Information Sciences 291 (2015) 115–127

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

On clarifying misconceptions when comparing variants of the Artificial Bee Colony Algorithm by offering a new implementation

Marjan Mernik[a,*], Shih-Hsi Liu[b], Dervis Karaboga[c], Matej Črepinšek[a]

[a] University of Maribor, Faculty of Electrical Engineering and Computer Science, Smetanova 17, 2000 Maribor, Slovenia
[b] California State University, Fresno, Department of Computer Science, 2576 E San Ramon Dr, Fresno, CA 93740, USA
[c] Erciyes University, Department of Computer Engineering, 38039 Melikgazi, Kayseri, Turkey

ARTICLE INFO

Article history:
Received 20 December 2013
Received in revised form 21 July 2014
Accepted 15 August 2014
Available online 28 August 2014

Keywords:
Swarm intelligence
Artificial Bee Colony Algorithm
Experiment replication

ABSTRACT

Artificial Bee Colony (ABC) is a Swarm Intelligence algorithm that has obtained meta-heuristic researchers' attention and favor over recent years. It comprises good balance between exploitation (employed bee phase and onlooker bee phase) and exploration (scout bee phase). As nowadays, more researchers are using ABC and its variants as a control group to perform comparisons, it is crucial that comparisons with other algorithms are fair. This paper points to some misapprehensions when comparing meta-heuristic algorithms based on iterations (generations or cycles) with special emphasis on ABC. We hope that through our findings this paper can be treated as a beacon to remind researchers to learn from these mistakes.

1. Introduction

A common recommendation for fair comparison amongst meta-heuristic algorithms [12] (e.g., Evolutionary Algorithms [6,26]) for artificially constructed benchmark functions is to compare algorithms based on an equal number of consumed fitness evaluations [9,25]. However, in experiments with meta-heuristic algorithms researchers usually employ a stopping condition based on the maximum number of iterations (generations or cycles), and not the maximum number of fitness evaluations. In order to overcome this problem, using the same maximum number of fitness evaluations for all algorithms whilst still using the original implementation of algorithms based on iterations, researchers usually use the following formula: $MCN = FEs/NP$, where MCN is the maximum number of iterations (generations or cycles), FEs is the maximum number of fitness evaluations, and NP is the size of a population. However, the proposed formula would hold only if there are no extra searches in meta-heuristic algorithm. It is quite common nowadays that meta-heuristic algorithms are blended with local searches (e.g., Memetic Algorithm [71]) or incorporate other algorithms (e.g., ensembles). Hence, applying the aforementioned formula needs to be done after careful consideration of this issue.

Artificial Bee Colony (ABC) [3,48–50] is an interesting Swarm Intelligence (SI) algorithm with good exploration and exploitation abilities [20,24]. It has been applied to numerous practical problems, as well as to artificially created optimization functions. For a detailed list of ABC usage please see recent comprehensive surveys on ABC [8,13,52,106], where it can be noticed that ABC is more popular than other bee algorithms (e.g., Bee Colony Optimization, Bee Algorithm) [51].

* Corresponding author.
E-mail addresses: marjan.mernik@uni-mb.si (M. Mernik), shliu@CSUFresno.edu (S.-H. Liu), karaboga@erciyes.edu.tr (D. Karaboga), matej.crepinsek@uni-mb.si (M. Črepinšek).

http://dx.doi.org/10.1016/j.ins.2014.08.040
0020-0255/© 2014 Elsevier Inc. All rights reserved.

Interestingly, the scout bee phase in the basic implementation of ABC [3,48–50] uses an extra fitness evaluation per iteration when a certain condition holds (namely, when a solution cannot be improved after a predetermined number of trials controlled by 'limit' parameter). Hence, the formula $MCN = FEs/NP$ cannot be used anymore when applying basic ABC. This fact has been overlooked by many researchers, despite that in ABC publicly available code it can be clearly seen that fitness is consumed during the scout bee phase due to the calling function *init*() used during the initialization phase (it is also obvious that each new random solution needs to be evaluated):

```
void init(int index)
begin
    ...
    fitness[index] = CalculateFitness(f[index]);
    trial[index] = 0;
end
void SendScoutBees()
begin
    ...
    if (trial[maxtrialindex] >= limit) then
        init(maxtrialindex);
    end
end
```

On the other hand, it should be noted that during the initialization phase of the basic ABC algorithm, the number of fitness evaluations is half of *NP*. Paper [3] gave an approximation of the number of consumed fitness evaluations of basic ABC. The ABC inventors wrote in [3]: *"Since the number of onlookers is equal to SN, in each cycle* $2 * SN$ *searches are conducted by employed bees and onlooker bees. Hence, when the maximum cycle number (MCN) is reached, totally,* $2 * SN * MCN$ *searches are carried out. So, the search complexity of ABC is proportional to* $2 * SN * MCN$.*"* Note that in ABC the colony size *NP* is $2 * SN$ (*SN* employed bees + *SN* onlooker bees). Clearly, this formula is only an approximation and cannot be directly used in comparison with other meta-heuristic algorithms. It seems that many researchers simply neglected the word 'proportional' in the aforementioned statement and literally took only the formula $FEs = 2 * SN * MCN$. In the basic ABC implementation, compared to some other algorithms (e.g., DE [91]), exact static determination of the number of fitness evaluations cannot be defined based on *NP* and *MCN* in advance (scout bee can be or cannot be employed in a particular iteration, that depends on the value of the control parameter 'limit'). In this paper, the upper-bound of the fitness evaluations in basic ABC is defined and given as: $FEs <= (SN + SN + 1) * MCN + SN$. In each iteration there are *SN* employed bees, *SN* onlooker bees, and at most 1 scout bee. During the initialization phase, additional *SN* solutions are created and evaluated. Of course, using an approximation (even an upper bound) for the number of fitness evaluations whilst comparing different meta-heuristic algorithms is often not a good and satisfactory option, as there will always be some threats to the validity to our conclusions. However, the problem can be easily fixed by slightly changing ABC implementation (see Algorithms 1 and 2 in Appendix A):

- setting the stopping condition with the maximum number of fitness evaluations and by counting fitness evaluations dynamically (during ABC run) in the function *calculateFitness*() used in all ABC phases (initialization, employed bee phase, onlooker bee phase, and scout bee phase), and
- achieving an employed bee that will become a scout bee does not consume a fitness evaluation in the employed bee phase as the solution is going to be replaced by a random solution anyway. In such an implementation there will be exactly $2 * SN$ fitness evaluations per iteration (generation or cycle).

