−8	8	6.2e5	6.2e4	1.3e6	1.8e5	8	2.3e6	2.7e5	6.2e6	5.6e5

Δf	**f_{22} in 5-D**, N=15, mFE=500058 #	ERT	10%	90%	$\mathrm{RT_{succ}}$	**f_{22} in 20-D**, N=15, mFE=2.00e6 #	ERT	10%	90%	$\mathrm{RT_{succ}}$
10	15	3.6e2	9.5e1	6.6e2	3.6e2	15	4.9e3	2.8e3	1.4e4	4.9e3
1	15	2.9e3	2.6e2	1.0e4	2.9e3	15	2.6e5	6.4e3	7.6e5	2.6e5
1e−1	15	3.3e4	1.4e3	6.5e4	3.3e4	10	1.8e6	2.4e5	3.6e6	8.2e5
1e−3	10	3.8e5	3.2e4	9.4e5	1.3e5	0	*44e−3*	*56e−4*	*69e−2*	1.0e6
1e−5	2	3.4e6	9.6e4	8.8e6	1.9e5	.	.	.	.	.
1e−8	0	*45e−5*	*64e−7*	*17e−3*	1.9e5	.	.	.	.	.

Δf	**f_{23} in 5-D**, N=15, mFE=500054 #	ERT	10%	90%	$\mathrm{RT_{succ}}$	**f_{23} in 20-D**, N=15, mFE=2.00e6 #	ERT	10%	90%	$\mathrm{RT_{succ}}$
10	15	6.5e0	2.0e0	1.7e1	6.5e0	15	3.6e0	2.0e0	5.0e0	3.6e0
1	15	1.0e4	2.2e3	1.8e4	1.0e4	15	1.3e5	2.5e4	2.4e5	1.3e5
1e−1	0	*37e−2*	*27e−2*	*42e−2*	1.5e5	0	*64e−2*	*54e−2*	*79e−2*	9.6e5
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	**f_{24} in 5-D**, N=15, mFE=500059 #	ERT	10%	90%	$\mathrm{RT_{succ}}$	**f_{24} in 20-D**, N=15, mFE=2.00e6 #	ERT	10%	90%	$\mathrm{RT_{succ}}$
10	15	2.2e4	9.1e3	6.7e4	2.2e4	0	*12e+1*	*92e+0*	*14e+1*	1.3e6
1	0	*42e−1*	*25e−1*	*56e−1*	2.4e5	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Table 1: Shown are, for a given target difference to the optimal function value Δf: the number of successful trials (#); the expected running time to surpass $f_{\mathrm{opt}}+\Delta f$ (ERT, see Figure 1); the 10%-tile and 90%-tile of the bootstrap distribution of ERT; the average number of function evaluations in successful trials or, if none was successful, as last entry the median number of function evaluations to reach the best function value ($\mathrm{RT_{succ}}$). If $f_{\mathrm{opt}}+\Delta f$ was never reached, figures in *italics* denote the best achieved Δf-value of the median trial and the 10% and 90%-tile trial. Furthermore, N denotes the number of trials, and mFE denotes the maximum of number of function evaluations executed in one trial. See Figure 1 for the names of functions.

