Memetic Comp. (2011) 3:149–162
DOI 10.1007/s12293-011-0065-8

REGULAR RESEARCH PAPER

Performance evaluation of artificial bee colony optimization and new selection schemes

Konrad Diwold · Andrej Aderhold · Alexander Scheidler · Martin Middendorf

Received: 22 November 2010 / Accepted: 7 July 2011 / Published online: 24 July 2011

Abstract The artificial bee colony optimization (ABC) is a population-based algorithm for function optimization that is inspired by the foraging behavior of bees. The population consists of two types of artificial bees: employed bees (EBs) which scout for new, good solutions and onlooker bees (OBs) that search in the neighborhood of solutions found by the EBs. In this paper we study in detail the influence of ABC's parameters on its optimization behavior. It is also investigated whether the use of OBs is always advantageous. Moreover, we propose two new variants of ABC which use new methods for the position update of the artificial bees. Extensive empirical tests were performed to compare the new variants with the standard ABC and several other metaheuristics on a set of benchmark functions. Our findings show that the ideal parameter values depend on the hardness of the optimization goal and that the standard values suggested in the literature should be applied with care. Moreover, it is shown that in some situations it is advantageous to use OBs but in others it is not. In addition, a potential problem of the ABC is identified, namely that it performs worse on many functions when the optimum is not located at the center of the search space. Finally it is shown that the new ABC variants improve the algorithm's performance and achieve very good performance in comparison to other metaheuristics under standard as well as hard optimization goals.

Keywords Swarm intelligence · Artificial bee colony optimization · Function optimization

K. Diwold (✉) · M. Middendorf
Department of Computer Science,
University of Leipzig, Leipzig, Germany
e-mail: kdiwold@informatik.uni-leipzig.de

M. Middendorf
e-mail: middendorf@informatik.uni-leipzig.de

A. Aderhold
School of Biology, University of St. Andrews,
St. Andrews, Fife, UK
e-mail: aa796@st-andrews.ac.uk

A. Scheidler
IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
e-mail: ascheidler@iridia.ulb.ac.be

1 Introduction

Bio-inspired computation, i.e., the application of biological principles in the context of computation, is as old as computer science itself. Based on biological principles, several prominent computational frameworks such as evolutionary computation and neural networks have been developed (for an extensive review of bio-inspired computation the interested reader should refer to [13]).

A prominent subfield of bio-inspired computation is swarm intelligence [8]. It applies concepts found in the collective behavior of swarms such as fish shoals, bird flocks or social insects to problems in various domains such as robotics or optimization [7]. In optimization, swarm intelligence is probably best known for ant colony optimization [11], which utilizes the concept underlying the pheromone laying behavior of ants; and particle swarm optimization (PSO), which uses group flight guidance for optimization purpose [27].

In recent years bee-inspired algorithms have emerged in the field of swarm intelligence. These algorithms are based on mechanisms underlying the behavior of honeybees and have been successfully applied to various problem domains such as optimization [4], robotics [38], network routing [44], multi-agent systems [31], and protein structure prediction [3].

Bee-inspired algorithms do not have a unified foundation, but are based on different behavioral concepts (see [21] or [10] for an in-depth review on bee-inspired optimization approaches). In general one can distinguish between two classes of algorithms: algorithms based on the mating behavior of honeybees, and algorithms based on their foraging behavior. Furthermore, a recent study [9] suggests that the nest-site selection behavior of honeybees involves principles that are interesting from an optimization point of view.

Mating-inspired algorithms draw their inspiration from the genetic diversity underlying a bee colony. Genetic diversity has been shown to be a driving factor in the ecological success of bees [33] and is due to the polyandrous behavior of a young queen on her maiden flight. Mating inspired algorithms are closely related to evolutionary computation and either introduce new bee-inspired mutation/crossover operators in that context (e.g., [26,37]) or evolve populations by imitating a queen's maiden flight (e.g., [1,32]).

The second kind of algorithms are inspired by foraging behavior. Foraging behavior of honeybees constitutes a decentralized process that works on the basis of decisions of individual bees. It enables a colony to maintain a good ratio of exploitation and exploration of food sources. In addition, foraging is adaptive, meaning that a colony's foraging effort can adapt toward changing needs for resources if necessary [6]. Scouts that successfully locate a resource will return to the hive and promote that resource by means of a waggle dance in order to recruit other bees to forage on that resource [39]. As well as the site's distance and direction, the bee's dance can also encode its quality. Using this mechanism foragers can distribute themselves over the available resources in terms of profitability. A recent study [12] has shown that the recruitment strategies used by honeybees are especially beneficial if resources are of poor quality, few in number, and of variable quality. A number of optimization algorithms have been proposed on the basis of foraging, such as the bees' algorithm (BA) [35], the bee colony optimization algorithm (BCO) [42] and the artificial bee colony optimization algorithm (ABC) [17].

In this paper (which is an extended version of our paper "Artificial Bee Colony Optimization: A New Selection Scheme and Its Performance" at NICSO 2010) the artificial bee colony optimization algorithm (ABC) is studied. ABC was introduced by Karaboga in 2005 [17] and constitutes one of the most prominent approaches in the field of bee-inspired algorithms. The algorithm has been applied to various problem domains including the training of artificial neural networks [19,25], the design of a digital filters [18], solving constrained optimization problems [22], and the prediction of the tertiary structures of proteins [3]. Its optimization performance has been tested and compared to other optimization methods such as Genetic Algorithms (GA), PSO, Particle Swarm Inspired Evolutionary Algorithm (PS–EA), Differential Evolution (DE), and different evolutionary strategies [23,24,20,2].

The ABC algorithm works with a population of artificial bees. The bees are divided into two groups—employed bees (EBs) are responsible for finding and maintaining promising solutions, and onlooker bees (OBs) for performing local search at these solutions. The exploration (via the EBs) and exploitation (via the OBs) is influenced by the qualities of the solutions currently maintained by the EBs. If an EB's solution does not improve over a certain number of steps it will abandon it's current solution and choose a new random solution in the search space (EBs choosing a new random solution are referred to as scouts).

ABC is clearly one of the most applied bee inspired algorithms but several interesting properties have not been studied so far. Therefore, one aim of this paper is to fill this gap and to examine in detail the influence of several ABC key parameters (i.e., size of the bee population, the ratio between the number of employed bees and onlooker bees, and the solution-abandon limit) on the optimization behavior. Some of these parameters have been considered before [2,17,24] and suggestions have been made regarding their parameterization. However, our study shows that is necessary to reconsider these settings because good parameter values depend strongly on the optimization context (e.g., hardness of goal). Moreover, the ratio between onlooker and employed bees is studied, which has not been investigated in detail before. We also test the ABC's optimization performance in scenarios where the global optimum is not located in the center of the search space (which is the typical for applications).

The second aim of this paper is to propose two variants of the standard ABC algorithm that use new methods for the selection of new positions. To show the quality of the new variants of ABC their performance is tested against the standard ABC and several other population-based optimization heuristics on several benchmark functions.

This paper is structured as follows. In Sect. 2 the ABC is described. The new variants of ABC are introduced in Sect. 3. The experimental setup is described in Sect. 4 and the experimental results are presented in Sect. 5. Concluding remarks and an outlook are given in Sect. 6.