As more and more researchers are using ABC variants nowadays it is crucial that comparison with other algorithms is fair. Omitted counting of fitness evaluations during the scout bee phase is an omnipresent problem of many ABC variants. Due to this fact, many conclusions that resulted from those experiments might have serious threats to validity, and some of them might now be invalid, particularly when extra fitness evaluations are consumed in the local searchers combined with the ABC algorithm. Note that nothing can be generalized, and comparisons need to be re-visited case by case. The main objectives of this paper are:

1. Offering alternative implementations of ABC to fix the total fitness evaluations to a constant number:
   (a) definition of the stopping condition based on counting each fitness evaluation during all phases;
   (b) re-implementation of ABC where the total fitness evaluation number is defined in terms of *MCN* and *SN*, but under the condition that the number of fitness evaluations consumed is not bigger than allowed.
2. Studying more than 100 papers on ABC (variants), whilst checking:
   (a) if ABC (variant) algorithm/pseudo-code/flowchart/description clearly indicates that fitness evaluations have been properly counted during the scout bee phase;
   (b) if comparison of ABC (variant) with other algorithms was based on equal number of fitness evaluations consumed;

(c) if convergence graphs comparing ABC (variant) with other algorithms were based on equal number of fitness evaluations consumed; and

3. Investigate if the conclusions from the experiments in [49] still hold if the experiments are replicated taking into account the equal number of fitness evaluations for both offered implementations.

The paper is organized as follows. Section 2 shows the investigation of ABC and its variants along with our findings, followed by replications of some experiments of ABC in Section 3. The concluding remarks are presented in Section 4. The ABC pseudo-codes of two new implementations are presented in Appendix A. Both Algorithms ensure that no more than the allowed maximum number of fitness evaluations are consumed.

2. Investigation of ABC usage in some existing papers

In order to have a good overview of existing ABC usage, we performed forward snowballing [56] which is a process during systematic mapping studies [78]. By forward snowballing, those papers that cited the original ABC journal paper [49] were searched for, studied and classified. Overall, paper [49] has been for a long period of time the most cited paper at the Applied Soft Computing journal. The Scopus search on August 22, 2013 returned 567 hits for papers published in the period January 2009–August 2013. However, only 108 papers were electronically available to us. Among those, 89 papers [1,2,4,5,7,11,14–17,19,22,27–32,34–47,53–55,58–60,62–70,73–77,79,80,82–90,92–102,104,105,107–119] were relevant for our study (ABC or its variant were explained in the papers or used for comparisons), whilst the original ABC journal paper [49] was only mentioned in the remaining 19 papers in related work or future work. We are aware of the fact that we did not study all ABC papers and that latter conclusions might have some threats to validity due to this fact. To strengthen the validity of our conclusions a confidence interval (margin of error) [18] has been added with a confidence level of 95%. The confidence interval within our study was 8.5. The following research questions have been asked:

1. How many papers clearly indicated in ABC pseudo-code/flowchart/description that fitness evaluations have been consumed in the scout bee phase?
2. How many papers used the condition of equal number of fitness evaluations consumed for ABC comparison?
3. How many papers presented convergence graphs comparing ABC with other algorithms based on equal number of fitness evaluations consumed?

After reading and studying those 89 papers the results were quite surprising and exemplified how omnipresent the problem discussed in Section 1 is:

- Although in ABC publicly available code (http://mf.erciyes.edu.tr/abc) there was clear indication that fitness evaluations have been consumed during scout bee phase, only a few authors (e.g., [68,82,95,108,113]) also indicated it in their work. There were 11 such studies (12.4%), whilst in 18 studies (20.2%) the answer was indecisive, as there were insufficient details in those studies. In the remaining 60 studies (67.4%) there were no indication that fitness evaluations had been counted in the scout bee phase. Of course, there is a chance that only the ABC description was not precise enough, but that their implementation fixed this problem. By using the confidence interval it can be concluded using the 95% confidence level that $67.4 \pm 8.5\%$ of the examined work did not have properly counted fitness evaluations.
- Comparing meta-heuristic algorithms based on an equal number of iterations is only suitable if all the compared algorithms consumed an equal number of fitness evaluations in each iteration (see discussion in Section 1). Indeed, 36 studies out of 89 (40.4%) had used inappropriate comparisons based on equal numbers of iterations, whilst 28 studies (31.5%) had compared algorithms based on an equal number of fitness evaluations or on CPU time. In the remaining 25 studies (28.1%), there were insufficient details to find out how the comparisons were done. In the latter case, it seems that in many studies only the best solution was of interest regardless of the computational costs. For solving hard practical problems, where results in real time are not needed, this might be a feasible path. Note that despite the fact that more than 30% of the studies tried to fairly compare algorithms based on equal number of fitness evaluations this does not mean that the comparisons were indeed fair since there might be some cases where fitness evaluations were uncounted (e.g., in the scout bee phase or in the additional local search phase). By using the confidence interval it can be concluded using the 95% confidence level that $40.4 \pm 8.5\%$ of the examined work were using inappropriate comparisons based on equal numbers of iterations.
- It should be clear by now that comparing on an equal number of iterations is mostly inappropriate for meta-heuristic algorithms (only in special cases). The same is true for convergence graphs which should not be based on an equal number of iterations if compared algorithms do not consume an equal number of fitness evaluations in each iteration. While reading 89 studies on ABC we often found claims about new ABC versions with the property of extremely fast convergence. But, was the convergence properly measured based on fitness evaluations or on CPU time? Out of 89 studies only 44 included discussions about convergence graphs, and 28 studies out of 44 (63.6%) were still using convergence graphs comparing ABC and other algorithms based on an equal number of iterations, whilst only 16 studies out of 44 (36.4%) were using convergence graphs based on the number of fitness evaluations or on CPU time. By using the confidence interval it can be stated using the 95% confidence level that $63.6 \pm 8.5\%$ of the examined work did not properly interpret convergence graphs.