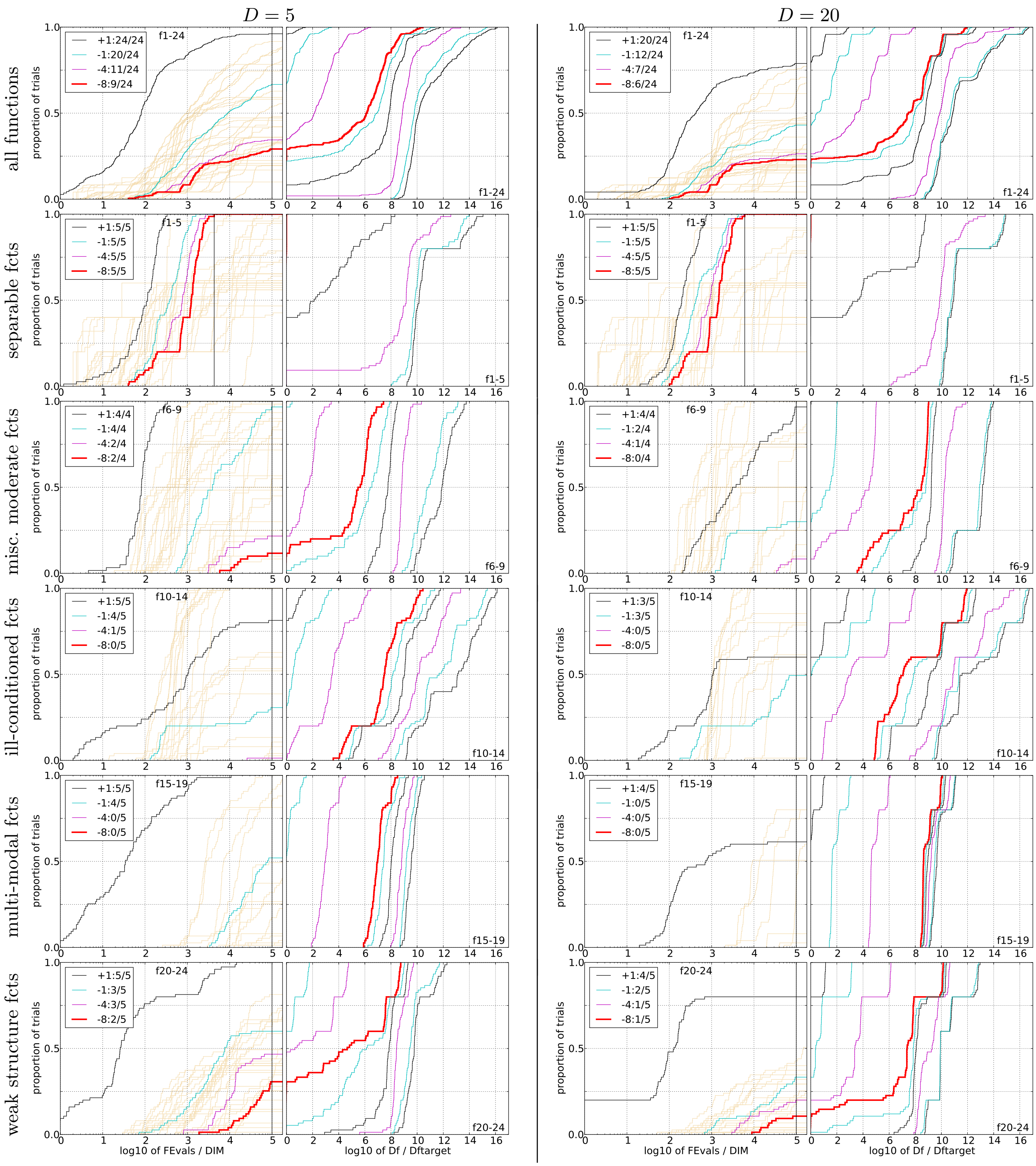

Figure 2: Empirical cumulative distribution functions (ECDFs), plotting the fraction of trials versus running time (left subplots) or versus Δf (right subplots). The thick red line represents the best achieved results. Left subplots: ECDF of the running time (number of function evaluations), divided by search space dimension D, to fall below $f_{\text{opt}} + \Delta f$ with $\Delta f = 10^k$, where k is the first value in the legend. Right subplots: ECDF of the best achieved Δf divided by 10^k (upper left lines in continuation of the left subplot), and best achieved Δf divided by 10^{-8} for running times of $D, 10\,D, 100\,D \ldots$ function evaluations (from right to left cycling black-cyan-magenta). The legends indicate the number of functions that were solved in at least one trial. FEvals denotes number of function evaluations, D and DIM denote search space dimension, and Δf and Df denote the difference to the optimal function value. Light brown lines in the background show ECDFs for target value 10^{-8} of all algorithms benchmarked during BBOB-2009.

MATLAB-code provided. The results were 2.0×10^{-4} seconds per function evaluation in dimensions 2 up to 20. A dependency of CPU time on the search space dimensionality is not visible.

5. REFERENCES

[1] B. Akay and D. Karaboga. Parameter tuning for the artificial bee colony algorithm. In *Proceedings of 1st International Conference on Computational Collective Intelligence*, pages 608–619. Springer-Verlag:Berlin Heidelberg, 2009.

[2] B. Basturk and D. Karaboga. An artificial bee colony (ABC) algorithm for numeric function optimization. In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 66–66. IEEE Press, 2006.

[3] S. Finck, N. Hansen, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2010: Presentation of the noiseless functions. Technical Report 2009/20, Research Center PPE, 2010.

[4] N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking 2010: Experimental setup. Technical Report RR-7215, INRIA, 2010.

[5] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Technical Report RR-6829, INRIA, 2009.

[6] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2010: Noiseless functions definitions. Technical Report RR-6829, INRIA, 2010.

[7] D. Karaboga. An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Engineering Faculty, Computer Engineering Department, Erciyes University, 2005.

[8] D. Karaboga. Artificial bee colony code. http://mf.erciyes.edu.tr/abc/software.htm, 2008.

[9] D. Karaboga and B. Akay. Artificial bee colony (abc), harmony search and bees algorithms on numerical optimization. In *Proceedings of Innovative Produxction Machines and Systems Virtual Conference, IPROMS*, 2009.

[10] D. Karaboga and B. Akay. A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation*, 214:108–132, 2009.

[11] D. Karaboga and B. Basturk. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, 39(3):459–471, 2007.

[12] D. Karaboga and B. Basturk. On the performance of artificial bee colony (ABC) algorithm. *Applied Soft Computing*, 8(1):687–697, 2008.

[13] D. T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, and M. Zaidi. The bees algorithm - a novel tool for complex optimisation problems. In *Proceedings of Innovative Produxction Machines and Systems Virtual Conference, IPROMS*, pages 451–461, 2006.

[14] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Technical Report 2005005, ITT Kanpur, India, 2005.