2 Artificial bee colony optimization

The ABC algorithm [17] is a population based algorithm for function optimization that can be seen as a minimal honeybee foraging model. The artificial bee population consists of two types of bees: employed bees (EBs) and onlooker bees (OBs). In ABC the search space represents the environment and each point in the search space corresponds to a food source (solution) that the artificial bees can exploit. The quality of a food source is given by the value of the function to be

optimized at the corresponding position. Initially the EBs scout and each EB decides to exploit a food source it has found. The number of EBs thus corresponds to the number of food sources that are currently exploited in the system. EBs communicate their food sources to the OBs. Based on the quality of a food source the OBs decide whether or not to visit it. Good food sources will attract more OBs. Once an OB has chosen a food source it tries to find a better position in its neighborhood by using a local search strategy. If the quality of a new position found by the OB is better than the quality of the position originally communicated by the corresponding EB, the EB will change its position and promote the new food source. Otherwise, the EB remains on its current food source. If the solution of an EB has not been improved for a certain number of steps the EB will abandon the food source and scout for a new one (i.e., it decides for a new food source in search space).

More formally: Given a D dimensional function F and a population of n virtual bees consisting of n_{eb} employed bees and n_{ob} onlooker bees (i.e., $n = n_{eb} + n_{ob}$). Initially and when scouting EB i ($i \in [1 \ldots n_{eb}]$) is placed on a randomly chosen position $p_i = (x_1^i, \ldots, x_D^i)$ in the search space. At the beginning of an iteration each EB i tries to improve its current position by creating a new candidate position p_i^* using the following local search rule

$$p_i^* = (x_1^i, \ldots, x_j^i + rand(-1, 1)(x_j^k - x_j^i), \ldots, x_D^i) \quad (1)$$

where $j \leq D$ is a randomly chosen dimension, $k \neq i$ denotes a randomly chosen EB (called *reference EB*), and $rand(-1, 1)$ is a real valued random number drawn from a uniform distribution between -1 and 1. Note, that only one dimension is changed via Eq. 1. Based on the following greedy selection mechanism, each EB decides whether to discard p_i in favor of p_i^*

$$p_i = \begin{cases} p_i & \text{if } f(p_i) > f(p_i^*) \\ p_i^* & \text{else} \end{cases} \quad (2)$$

where $f(p)$ denotes the fitness at position p such that $f(p) = F(p)$ for maximization problems and $f(p) = U - F(p)$ for minimization problems with a given upper bound U.

After each EB has updated its position, every OB chooses one of the current EB positions by using a standard roulette wheel selection so that the probability P_i of choosing the position p_i of EB i is

$$P_i = \frac{f(p_i)}{\sum_{k=1}^{n_{eb}} f(p_k)}. \quad (3)$$

After an OB has chosen the position of an EB i it tries to find a better position using Eq. 1. In response, the corresponding EB updates its position as described before in case the OB has found a better position. The algorithm monitors the number of times the position of an EB is not improved by local search (either by EB or OB). When this number reaches a limit $l \geq 1$ the EB abandons its position and scouts for a new one. The algorithm stops when a certain stop criterion (e.g., maximum number of iterations, or a good function value has been found) is met. An outline of ABC is given in Algorithm 1.

Algorithm 1 Artificial Bee Colony

```
1: place each employed bee on a random position in the search space
2: while stop criterion not met do
3:   for all employed bees do
4:     if # steps on same position = l then
5:       choose random position in search space
6:     else
7:       try to find better position (according to equations 1 and 2)
8:       if better position found then
9:         move from current position to found position
10:      end if
11:    end if
12:  end for
13:  for all onlooker bees do
14:    choose an employed bee and move to its position (according to
       Equation 3)
15:    try improve position (according to equations 1 and 2)
16:  end for
17: end while
```

For the standard ABC algorithm it was defined that the number of employer bees equals the number of onlooker bees, i.e., $n_{eb} = n_{ob} = n/2$. The algorithm thus depends only on the parameters n and l. In [24] experiments with different population sizes n were performed with the conclusion that, a population size of 50–100 bees can provide reasonable convergence behavior. The parameter l determines how fast solutions are abandoned. In [24] it is argued that $l = n_e \cdot D$ shows better performance than very high or low values of l. In a recent study on ABC parameter tuning [2] it was concluded that for small colony sizes $l = n_e \cdot D$ might not be sufficient, as the algorithm is not able to explore EB solutions enough before they are abandoned. Hence, it is suggested to use higher values of l for small colonies.

3 ABC variants

The variants of ABC that are proposed in this section concern the selection of reference EBs that is done when OBs and EBs generate candidate solutions according to Eq. 1. In the standard ABC algorithm a reference EB is selected randomly with uniform distribution. A potential disadvantage is that the position of the chosen reference EB might not fit well to the current position of the bee. The two modifications of the reference selection rule that are proposed in the following aim to overcome this problem.

ABC$_{\text{gBest}}$. In the ABC$_{\text{gBest}}$ the global best solution found so far is used in addition to the randomly chosen reference EB in order to generate new candidate solutions. Note, that this has some similarity to the functioning of a PSO algorithm where the global best particles influences the position update of the particles [27]. To incorporate the global best solution Eq. 1 is altered as follows

$$p_i^* = (x_1^i, \ldots, x_j^i + rand(-1,1)(x_j^k - x_j^i) + rand(0,1)(x_j^{best} - x_j^i), \ldots, x_D^i) \quad (4)$$

where k refers to the randomly chosen reference EB p_k, $best$ refers to the best position p_{best} found so far, and $j \leq D$ denotes a random dimension. To make sure that the global best term in Eq. 4 always points towards the global best reference $rand(0, 1)$ was used (instead of $rand(-1, 1)$).

ABC$_{\text{gBestDist}}$. Besides including the global best reference in the generation of candidate solutions, in this modification the distance between the current position and a potential reference EB influences the selection probability. Therefore, instead of using the same probability for all reference EBs, an EB (or OB) at position p_i chooses the reference EB $k \in \{1, \ldots, n_{eb}\}$ with $k \neq i$ according to the following probability

$$P_k = \frac{\frac{1}{dist(p_i, p_k)}}{\sum_{j=1, j\neq i}^{n_{eb}} \left(\frac{1}{dist(p_i, p_j)}\right)} \quad (5)$$

where $dist(x, y)$ is the euclidean distance between positions x and y.

After a reference EB has been chosen the new candidate solution is created using Eq. 4. As can be seen, the farther away a potential reference EB is located from the current location, the smaller is the probability to be selected as a reference. The idea of this modification is to prefer near references because for many types of optimization functions it is more reasonable to search between good positions that are close to each other. To give an example: Assume a solution is located close to an optimum. In order to increase the chance to step into the optimum it would be an advantage to select among solutions that are located in the close vicinity. These solutions could provide short step vectors to the optimum. However, this certainly would lead to a decrease in diversity of possible search directions. Furthermore, the distribution of solutions will most likely tend to cluster to areas of interest with solutions stuck to these areas. The results below will elucidate the actual performance of this modification.

4 Experimental setup

The performance of ABC, the two newly proposed ABC modifications, and other reference algorithms has been tested on several standard benchmark problems (see Table 1 for details). The following five algorithms were used as reference algorithms: the PSO algorithm from [43], two forms of the hierarchical PSO (H-PSO and $\bigvee$ H-PSO) from [16], the differential evolution (DE) algorithm from [41] and [30], and the Ant Colony Optimization algorithm for continuous functions (ACO$_R$) from [40]. The parameter values that were used for these algorithms have been adopted from the given references (see Table 2).