After the quantitative analysis it is informative to look into some interesting details of particular studies. It is often the case that researchers are convinced that their design and execution of the experiment is correct. From this study we did not form an impression that wrong design was used deliberately. Researchers are simply truly unaware of this problem. It is quite common that researchers simply believe that comparison on an equal number of iterations is fair, and they express this belief in their papers. Some examples using ABC are:

- *"To make a fair comparison, the maximum generation number and the searching range of the parameters are the same for all algorithms. Namely, the maximum generation number is set as 100, and the population size is set as 120* [102]*."* But, this is far from being fair in this particular case [102]. In each iteration the compared algorithms in [102] (e.g., HTCMIABC, CLONALG and ABC) consumed different number of fitness evaluations per individual (see Fig. 2 in [102]). Based on such experiment the authors [102] concluded: *"It is obvious from Fig. 3* [102] *that the convergence speed of the HTCMIABC is very quickly and all the parameters a, b and c converge to the matching true values rapidly (less than 30 generations). So the HTCMIABC is very efficient for parameter identification of a chaotic system* [102]*."* The conclusion might be valid but a wrongly designed experiment cannot support or prove such a conclusion.
- *"For the parameters of ABC, let the iteration number is 500 for all the functions below for ABC while the single QPSOs iterations are 1000. Let ABC and QPSO run 100 times independently with 50 employed bees and 50 particles, respectively [28]."* In this particular case the number of fitness evaluations for QPSO is: $FEs(QPSO) = num\_particles * num\_iterations = 50 * 1000 =$ 50,000, whilst using formula from [3] giving also $FEs(\mathrm{ABC}) = (2 * employed\_bees) * num\_iterations = (2 * 50) * 500 =$ 50,000. Unfortunately, the formula from [3] are only an approximation. In spite of a faithful intention, a fair comparison was unachieved in [28].
- *"The iterations count is no longer a reasonable measure as different computational complexity maybe taken in each iteration for different algorithms. In order to compare the different algorithms, a fair measure method should be selected. In this paper, we use number of function evaluations (FEs) as a measure criterion. All algorithms were terminated after 100,000 FEs* [110]*."* The experiment would be correct if all the fitness evaluations had been properly counted. Unfortunately, there is no indication that fitness evaluations had been counted during the scout bee phase and in the newly-suggested crossover operator of Hybrid Artificial Bee Colony (HABC) in [110].

The original ABC used at most one scout bee per iteration [49]. The problem of inproperly counting fitness evaluations in the scout bee phase is extravagated in the sense that more scout bees are used per iteration. Indeed, there were many suggestions for ABC improvement that includes more exploration of the search space using extra scout bees. Some of the included studies that used more than one scout bee per iteration are: [7,27,41,67,69,80,88,90,93,96,97,104,105, 107,113,117]. In these works fitness evaluations for scout bee phase may or may not be counted towards cumulative fitness evaluations. Moreover, in some cases the number of onlooker bees is not the same as the number of employed bees (e.g., [69]). Hence, the correct number of fitness evaluations in this case should be carefully examined.

In addition to the scout bee phase of ABC, a similar problem might also arise when local searches are included within ABC spending even more fitness evaluations per iteration. Some of the included studies that suggested additional local searches into ABC are: [42,45–47,69,77,83,96,100,109,117]. Note that in these studies extra fitness evaluations consumed in local search were not always counted. This might or might not invalidate their conclusions. For example, in [77,83,100] the comparison between ABC and other algorithms is done on CPU time. In that case properly counting fitness evaluations is of lesser importance. Their conclusions from such designed experiments are still valid. However, these studies might not be used in future comparisons as their results heavily depend on the particular hardware used. In such cases it is harder to achieve the same experimental environment for a comparison of the results. The other problem might be that the time reported might sometimes also include the time spent on some other tasks (e.g., garbage collection, operating system task). Some examples of including extra steps into ABC are vividly described in:

- [38] where EABC was proposed (see Fig. 1 in [38]): *"In summation, in order to obtain the improved EABC, we strengthened each phase of the ABC algorithm by adding the characteristics of the PSO algorithm: each bee was given a velocity, as if it were a real bee, to enable it to search in its own direction and at its own pace. Velocity updating processes were provided for initial bees, employed bees and on looker bees; thus, EABC can be regarded as having enhanced the process of solution."* In other words, after initialization, during the employed bee phase and onlooker bee phase extra fitness evaluations were consumed by moving bees according to Particle Swarm Optimization (PSO) rules. Hence, in each cycle twice as many fitness evaluations were consumed (in addition to extra fitness evaluations during the initialization). In such cases comparison should be done on equal number of fitness evaluations (or on CPU time as was done in [77,83,100]). But, the authors used the equal number of cycles: *"For a fair comparison, all of the undetermined parameters are based on the 2000 cycle-execution."*
- [107] where ERABC was proposed by changing the onlooker phase: *"Produce two new candidate food sources V_i and V'_i for the employed bee corresponding to food source X_i using Eqs. (...) and (...), respectively. Select a best solution from $\{V_i, V'_i, X_i\}$."* Clearly, one additional fitness evaluation has been consumed in the onlooker bee phase with respect to original ABC. ERABC proposed also special chaotic search for the scout bee phase spending even more fitness evaluations per cycle. Despite the authors [107] believe that the design of experiment is fair, this is not the case: *"The parameter settings are*

the same as those of [. . .] in order to just compare the performance of ABC, PSABC [. . .] together with ERABC on all the test functions. In other words, the maximum number of cycles MCN is set to 1000, and the number of population size SN is taken as 40 and the control parameter limit equals 100 for all the test functions."

Furthermore, in addition to the scout bee phase of ABC and various local searches there is another possible caveat for improperly counting fitness evaluations within the initialization phase. Indeed, [30–32] integrated a chaotic search and opposition-based approach into initialization phase consuming twice as much fitness evaluations as other approaches for the initial population.

Using more fitness evaluations per iteration than other competitive algorithms also invalidates convergence graph comparison amongst the compared algorithms. It is clear that convergence to a solution using more fitness evaluations per iteration will be faster. Hence, convergence graphs comparing different meta-heuristic algorithms should be based on the number of fitness evaluations or CPU time (the latter is less desirable due to difficulties in future comparisons). An example where researchers reported fast convergence of ABC variant are given in [58] (see Fig. 1 in [58]): *"In PS-ABC algorithm, every employed bee or looked bee could predict their next solutions by three different solution search equations, and select the best one. Then the bee could decide where they should go to explore or exploit. Simulation results show that the PS-ABC could not only have the advantages of I-ABC, but also own the bright sides of ABC and GABC. Namely, PS-ABC owns an extremely fast convergence speed like I-ABC and very good search performance."* In such a case comparing an equal number of iterations is of course inappropriate. Convergence graphs (Figs. 2–15 in [58]) comparing ABC, GABC, I-ABC and PS-ABC do not show real convergence speed. Derived conclusion in [58]: *"The results indicate that the search ability of PS-ABC has been improved. In addition, the new hybrid algorithm shows the very fast convergence like I-ABC"* are based on wrong convergence graphs, and hence it is at least questionable.

Finally, in several papers (e.g., [7,11,31,32,89,93,97,104,105]) some minor inconsistencies with original ABC implementation were found regarding the scout bee phase. While in [49] the condition for scout bee was stated as $(max(trial_i) >= limit)$ in the aforementioned papers the condition was not the same ($(max(trial_i) > limit)$ or $(max(trial_i) = limit)$). This is a variation of the original ABC algorithm and readers just need to be aware of this fact.

In summary, the study of some existing works using ABC reveals that comparison amongst ABC variants and other algorithms is less than satisfactory. This is in accordance with a recent study [61] where it was shown that for high dimensional problems, with tuned control parameters and local search, the original ABC performs equally well as ABC variants.

3. Replication of some ABC experiments

In order to show how comparison based on an equal number of iterations but different number of fitness evaluations consumed, might or might not influence the drawn conclusions, we replicated the experiment from the original ABC paper [49]. In order to evaluate performance of ABC its inventors in [49] took a small experiment on classical benchmark functions presented in [57], where the results for DE, PSO and EA were shown on the following numerical benchmark functions: 2-dimensional Schaffer function $f1$, 5-dimensional Sphere function $f2$, 50-dimensional Griewank function $f3$, 50-dimensional Rastrigin function $f4$, and 50-dimensional Rosenbrock function $f5$. In comparison [57] DE, PSO and EA applied 100,000 fitness evaluations for functions $f1-f2$, and 500,000 fitness evaluations for $f3-f5$. To replicate the experiment from [57] and compared results with ABC the following control parameter setting was used in [49]: $colony\_size = 100$ $(employed\_bees = 50, onlooker\_bees = 50)$, $MCN = 1000$ for $f1-f2$ and $MCN = 5000$ for $f3-f5$, and $runs = 30$. However, this setting is only an approximation as on each iteration (cycle) one extra fitness evaluation can be applied for scout bee in addition to fitness evaluations consumed during the initialization phase. Note that only those control parameters were mentioned which influenced the number of fitness evaluations (others can be found in [49] and were applied also in our work). The goal of our replication of the experiment in [49] was to check whether the conclusions were still valid if all the compared algorithms (DE, PSO, EA, and ABC) had used equal numbers of fitness evaluations taking into account that extra fitness evaluations can be consumed during the scout bee phase in addition to those also consumed during the initialization phase. The results obtained by the first implementation, the stopping condition of which is dependent on fitness evaluation (Algorithm 1 from Appendix A), and for the second implementation (Algorithm 2 from Appendix A), the stopping condition of which is defined in terms of *MCN* and *SN*, are presented in Table 1 (note that for practical reasons values less than $E-12$ were reported as 0 in [49] as well as in Table 1).