Unless stated otherwise all test runs were repeated 100 times. The number of function evaluations that each algorithm required to reach the specified goal—the standard optimization goal (G$_{std}$) and the hard optimization goal (G$_{hrd}$) as given in Table 1—was recorded for each run. To evaluate

Table 1 Test function names and equations (F), domain space range (R), a standard optimization goal (G$_{std}$) that is often used in the literature and a harder optimization goal (G$_{hrd}$)

F		R	G$_{std}$	G$_{hrd}$
Schaffer's F6	$f_{sc}(\mathbf{x}) = 0.5 + \frac{sin^2\left(\sqrt{x_1^2+x_2^2}\right)-0.5}{(1+0.001(x_1^2+x_2^2))^2}$	$[-100; 100]^D$	10^{-5}	10^{-25}
Sphere	$f_{sp}(\mathbf{x}) = \sum_{i=1}^{D} x_i^2$	$[-100; 100]^D$	0.01	10^{-10}
Griewank	$f_{gr}(\mathbf{x}) = \frac{1}{4000}\left(\sum_{i=1}^{D} x_i^2\right) - \prod_{i=1}^{D} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600; 600]^D$	0.1	10^{-9}
Rastrigin	$f_{rg}(\mathbf{x}) = \sum_{i=1}^{D} (x_i^2 - 10cos(2\pi x_i) + 10)$	$[-5.12; 5.12]^D$	100	10^{-7}
Rosenbrock	$f_{rn}(\mathbf{x}) = \sum_{i=1}^{D-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	$[-30; 30]^D$	100	1
Ackley	$f_{ac}(\mathbf{x}) = -20exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - exp\left(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi x_i)\right) + 20 + e$	$[-32; 32]^D$	0.1	10^{-7}

The hard goals were chosen in such a way that a standard ABC (with $n = 100$) will need approximately 10^5 function evaluations to reach them. A dimension of $D = 30$ was used for five of the six the test functions. Schaffer's F6 function constitutes an exception regarding its dimensionality as it is limited to $D = 2$ dimensions

Table 2 Setting of control parameters used in the final experiment

ABC	PSO	H-PSO	⋁ H-PSO	DE	ACO_R
$n = 30$	$n = 40$	$n = 31$	$n = 31$	$n = 50$	$n = 2$
$n_{eb} = 15$	$\omega = 0.6$	$\omega = 0.6$	$\omega = [0.729; 0.4]$	$CR = 0.8$	$k = 50$
$n_{ob} = 15$	$c_1 = 1.7$	$c_1 = 1.7$	$c_1 = 1.7$	$F = 0.5$	$q = 0.1$
$l = D * n_e$	$c_2 = 1.7$	$c_2 = 1.7$	$c_2 = 1.7$		$\epsilon = 0.85$

n is the population size, swarm size, or colony size respectively; n_{eb} is the number of employed bees; n_{ob} is the number of onlooker bees; l is the abandon limit; D is the dimension of problem function; ω is the inertia weight; $c_{(*)}$ is the constriction factors; CR is the crossover rate; F is the scaling factor; k is the archive size; q is the locality of search; ε is the convergence speed

the significance of the observed performance differences the algorithms were pairwise tested against each other using a one-sided Wilcoxon Rank Sum Test with a significance level of $\alpha = 0.05$. In the case of multiple comparisons, p-values from the Wilcoxon Rank Sum Test were corrected using the Bonferroni Step-down (Holm) correction [14]. In addition a Kruskal–Wallis test (factor: Algorithm, $\alpha = 0.05$) was used to test whether the use of different algorithms had an effect on the observed optimization in the given test functions.

5 Results

5.1 Population size

As pointed out in Sect. 2, population size is one of ABC's two control parameters. In recent studies [2,24] the influence of this parameter on the performance of the ABC was investigated. Based on a comparison of the fitness improvement per algorithm step Karaboga and Basturk [24] argue, that an increase of population size up to a certain value increases the algorithm's performance. Their suggestion is to use a population size of 50–100 as it provides acceptable convergence speed and good solutions.

However, there is a problem in using the number of algorithmic steps as a basis for evaluating performance, namely that larger population sizes require more function evaluations per step. For example, an ABC with a population size of 100 needs 10 times as many function evaluations per step as one with a population of 10.

For this reason, we compared different population sizes with respect to the total number of function evaluations. Fig. 1 depicts the median quality of the best found solutions so far over the first $3 \cdot 10^5$ function evaluations for different population sizes $n \in \{10, 30, 60, 100, 120, 140\}$ and all six test functions (Note that the scale on the Schaffer function (x-axis) differs from the rest of the figures as the the global maximum was always reached before $3 \cdot 10^5$ function evaluations). As can be seen, the relative quality differs in different stages of the optimization process. For most test functions (i.e., Griewank, Rosenbrock, Rastrigin, Ackley) very small populations (i.e., $n = 10$) showed fast convergence at the beginning of the optimization process (i.e., in the first 20, 000 evaluation steps). However, larger populations performed better in later stages. Only in the case of the Sphere function did we find that very small populations performed best throughout the whole optimization process, but this is a very simple optimization function. For the more complex functions such as Schaffer, Griewank, Rastrigin, and Ackley, population size 30–60 performed best. Only for the Ackley function a population size of 100 was best for a higher number of function evaluations. Thus, in contrast to [2,24], our results suggest that the best size of an ABC population is problem-dependent and should be tuned accordingly.

5.2 Number of onlooker bees

Concerning the influence of the number of onlooker bees, Table 3 presents the number of function evaluations that were necessary to reach the optimization goals for populations containing 15 and 50 EBs. For each number of EBs, the ABC with a standard number of onlooker bees (i.e., the number of OBs equals the number of EBs) was compared to a modified ABC in which no OBs were used. Each version was tested on the standard and hard optimization goals (see Table 1).

As can be seen, the number of OBs had an influence on performance. Using OBs increased the performance of the algorithm significantly for the standard optimization goal G_{std} in five of six test functions for both numbers of EBs. This was not the case for the hard optimization goal G_{hrd}. For the case of 15 EBs the algorithm containing no OBs performed significantly better for three of the six test functions. Only for the Rastrigin function was the algorithm with OBs able to perform significantly better. For two test functions no significant difference was found. When 50 EBs were used, the algorithm with no OBs performed significantly better in three of the six test cases, whereas the algorithm with OBs performed significantly better for only two test function (Rastrigin and Rosenbrock). For the Schaffer function no statistic difference can be constituted.

Another feature that can be observed is that for small population sizes (i.e., $n = \{15, 30\}$) the standard deviation of the number of function evaluations needed to reach the