By comparing the results obtained in [49] (column ABC [49] in Table 1) and our replication of the experiment with basic ABC (column ABC(*MCN*) in Table 1) we can conclude that the results for $f1-f4$ are the same, whilst there is small but insignificant difference on $f5$. However, it can also be noted that the number of fitness evaluations (Table 1) for $f1-f2$ and for $f3-f5$ were not 100,000 and 500,000, respectively, as assumed in [49] and actually used in [57]. ABC consumed as much as 1046 extra fitness evaluations for $f1$ and as low as 50 extra evaluations for $f5$. The number of extra fitness evaluations heavily depends on the setting of the ABC control parameter 'limit', which was set in [49] as $limit = employed\_bees * D$. Hence, the solution was abandoned in each cycle after $50 * 2 = 100$ trials in $f1$, $50 * 5 = 250$ trials in $f2$, and $50 * 50 = 2500$ trials in $f3-f5$. This explains very different number of extra fitness evaluations in the scout bee phase for $f1-f5$. However, our goal at this point was not to find the best setting for the control parameter 'limit' but to exactly replicate the original experiment in [57]. Hence, we ran ABC using two implementations from Appendix A (Algorithms 1 and 2), which strictly obeyed a

Table 1
Replication of experiments.

f	ABC [49]	ABC(MCN)	#FEs	ABC$_{imp1}$ (FEs)	#FEs	ABC$_{imp2}$ (FEs)	#FEs
$f1$	0 ± 0	0 ± 0	101,064.2 ± 0.49	0 ± 0	100,000	0 ± 0	99,950
$f2$	0 ± 0	0 ± 0	100,263.7 ± 3.62	0 ± 0	100,000	0 ± 0	99,950
$f3$	0 ± 0	0 ± 0	500,100.0 ± 0.0	0 ± 0	500,000	0 ± 0	499,950
$f4$	0 ± 0	0 ± 0	500,069.26 ± 2.90	0 ± 0	500,000	0 ± 0	499,950
$f5$	0.133109 ± 0.262242	0.116999 ± 0.156896	500,050.0 ± 0.0	0.168245 ± 0.225376	500,000	0.156299 ± 0.211091	499,950

condition not to consume more than the maximum allowed fitness evaluations. Comparing the results in Table 1 (columns ABC [49], ABC(MCN), ABC$_{imp1}$(FEs), and ABC$_{imp2}$(FEs)) we can again notice that there is no difference in the results for $f1$–$f4$, and an insignificant difference for $f5$. For significance testing [23,33] the statistical z-Test [10] has been applied since the results for mean and standard deviation were presented in [49]. The null hypothesis H_0 (mean values are equal: $H_0 : \mu_1 = \mu_2$) is accepted if the p-value is bigger than the significance level α ($\alpha = 0.05$ was used in our case). Since all p-values were bigger (the smallest p-value 0.311 was between ABC(MCN) vs. ABC$_{imp1}$(FEs)) than the significance level the null hypothesis H_0 can be accepted with 95% confidence. In other words, results for $f5$ in Table 1 are insignificantly different and the conclusion from [49]: *"As seen from the results presented in Table 1, the ABC algorithm produces the best performance among the algorithms considered in the present investigation"* can still be supported if applied on this particular small benchmark functions $f1$–$f5$. This is an example that despite inexact replication of experiments between [49,57] the drawn conclusions can still be valid but need to be supported by additional experiments, where it was shown that better results were found by ABC not due to extra fitness evaluations but indeed due to better exploration and exploitation of the search space. In fact, due to a particular implementation (Algorithm 2) the ABC$_{imp2}$(FEs) even consumed slightly less fitness evaluations than allowed (Table 1) as during the initialization phase only $SN = NP/2$ solutions are initially generated. For this ABC implementation the experiment was replicated even under stricter conditions (fewer fitness evaluations than allowed) [21].

The next goal of the very influential paper [49] was to analyze the behavior of ABC using different colony sizes. The colony sizes of 10, 50 and 100 were compared in [49]. However, the comparison in [49] was based on an equal number of iterations (cycles). Obviously, with employed bees size 5 ABC will spend approximately 10 fitness evaluations (employed_bees, onlooker_bees, and a possible scout bee) per iteration (cycle), whilst with the employed bees size 50 ABC will spend approximately 100 fitness evaluations per iteration (cycle). Hence, comparison based on iterations (cycles) is at least harder to investigate, if appropriate at all. From the convergence graphs presented in [49] (Figs. 7–11) the comparison among different colony sizes is difficult to follow since iterations (cycles) should be mentally converted into the number of fitness evaluations consumed in each iteration (cycle). A task which is almost impossible to do and wrong conclusions can be very easily derived (See Section 2). Therefore, our goal was to replicate the experiment in [49], but this time comparison was based on equal number of fitness evaluations. The results are presented in Figs. 1 and 2 (the same experimental setting for $f1$–$f5$ was used as in the previous experiment with the results presented in Table 1).

Comparison of the convergence graphs from [49] (Left Figs. 1 and 2), which are based on an equal number of iterations (cycles), with the convergence graphs (Right Figs. 1 and 2), which are based on equal number of fitness evaluations, reveals

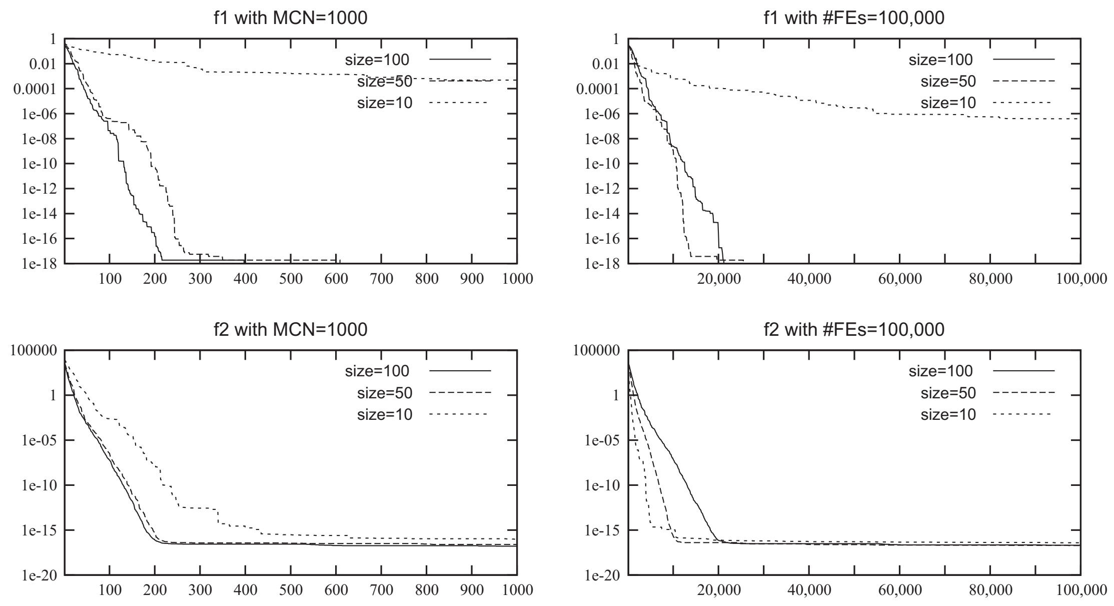

Fig. 1. $f1$ and $f2$ with MCN = 1000 (left) and #FEs = 100,000 (right).

quite a different situation. It can be seen from the convergence graphs (Right Figs. 1 and 2) that the most appropriate colony size heavily depends on the problem. For example, the most cost-effective (good and fast solution) for $f2$–$f3$ is $colony\_size = 10$, for $f1$ and $f4$ is $colony\_size = 50$, and $colony\_size = 100$ for $f5$. On the other hand, the convergence graphs (right Figs. 1 and 2) reveal that $colony\_size = 10$ is an unsuitable setting for $f1, f4$–$f5$, and $colony\_size = 50$ is an unsuitable setting for $f5$, whilst $colony\_size = 100$ for $f1$–$f5$ always find a good solution, although the convergence is not amongst the fastest (for obvious reasons). From the experiments performed in [49] the authors concluded: *"From Table 4 and Figs. 7–12, it can be concluded that as the population size increases, the algorithm produces better results. However, after a sufficient value for colony size, any increment in the value does not improve the performance of the ABC algorithm significantly. For the test problems carried out in this work, the colony size of 50–100 can provide an acceptable convergence speed for search."* According to the results presented in this section the aforementioned conclusion from [49] can only be partially supported. The first part of conclusion from [49]: *"as the population size increases, the algorithm produces better results"* were unsupported by our experiments. It seems that the 'rule of thumb': (bigger population is needed for multimodal, high dimensional and harder optimization problems) often used in meta-heuristic algorithms is also valid for ABC. Whilst the second part of the conclusion from [49]: *"For the test problems carried out in this work, the colony size of 50–100 can provide an acceptable convergence speed for search"* can still be supported. Indeed, $colony\_size = 100$ for included benchmark functions $f1$–$f5$ is always a suitable choice as the obtained results are of good quality, but not always of the fastest convergence (see for example $f3$ in Fig. 2). Our finding that the most appropriate colony size depends on the problem is not surprising and with accordance to the theory and practice of meta-heuristic algorithms. For example, for the optimization of simple uni-modal and low dimensional functions such as Sphere function $f2$, a small population size is sufficient and the most effectively ($f2$ in Fig. 1). Our finding is also in accordance with the recent study [61] where it was shown that with properly tuned control parameters (including population size) the original ABC results can be improved. This is an example that the drawn conclusions cannot always be valid if algorithms are compared based on an equal number of iterations, but with different number of fitness evaluations consumed in each iteration.

The previous two experiments exhibit different errors made during the counting of fitness evaluations. An obvious question is whether a small number of extra fitness evaluations would also lead to minor differences in solution quality. The following small experiment shows that this is not the case. From $f4$ (Fig. 2, $colony\_size = 100$) an error graph was