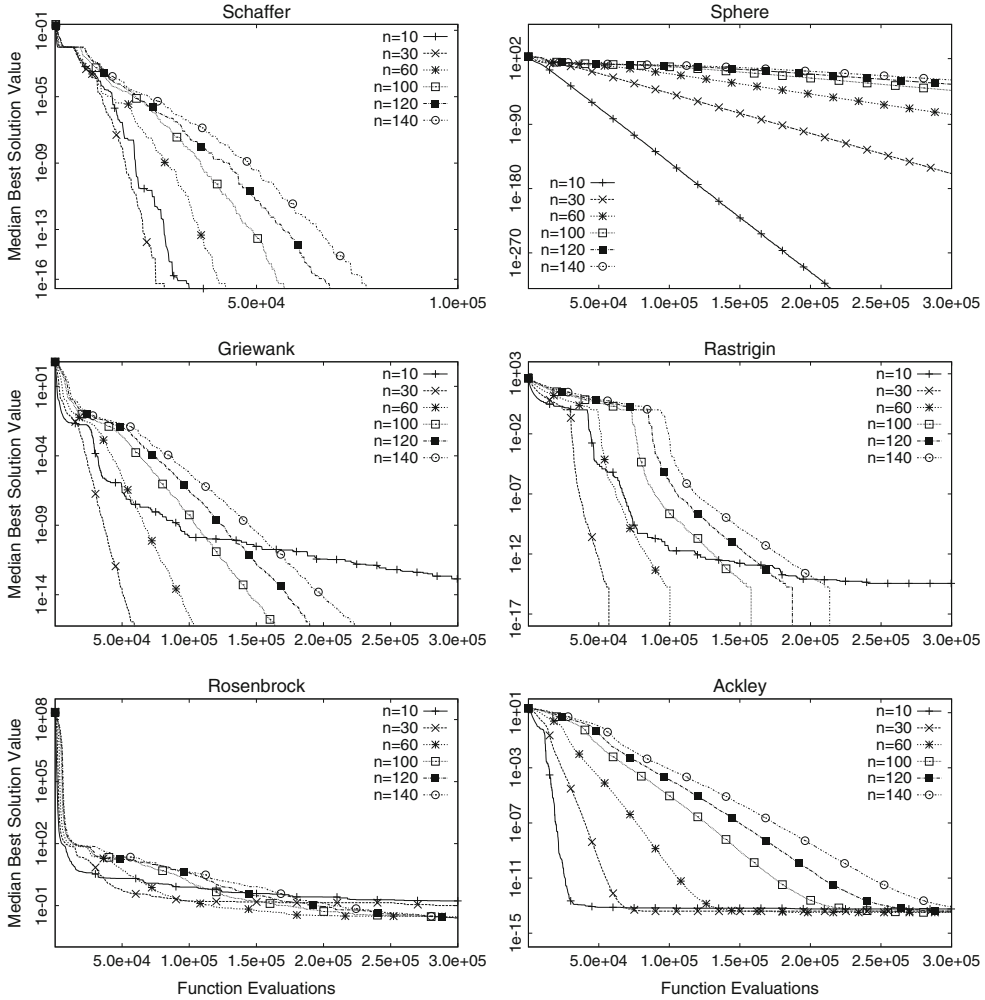

Fig. 1 ABC population size test: comparing improvement of median best solution (y-axis) per iteration (x-axis) for different population sizes n over $3 \cdot 10^5$ function evaluations; Standard ABC settings are used except for l. To avoid very small limit values with small population sizes $l = 100$ if $n_e\ D < 100$

goal seems to depend on the hardness of the goal and the problem domain. While small populations exhibit a small standard deviation in Griewank and Rosenbrock given the standard optimization goal G_{std}, the deviation increases drastically under the hard optimization goal G_{hrd}. This increase is caused by several outliers which exhibit very bad performance by getting stuck in local optima and thus impact mean and standard deviation. That this is not a general trend is reflected by the median which is unaffected by such outliers. Larger populations on the other hand (i.e., $n = \{50, 100\}$) seem to be more resilient in this respect and are able to maintain a low standard deviation, thus not as many outliers, which is also reflected by the fact that median and mean are quite similar. However, their convergence speed is higher than those of small populations. This effect is interesting as it could be interpreted as some sort of size-dependent speed-accuracy trade-off.

These results suggest that the advantage of using OBs in the ABC algorithm is not clear for the hard optimization goal G_{hrd}, while OBs are advantageous for most cases when the standard optimization goal G_{std} is given. This questions the standard rule to set the ratio between the number of OB and EBs to 50:50.

5.3 Limit parameter

As the ABC uses a greedy selection scheme it needs a means to prevent premature convergence to local optima. One feature of the ABC that counteracts premature convergence is its reference selection scheme, as it chooses a reference at random and without taking its fitness or location in the search space into account. Another characteristic that prevents premature convergence is the fact that solutions that did not improve over a certain amount of time-steps will be

Table 3 ABC with different number of employed bees n_{eb} and with or without onlooker bees n_{ob} for the standard optimization goal G_{std} and the hard optimization goal G_{hrd}

(n_{eb}, n_{ob})	Standard goal G_{std}				Hard goal G_{hrd}			
	Median	Mean	Stdv	Sig	Median	Mean	Stdv	Sig
Schaffer								
(15, 0)	13,370	17,525	13, 942	–	27, 201	31, 419	19, 568	–
(15, 15)	11,145	14,820	11, 113	–	24, 050	31, 412	20, 872	–
(50, 0)	18,865	21,644	11, 318	X	61, 368	68, 079	31, 557	–
(50, 50)	20,898	23,599	10, 896	–	66, 451	74, 350	36, 830	–
Sphere								
(15, 0)	13,897	13,915	667	–	28, 387	28, 318	735	X
(15, 15)	7,695	7,810	787	X	32, 977	32, 970	1, 364	–
(50, 0)	43,225	43,345	1, 637	–	89, 650	89, 694	2, 034	X
(50, 50)	15,550	15,632	1, 426	X	104, 750	104, 216	3, 307	–
Griewank								
(15, 0)	14,017	14,085	1, 083	–	36, 810	53, 337	38, 192	X
(15, 15)	10,087	10,404	2, 559	X	39, 952	60, 383	42, 292	–
(50, 0)	42,575	42,739	2, 194	–	100, 950	102, 639	7, 536	X
(50, 50)	21,675	22,326	5, 647	X	105, 800	107, 023	9, 345	–
Rastrigin								
(15, 0)	3,562	3,578	389	–	44, 385	45, 041	4, 599	–
(15, 15)	3,135	3,162	493	X	36, 405	38, 361	8, 145	X
(50, 0)	10,600	10,580	1, 058	–	126, 025	127, 124	8, 583	–
(50, 50)	8,975	8,909	1, 000	X	93, 275	93, 845	9, 329	X
Rosenbrock								
(15, 0)	16,102	16,073	1, 269	–	39, 757	68, 847	84, 683	–
(15, 15)	8,497	10,017	4, 632	X	41, 152	82, 921	136, 730	–
(50, 0)	48,875	48,751	2, 797	–	113, 900	118, 932	25, 340	–
(50, 50)	14,850	18,055	8, 581	X	93, 200	92, 714	29, 518	X
Ackley								
(15, 0)	18,067	18,162	1, 020	–	39, 315	39, 292	1, 198	X
(15, 15)	15,937	15,969	1, 336	X	39, 952	39, 947	1, 153	–
(50, 0)	54,675	54,750	2, 090	–	124, 275	124, 140	2, 087	X
(50, 50)	45,825	45,794	3, 538	X	124, 750	125, 344	2, 409	–

Median, mean and standard deviation (Stdv) of the number of function evaluations needed to reach the goal for the six test functions. The performance of ABC with and without onlooker bees was compared by pairwise testing of populations containing $n_{ob} = n_{eb}$ with those containing $n_{ob} = 0$. In column Sig 'X' denotes a significantly better performance; '–' denotes not significant difference

abandoned in exchange for new random solutions. As outlined in Sect. 2, this behavior is governed by the abandonment limit parameter, l.

The limit parameter has been investigated in previous studies of the ABC (e.g., [2,24]) and it is suggested to make l dependent on the number of EBs n_{eb} and the number of dimensions D of the corresponding problem. The setting of $l = 0.5 \cdot n_{eb} \cdot D$ was suggested ($l = n_{eb} \cdot D$ in the case of multi-modal functions).

We tested the impact of the limit parameter l on the ABC's performance for population sizes $n \in [10, 30, 60, 100, 140]$ using the standard optimization goals G_{Std} and standard ABC settings (see Sect. 4). The values of the limits were in the interval $l \in [10 \ldots 50{,}000]$ with a resolution of $\triangle l = 50$ (please note that up to a limit of $l = 100$ a higher resolution of $\triangle l = 10$ was used). The rather high upper bound of $l = 50{,}000$ was selected to simulate an unlimited solution lifetime (the benchmark functions require on average less than 50,000 function evaluations to reach the optimization goal).