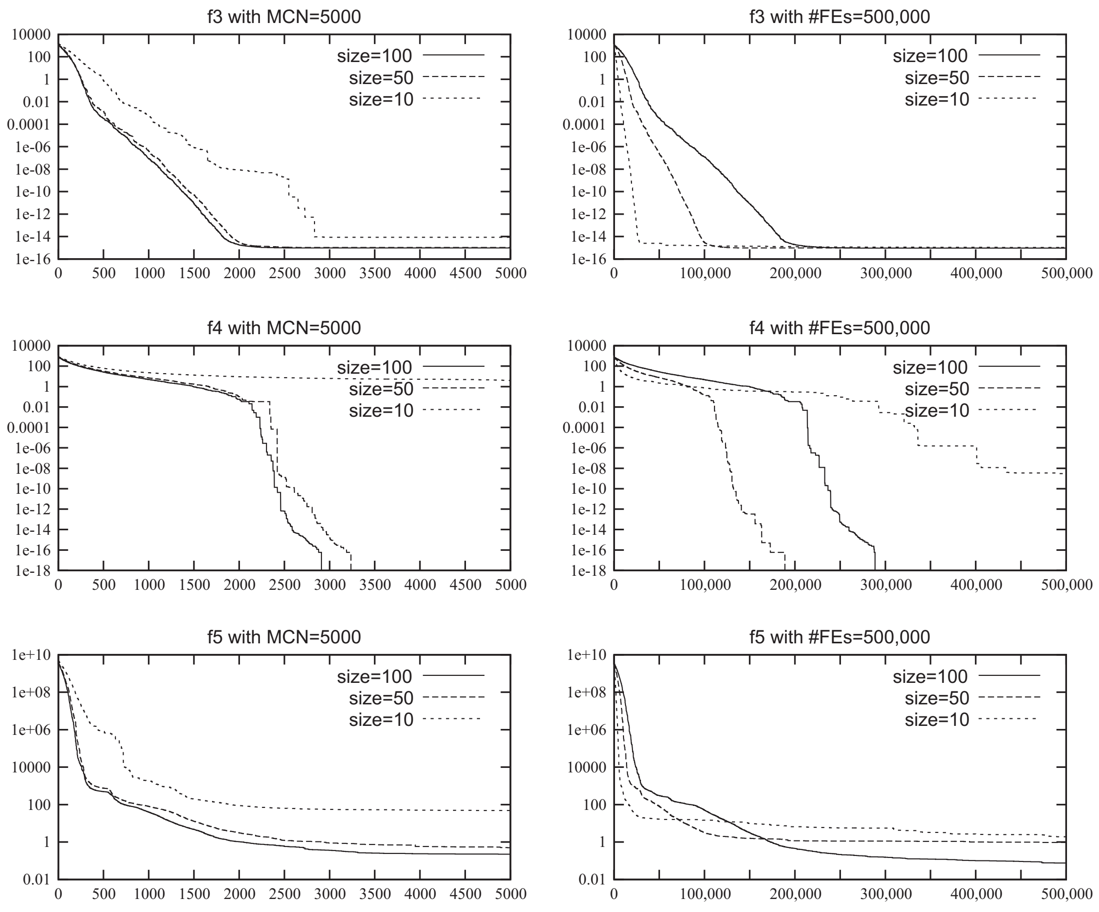

Fig. 2. $f3$, $f4$ and $f5$ with $MCN = 5000$ (left) and $\#FEs = 500{,}000$ (right).

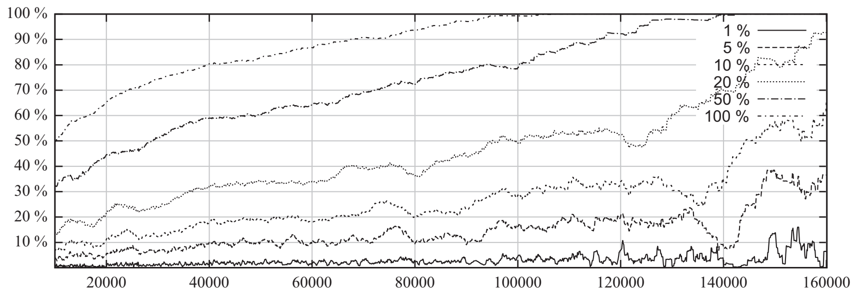

Fig. 3. Error graph for $f4$ from Fig. 2 and $colony\_size = 100$.

generated (i.e., Fig. 3) showing how big an error is when a certain percentage of extra fitness evaluations are allowed. It can be noticed from Fig. 3 that when increasing the number of fitness evaluations by 1%, 5%, 10%, 20%, 50%, and 100% the error, or improvement in fitness, can be as great as 18%, 40%, 66%, 92%, 100%, and 100% respectively. Of course, such curves are very problem-dependent and cannot be generalized to other problems. Furthermore, it depends on the evolutionary stage. During the early stages improvements by a small number of fitness evaluations might be great, but at later stages when improvements are not achieved anymore more fitness evaluations only waste processing time. For this reason the graph in Fig. 3 shows the middle stage of evolution (from 10,000 to 160,000 fitness evaluations) for $f4$ regarding the convergence graph from Fig. 2.

4. Conclusions

The main objective of this study was to remind about several misconceptions when comparing meta-heuristic algorithms based on an equal number of iterations. Such comparison is unfair if compared algorithms do not consume equal number of fitness evaluations per iterations. It was shown that such a practice is omnipresent in the case of ABC variants (Section 2) as many researchers neglected fitness evaluations consumed during the scout bee phase and during additional local search (if introduced into ABC variants) despite that in the publicly available ABC code it was clearly observable that scout bees consumed fitness evaluations.

In order to overcome the problem of unfair comparison based on an equal number of iterations ABC was re-implemented (see Algorithms 1 and 2 in Appendix A) assuring that ABC did not consume more fitness evaluations than allowed. As such ABC can be easily used for fair comparisons with other algorithms in the future, where there may be no threats of validity to conclusions due to more fitness evaluations consumed than allowed. ABC is a very successful meta-heuristic algorithm and many new ABC variants are foreseen. Hence, it is very important that fair comparisons with other algorithms are achieved and that any conclusions drawn are supported by the performed experiments. Note that the problem discussed in this paper is not only pertinent for ABC variants but similar problems have also been identified in other meta-heuristic algorithms. For example, in the original implementation of TLBO, the number of fitness evaluations during the duplicate elimination step was only approximated [81], which can be regarded as imprecise and incorrect counting of fitness evaluations. This is especially true when an approximation was defined for a particular population size and then used for different population sizes (as in [103]). Furthermore, researchers have often reported that comparisons between different algorithms are unsuitable since some compared algorithms consume a much higher amount of CPU time [72]. This is often the case, but not always when comparison is done over an equal number of iterations, but algorithms consume much more fitness evaluations per iteration. Overall, it is quite surprising how omnipresent the discussed problem is. Hence, we call for attention to the current situation.

Appendix A

Implementation of basic ABC is freely available at http://mf.erciyes.edu.tr/abc, which is an additional reason for ABC's popularity. However, this implementation is based on an equal number of iterations (cycles) and hence less suitable when comparison between meta-heuristic algorithms is done on an equal number of fitness evaluations. The pseudo-code of ABC using counter for fitness evaluations and termination criterion based on maximum fitness evaluations is presented in Algorithm 1. The pseudo-code of ABC using termination criterion based on *MCN* and *SN*, but consuming one fitness evaluation less during the employed bee phase for an employed bee that will become a scout, is presented in Algorithm 2. Note that in this implementation, the scout bee phase is between the employed bee phase and the onlooker bee phase, whilst it is after the onlooker bee phase in the basic ABC or in implementation 1. It is also possible to sort the phases in different orders. This shows the flexibility of ABC.