The results depicted in Fig. 2 show that convergence speed decreases drastically for small values of l on all functions. A low value eventually leads to a short solution lifetime. Consequently, new solutions are abandoned after a short

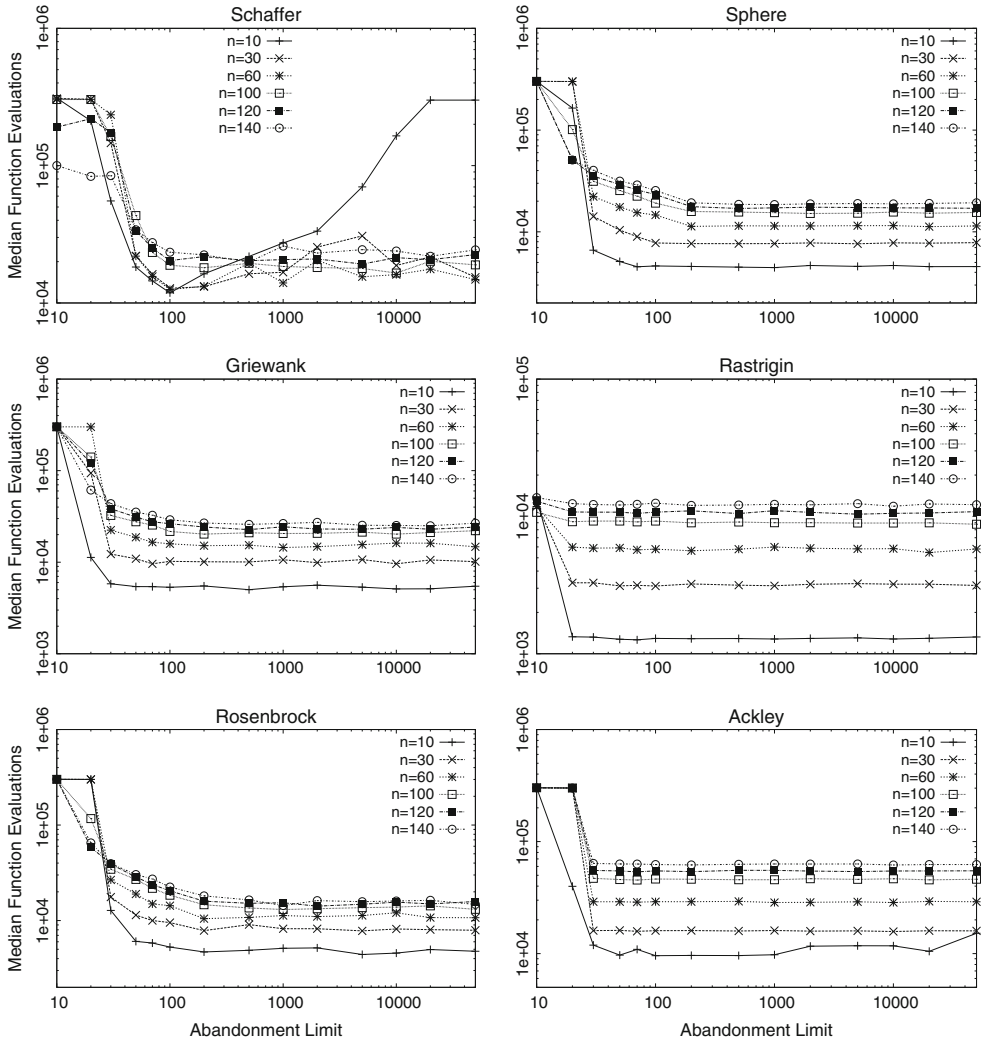

Fig. 2 Test of the ABC with different population sizes n and abandonment limits l. The median number of function evaluations needed to reach the standard optimization goal G_{Std} are shown on the y-axis. Both axes are in logarithmic scale

time period leaving a low chance of improvement. The second observation is that for all functions, except Schaffer, an increased value of the limit does not appear to change convergence speed with this test setting. This does not hold for a very low population size of $n = 10$, which seems to be less robust given very high limit values.

From these results it can be concluded that for the majority of functions and colony sizes, the limit value imposes little influence if it exceeds a certain threshold. The best setting for l observed in this experiment was not in agreement with the standard setting as recommended in [24]. For two functions the default setting was much higher than the best (rather low) found setting. This indicates that the setting of l is likely to depend on the problem domain. The appropriateness of a limit setting derived from the population size and dimension size of the problem function (as defined in [24]) is open to argument. This is especially evident for the Schaffer function with $D = 2$. In this case, a decrease of the total population size would lead to l values below 100 and thus to weak performance (see Fig. 2).

5.4 Non-centered global optimum

It is known that the initial location of candidate solutions and the location of the global optimum relative to the borders of the search space can influence the convergence behavior of an optimization method (see, e.g., [15]). A centered global optimum can provide idealized conditions for optimization, as candidate solutions are often initially distributed uniformly around the center. For this reason it is important to investigate how a method's performance changes if the global optimum is moved away from the center. If convergence speed decreases, this may indicate a problem with the robustness of the method.

Table 4 Shifted test functions, their respective ranges, and the location of their shifted global optima

Function	Range	Function shift
Schaffer's F6	$[-100; 100]^D$	$(80)^D$
Sphere	$[-100; 100]^D$	$(80)^D$
Griewank	$[-600; 600]^D$	$(480)^D$
Rastrigin	$[-5.12; 5.12]^D$	$(4.09)^D$
Rosenbrock	$[-30; 30]^D$	$(24)^D$
Ackley	$[-32; 32]^D$	$(25.6)^D$

In order to test the effect of an uncentered global optimum on the performance of the ABC, each of the used benchmark functions was shifted to the corner of the search space. The distance of the shift was chosen to be 80% of the domain range of the respective function. Hence, a function with domain $[x; y]$ was moved with distance $0.8(x - y)$. The search-space positions of the resulting global optima are listed in Table 4. The function dimensions remained as defined in Table 1 and the standard optimization goal G_{std} was used.

As can be seen in Table 5, the ABC method does not perform as well on functions with moved global optimum as on the same functions with standard (centered) global optimum. These observations indicate that ABC is indeed dependent on the location of the global optimum. Convergence speed is worse if the optimum moves away from the center of the domain space. A possible explanation is the manner in which the algorithm updates positions. As the ABC uses other solutions as references in the location update, it depends on the positions of other solutions to reach certain areas of interest. For a centered global optimum the average location fitness tends to improve towards the center from all directions. Relative to the center, an updating solution can select a reference from the same or the other side of the domain space, allowing for search vectors that have reasonable distance and directions to potentially better positions. In the case that the global optimum is located at a corner of domain space, as in the setting used here, the situation is different. Only one quarter of the function topology around the global optimum is fully calculated in most cases from reference solutions that are located at the same side as the global optimum. This implies a less diverse set of potentially good search directions and distances.

5.5 Comparison of ABC and other algorithms

In this section the performance of the standard ABC and the two proposed variants $\text{ABC}_{gBestDist}$ and $\text{ABC}_{gBestDist}$ are compared with other optimization algorithms.

For each test function, we assessed whether the different algorithms had an effect on optimization performance using a Kruskal–Wallis test. In all cases choice of the algorithm had a significant effect on the observed optimization performance in the respective test domain.

Table 6 shows that the proposed variants of ABC—$\text{ABC}_{gBestDist}$ and $\text{ABC}_{gBestDist}$—significantly improve the performance of ABC on all benchmark functions. $\text{ABC}_{gBestDist}$ is able to enhance the performance of ABC_{gBest} on one test function (i.e., Sphere). In the other cases no significant difference between the two ABC variants could be observed.

The standard ABC algorithm and $\text{ABC}_{gBestDist}$, the best performing ABC variant, were tested against five reference algorithms for the standard optimization goal G_{std}. Fig. 3 depicts boxplots of the number of function evaluations for each algorithm on each test function.

In terms of the necessary number of function evaluations, the proposed ABC variant $\text{ABC}_{gBestDist}$ performs significantly better than all the reference algorithms for two test functions (Ackley and Rosenbrock). For two test functions (Sphere and Schaffer) its performance is on par with the performance of the PSO and the $\bigvee$ H-PSO algorithm, respectively. For the Rastrigin test function $\text{ABC}_{gBestDist}$ shows a comparable performance with the hierarchical PSO variant $\bigvee$ H-PSO and outperforms all the other reference algorithms. For the Griewank test function the hierarchical PSO variant $\bigvee$ H-PSO outperforms all other algorithms signifi-

Table 5 Test with moved global optima of test functions (Funtion$^{\rightarrow}$) for the ABC

Function	Mean	Stdv	Median	Min	Max	Sig
Schaffer$^{\rightarrow}$	18,068	14,831	13,280	2,642	97,418	–
Sphere$^{\rightarrow}$	9,055	1,060	8,880	7,245	14,235	W
Griewank$^{\rightarrow}$	10,364	2,730	9,885	6,525	18,630	–
Rastrigin$^{\rightarrow}$	3,785	654	3,855	2,055	5,385	W
Rosenbrock$^{\rightarrow}$	16,201	7,598	13,920	6,615	33,045	W
Ackley$^{\rightarrow}$	24,843	2,465	24,750	18,735	30,675	W

Median, mean, standard deviation (Stdv), minimum (Min) and maximum (Max) values of function evaluations needed to reach the standard goal G_{std}; In column Sig '*W*' denotes significantly worse performance compared to the same test function without a moved global optima, '–' denotes no significant difference

Table 6 Median, mean and standard deviation (Stdv) of the number of function evaluations required to reach the standard goal G_{std} for ABC, ABC_{gBest}, and $ABC_{gBestDist}$

Function	Method	Median	Mean	Stdv	Significance		
					ABC	ABC_{gBest}	$ABC_{gBestDist}$
Schaffer	ABC	11,145	14,820	11,169		–	–
	ABC_{gBest}	5,715	6,680	4,012	X		–
	$ABC_{gBestDist}$	4,611	6,377	4,686	X	–	
Sphere	ABC	7,695	7,810	791		–	–
	ABC_{gBest}	6,495	6,509	676	X		–
	$ABC_{gBestDist}$	6,150	6,245	698	X	X	
Griewank	ABC	10,087	10,404	2,572		–	–
	ABC_{gBest}	8,872	9,020	1,408	X		–
	$ABC_{gBestDist}$	8,430	8,680	1,499	X	–	
Rastrigin	ABC	3,135	3,162	496		–	–
	ABC_{gBest}	2,527	2,506	270	X		–
	$ABC_{gBestDist}$	2,467	2,466	260	X	–	
Rosenbrock	ABC	8,497	10,017	4,655		–	–
	ABC_{gBest}	5,805	6,682	2,678	X		–
	$ABC_{gBestDist}$	6,225	7,049	2,868	X	–	
Ackley	ABC	15,937	15,969	1,343		–	–
	ABC_{gBest}	10,110	10,118	488	X		–
	$ABC_{gBestDist}$	10,035	10,038	560	X	–	

Population size $n = 30$, number of EBs $n_{eb} = 15$, number of OBs $n_{ob} = 15$. For each test function the significance between each pair of algorithms is shown, 'X' denotes that the algorithm in the corresponding row is significantly better than the algorithm in the corresponding column, '–' denotes no significant difference

cantly, with the ABC, $ABC_{gBestDist}$ and H-PSO scoring the second place.

5.6 Runtime behavior

The results presented in the previous section are based on the standard optimization goals defined in Sect. 4. To test the $ABC_{gBestDist}$'s performance over long runtimes, a test with all benchmark functions and reference algorithms was performed using the hard optimization goals. As the local best PSO (lbest-PSO) showed better convergence speed than the gbest-PSO over long optimization runs in previous studies [16], it was used here instead of the gbest-PSO.

Each algorithm was allowed a maximum of $3 \cdot 10^5$ function evaluations, after which it was terminated. Fig. 4 depicts the evolution of the median best solution value of each algorithm over 50 runs. It can be seen that $ABC_{gBestDist}$ showed the fastest convergence on three out of six functions (Schaffer, Rastrigin, Griewank). It also achieved second fastest convergence on two of the remaining three functions (Sphere, Rosenbrock) and ranked fourth on the Ackley function. This indicates that $ABC_{gBestDist}$ is able to maintain its performance even over longer optimization runs. Its convergence behavior is comparable to state-of-the-art optimization methods even under hard optimization goals.

6 Discussion and outlook

In the first part of this paper we investigated in detail the influence of several parameters (i.e., colony size, onlooker employed ratio and abandon limit) on the performance of the ABC using six standard benchmark test functions.

Regarding the population size, the results suggest that ABC performs better with a smaller population size than usually used in standard ABC setups, given a moderate optimization goal. However, it was also shown that the ideal population size depends on the hardness of the optimization goal. For harder goals, larger populations seem to be advantageous. In contrast to previous studies [2,24], which suggested standard population sizes regardless of problem domain, the current results indicate that the algorithm's performance in terms of population size is problem dependent and thus requires tuning to achieve optimal results.

Whether it is advantageous to use onlooker bees also depends on the optimization goal. For weaker optimization goals using OBs was advantageous for all test functions, but for the harder optimization goals it was in most cases advantageous not to use OBs. This questions the standard division of the ABC population into an equal number of EBs and OBs.

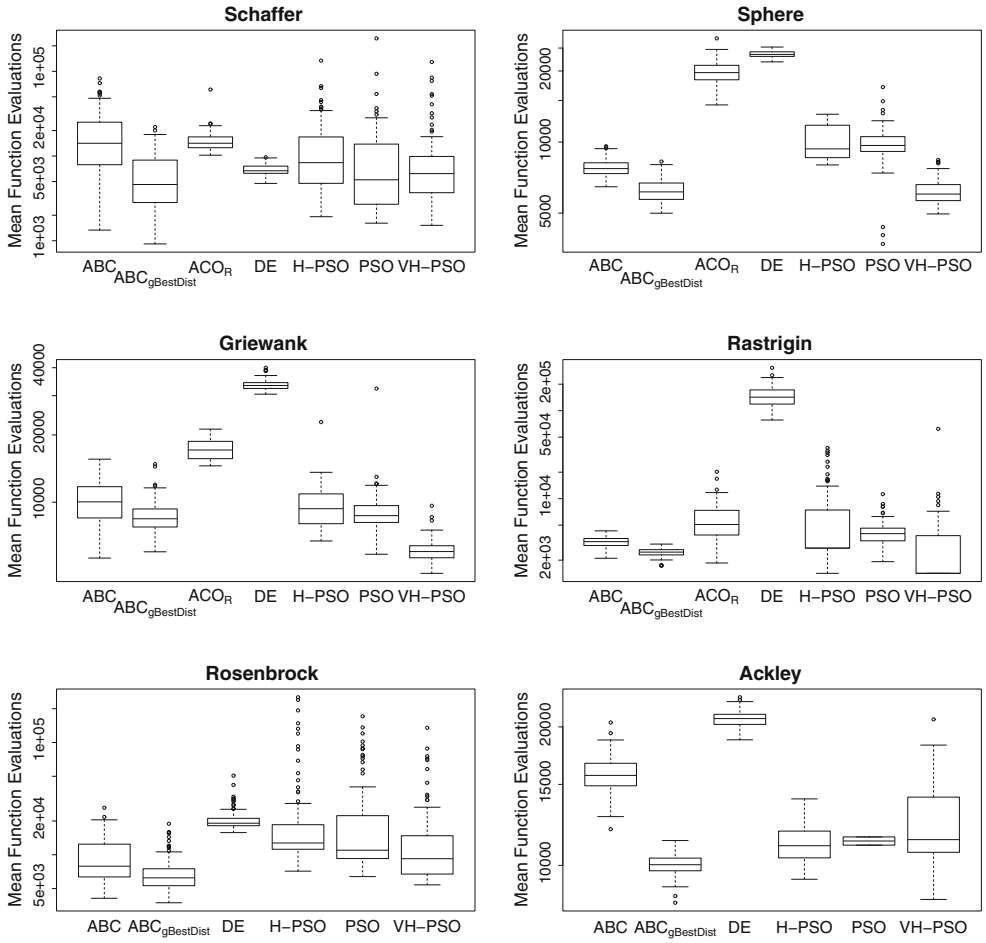

Fig. 3 Boxplots of the number of function evaluations needed to reach the standard optimization goals G_{std} for ABC, $\text{ABC}_{gBestDist}$, PSO, $\bigvee$ H-PSO, H-PSO, DE and ACO_R. Results for ACO_R are omitted when it was not able to reach the optimization goal in $5 \cdot 10^5$ function evaluations

Our results on the abandon-limit parameter suggest that it needs to be set sufficiently high. Given a small limit (i.e., solutions are abandoned very fast), the number of function evaluations required for the algorithm to converge increases. Once the parameter crosses a certain threshold, its setting seems to impose little effect on the algorithm's performance. We also found that the robustness regarding high limit values is influenced by colony size: small colonies showed slow convergence speed under low and high settings of the limit parameter. For the Schaffer function a bad performance was noted for colonies of all sizes under low and high limit value conditions. This indicates that the setting of the abandon-limit parameter might depend strongly on the problem domain.

The ABC was also tested in scenarios where the global optimum was not located in the center of the domain space. This investigation was performed as other optimization algorithms such as the PSO, have previously been found to lack the ability to deal with global optima that are located on the border of the domain space [15]. ABC performs significantly worse in such a scenario compared to a scenario where the global optimum is located in the center of the domain space. This indicates that the performance of the ABC is dependent on the location of the global optimum and constitutes an aspect of the ABC that needs to be improved in future work. In summary the parameter tests outline that the ABC is less robust than previously suggested and that it needs to be carefully tuned regarding the optimization domain in order to achieve optimal results.

As the past has shown many optimization algorithms are able to yield increased performance when extended or hybridized. There are several aspects regarding hybridization and modification that could be investigated in future work on the ABC.

One aspect concerns the local search operator that is used in the ABC. In its current form, the position of a new candidate solution will only differ in one dimension from the position of the employed bee it was generated from.

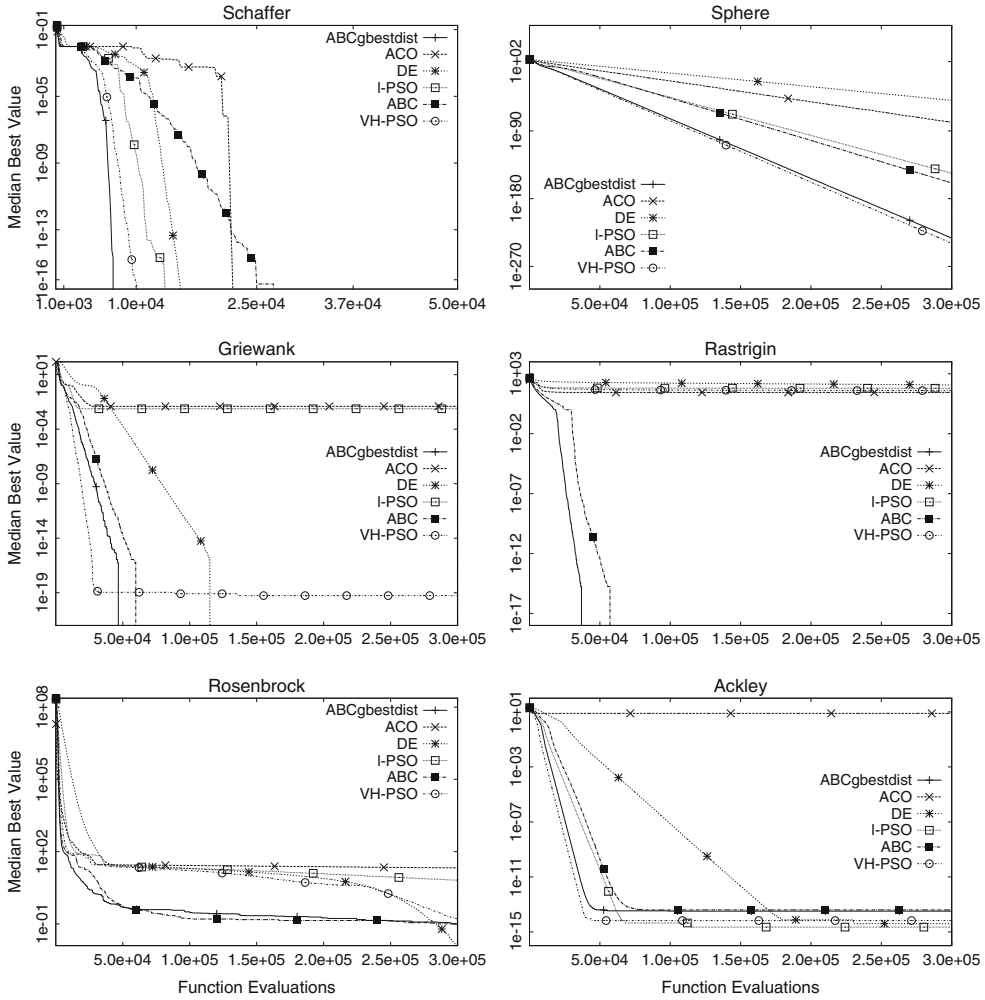

Fig. 4 Evolution of best solution values over 50 runs on six different functions with a maximum of $3 \cdot 10^5$ function evaluations permitted. Best solution value (y-axis) on each iteration was recorded. The number of function evaluations (x-axis) was also recorded at each iteration

Given the "curse of dimensionality" [5] this could impact the convergence of the algorithm in very high dimensional problem domains. A modification of the local search operator might thus be interesting to further improve the performance of the algorithm. This could be done by allowing the modification of more than one dimension (e.g., the approach used in differential evolution [41]). Another possibility would be a hybridization of the ABC and other optimization methods, which could be achieved by substituting the ABC's local search operator. Here, there are many possibilities. One could, for example, consider the use of a particle-swarm-like search, where the number of search steps an employed bee is allowed to do correlates with its relative fitness. Other optimization techniques could also be used to implement the local search, such as simulated annealing [28], the Nelder Mead Simplex [34], Powell's method like search strategies (e.g., [29]) or the recently introduced BOBYQA algorithm [36].

In this study we did not alter the local search operator. Instead, the selection mechanism of the reference positions that influence the location updates of the artificial bees was modified. In its original form the reference selection is random, while the two variants that were proposed here—called ABC_{gBest} and $\text{ABC}_{gBestDist}$—include the global best reference in the generation of candidate solutions as well as taking the distances to reference solutions into account during selection. Both proposed variants performed significantly better than the standard ABC on all six test functions. $\text{ABC}_{gBestDist}$ performed slightly better than ABC_{gBest}. In comparison to other optimization algorithms, $\text{ABC}_{gBestDist}$ was better than, or at least as good as, all tested algorithms on five of the six benchmark functions (in the case of Griewank it was outperformed by $\bigvee$ H-PSO). Under hard optimization goals $\text{ABC}_{gBestDist}$ also showed excellent performance. In terms of convergence speed it performed best in three out of the six benchmark functions, was second in two of the remaining three and third in the last function, which underlines the robust performance of $\text{ABC}_{gBestDist}$.

Acknowledgments We are grateful to Cliodhna Quigley for assistance with the manuscript. This work was supported by the Human Frontier Science Program Research Grant "Optimization in natural systems: ants, bees and slime molds".

References

1. Abbass HA (2001) Marriage in honeybees optimization (MBO): a haplometrosis polygynous swarming approach. IEEE Press, Piscataway, NJ 207–214
2. Akay B, Karaboga D (2009) Parameter tuning for the artificial bee colony algorithm. In: Nguyen N, Kowalczyk R, Chen SM (eds) Proceedings of the ICCCI 2009, LNCS, vol 5796. Springer, Berlin/Heidelberg, Germany, pp 608–619
3. Bahamish HAA, Abdullah R, Salam RA (2009) Protein tertiary structure prediction using artificial bee colony algorithm. In: Proceedings of the Asia International Conference on Modelling & Simulation. IEEE Computer Society, pp 258–263
4. Baykasoglu A, Oezbakr L, Tapkan P (2007) Artificial bee colony algorithm and its application to generalized assignment problem. In: Chan FTS, Tiwari MK (eds) Swarm intelligence: focus on ant and particle swarm optimization. Itech Education and Publishing, Vienna, Austria, pp 113–144
5. Bellman R (1961) Adaptive control processes: a guided tour. Princeton University Press, Princton, NJ
6. Biesmeijer JC, de Vries H (2001) Exploration and exploitation of food sources by social insect colonies: a revision of the scout-recruit concept. Behav Ecol Sociobiol 49(2):89–99
7. Blum C, Merkle D (eds) (2008) Swarm intelligence: introduction and applications. Springer, Berlin/Heidelberg, Germany
8. Bonabeau E, Dorigo M, Theraulaz G (1999) Swarm intelligence: from natural to artificial systems. Oxford University Press, Oxford
9. Diwold K, Beekman M, Middendorf M (2010) Bee nest site selection as an optimization process. In: Proceedings of the 12th international conference on the synthesis and simulation of living systems (Alife XII). The MIT Press, Cambridge, MA, pp 626–633
10. Diwold K, Beekman M, Middendorf M (2010) Honeybee optimisation an overview and a new bee inspired optimisation scheme. In: Hiot LM, Ong YS, Panigrahi BK, Shi Y, Lim MH (eds) Handbook of swarm intelligence, adaptation, learning, and optimization, vol 8. Springer, Berlin/Heidelberg, Germany, pp 295–327
11. Dorigo M, Maniezzo V, Colorni A (1996) The ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern–Part B 26(1):29–41
12. Dornhaus A, Kluegl F, Oechslein C, Puppe F, Chittka L (2006) Benefits of recruitment in honey bees: effects of ecology and colony size in an individual-based model. Behav Ecol 17(3):333–344
13. Forbes N (2004) Imitation of life: how biology is inspiring computing. The MIT Press, Cambridge, MA
14. Holm S (1979) A simple sequentially rejective multiple test procedure. Scand J Stat 6:65–70
15. Huang T, Mohan AS (2005) A hybrid boundary condition for robust particle swarm optimization. IEEE Antennas Wirel Propag Lett 4:112–118
16. Janson S, Middendorf M (2005) A hierarchical particle swarm optimizer and its adaptive variant. IEEE Trans Syst Man Cybern–Part B 35(6):1272–1282
17. Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Tech. rep., Erciyes University, Engineering Faculty
18. Karaboga D (2009) A new design method based on artificial bee colony algorithm for digital IIR filters. J Franklin Inst 346(4):328–348
19. Karaboga D, Akay B (2007) Artificial bee colony (ABC) algorithm on training artificial neural networks. In: IEEE 15th Signal Processing and Communications Applications. IEEE Press, Piscataway, NJ, pp 1–4
20. Karaboga D, Akay B (2009) A comparative study of artificial bee colony algorithm. Appl Math Comput 214(1):108–132
21. Karaboga D, Akay B (2009) A survey: algorithms simulating bee swarm intelligence. Artif Intell Rev 31:61–85
22. Karaboga D, Basturk B (2007) Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. In: Melin P, Castillo O, Aguilar L, Kacprzyk J, Pedrycz W (eds) Foundations of fuzzy logic and soft computing, LNCS, vol 4529. Springer, Berlin/Heidelberg, Germany, pp 789–798
23. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Glob Optim 39(3):459–471
24. Karaboga D, Basturk B (2008) On the performance of artificial bee colony (ABC) algorithm. Appl Soft Comput 8:687–697
25. Karaboga D, Akay B, Ozturk C (2007) Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks. In: Torra V, Narukawa Y, Yoshida Y (eds) Modeling decisions for artificial intelligence, LNCS, vol 4617. Springer, Berlin/Heidelberg, Germany, pp 318–329
26. Karci A (2004) Imitation of bee reproduction as a crossover operator in genetic algorithms. In: Zhang C, Guesgen HW, Yeap WK (eds) PRICAI 2004: trends in artificial intelligence, LNCS, vol 3157. Springer, Berlin/Heidelberg, Germany, pp 1015–1016
27. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks. IEEE Press, Piscataway, NJ, vol 4, pp 1942–1948
28. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
29. Kramer O (2010) Iterated local search with powells method: a memetic algorithm for continuous global optimization. Memet Comput 2:69–83
30. Krink T, Filipic B, Fogel G, Thomsen R (2004) Noisy optimization problems - a particular challenge for differential evolution? In: Proceedings of the Congress on Evolutionary Computation. IEEE Press, Piscataway, NJ, vol 1, pp 332–339
31. Lemmens N, de Jong S, Tuyls K, Nowé A (2008) Bee behaviour in multi-agent systems. In: Tuyls K, Nowe A, Guessoum Z, Kudenko D (eds) Adaptive agents and multi-agent systems III. Adaptation and multi-agent learning, LNCS, vol 4865. Springer Berlin/Heidelberg, Germany, pp 145–156
32. Marinakis Y, Marinaki M, Matsatsinis N (2010) A bumble bees mating optimization algorithm for global unconstrained optimization problems. In: González J, Pelta D, Cruz C, Terrazas G, Krasnogor N (eds) Nature inspired cooperative strategies for optimization (NICSO 2010), studies in computational intelligence, vol 284. Springer, Berlin/Heidelberg, Germany, pp 305–318
33. Mattila HR, Seeley TD (2007) Genetic diversity in honey bee colonies enhances productivity and fitness. Science 317:362–364
34. Nelder JA, Mead R (1965) A simplex method for function minimization. Comput J 7(4):308–313
35. Pham D, Ghanbarzadeh A, Koc E, Otri S, Rahim S, MZaidi (2006) The bees algorithm a novel tool for complex optimisation problems. In: Proceedings of IPROMS 2006 conference, pp 454–461
36. Powell M (2009) The bobyqa algorithm for bound constrained optimization without derivatives. Tech. Rep. NA2009/06, Department of Applied Mathematics and Theoretical Physics, University of Cambridge
37. Sato T, Hagiwara M (1997) Bee system: finding solution by a concentrated search. In: IEEE International Conference on Systems, Man, and Cybernetics, 1997. 'Computational Cybernetics and Simulation'. IEEE Press, Piscataway, NJ, vol 4, pp 3954–3959

38. Schmickl T, Crailsheim K (2008) Trophallaxis within a robotic swarm: bio-inspired communication among robots in a swarm. Auton Robots 25:171–188
39. Seeley TD (2010) Honeybee democracy. Princeton University Press, Princeton, NJ
40. Socha K, Dorigo M (2008) Ant colony optimization for continuous domains. Eur J Oper Res 185(3):1155–1173
41. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11:341–359
42. Teodorovic D, Lucic P, Markovic G, Orco MD (2006) Bee colony optimization: Principles and applications. In: 8th seminar on neural network applications in electrical engineering, NEUREL 2006. IEEE Press, Piscataway, NJ, pp 151–156
43. Trelea IC (2003) The particle swarm optimization algorithm: convergence analysis and parameter selection. Inf Process Lett 85:317–325
44. Wedde HF, Farooq M (2005) Handbook of Bioinspired Algorithms and Applications, Chapmann & Hall, chap New ideas for developing routing algorithms inspired by honey bee behavior, pp 321–339