Algorithm 1. The pseudo-code of $\text{ABC}_{imp1}(FEs)$.

```
Data: Set the control parameters of the ABC algorithm
$SN$: Number of Foods
$limit$: Maximum number of trial for abandoning a source
$MFE$: Maximum number of fitness evaluations
begin
    //Initialization;
    $num\_eval \longleftarrow 0$ ;
    for $s = 1$ to $SN$ do
        $X(s) \longleftarrow$ random solution by Eq. 1 [3];
        $f_s \longleftarrow f(X(s))$;
        $trial(s) \longleftarrow 0$;
        $num\_eval++$ ;
    end
    repeat
        //Employed Bees Phase;
        for $s = 1$ to $SN$ do
            $x' \longleftarrow$ a new solution produced by Eq. 2 [3];
            $f(x') \longleftarrow$ evaluate new solution;
            $num\_eval++$;
            if $f(x') < f_s$ then
                $X(s) \longleftarrow x'$; $f_s \longleftarrow f(x')$; $trial(s) \longleftarrow 0$;
            else
                $trial(s) \leftarrow trial(s) + 1$;
            end
            if $num\_eval == MFE$ then
                Memorize the best solution achieved so far and exit main repeat;
            end
        end
        Calculate the probability values $p_i$ for the solutions using fitness values by Eqs. 3 and 4 [3];
        //Onlooker bee phase;
        $s \longleftarrow 1$; $t \longleftarrow 1$ ;
        repeat
            $r \longleftarrow rand(0, 1)$;
            if $r < p(s)$ then
                $t \longleftarrow t + 1$;
                $x' \longleftarrow$ a new solution produced by Eq. 2 [3];
                $f(x') \longleftarrow$ evaluate new solution;
                $num\_eval++$;
                if $f(x') < f_s$ then
                    $X(s) \longleftarrow x'$; $f_s \longleftarrow f(x')$; $trial(s) \longleftarrow 0$;
                else
                    $trial(s) \longleftarrow trial(s) + 1$;
                end
                if $num\_eval == MFE$ then
                    Memorize the best solution achieved so far and exit main repeat;
                end
            end
            $s \longleftarrow (s \mod SN) + 1$;
        until $t = SN$ ;
        //Scout Bee Phase;
        $mi \longleftarrow \{s : trial(s) = max(trial)\}$;
        if $trial(mi) >= limit$ then
            $X(mi) \longleftarrow$ random solution by Eq. 1 [3];
            $f_{mi} \longleftarrow f(X(mi))$;
            $num\_eval++$ ;
            $trial(mi) \longleftarrow 0$;
            if $num\_eval == MFE$ then
                Memorize the best solution achieved so far and exit main repeat;
            end
        end
        Memorize the best solution achieved so far;
    until $num\_eval = MFE$ ;
end
```

Algorithm 2. The pseudo-code of the ABC$_{imp2}$(*FEs*).

```
Data: Set the control parameters of the ABC algorithm
$SN$: Number of Foods
$limit$: Maximum number of trial for abandoning a source
$MCN$: Maximum number of cycles
begin
    //Initialization;
    $num\_eval \longleftarrow 0$ ;
    for $s = 1$ to $SN$ do
        $X(s) \longleftarrow$ random solution by Eq. 1 [3];
        $f_s \longleftarrow f(X(s))$;
        $trial(s) \longleftarrow 0$;
        $num\_eval++$;
    end
    $cycle \longleftarrow 1$;
    while $cycle < MCN$ do
        //Employed Bees Phase;
        $mi \longleftarrow \{s : trial(s) = max(trial)\}$;
        for $s = 1$ to $SN$ do
            if $(trial(s) < limit$ or $s \,!= mi)$ then
                $x' \longleftarrow$ a new solution produced by Eq. 2 [3];
                $f(x') \longleftarrow$ evaluate new solution;
                $num\_eval++$ ;
                if $f(x') < f_s$ then
                    $X(s) \longleftarrow x'$; $f_s \longleftarrow f(x')$; $trial(s) \longleftarrow 0$;
                else
                    $trial(s) \longleftarrow trial(s) + 1$;
                end
            end
        end
        Memorize the best solution achieved so far;
        //Scout Bee Phase;
        if $(trial(mi) >= limit$ then
            $X(mi) \longleftarrow$ random solution by Eq. 1 [3];
            $f_{mi} \longleftarrow f(X(mi))$;
            $num\_eval++$ ;
            $trial(mi) \longleftarrow 0$;
        end
        Calculate the probability values $p_i$ for the solutions using fitness values by Eqs. 3 and 4 [3];
        //Onlooker Bees Phase;
        $s \longleftarrow 1$; $t \longleftarrow 1$ ;
        while $t \leq SN$ do
            $r \longleftarrow rand(0, 1)$;
            if $r < p(s)$ then
                $t \longleftarrow t + 1$;
                $x' \longleftarrow$ a new solution produced by Eq. 2 [3];
                $f(x') \longleftarrow$ evaluate new solution;
                $num\_eval++$ ;
                if $f(x') < f_s$ then
                    $X(s) \longleftarrow x'$; $f_s \longleftarrow f(x')$; $trial(s) \longleftarrow 0$;
                else
                    $trial(s) \longleftarrow trial(s) + 1$;
                end
            end
            $s \longleftarrow (s \bmod SN) + 1$;
        end
        Memorize the best solution achieved so far;
        $cycle++$;
    end
end
```

References

[1] M.R. Adaryani, A. Karami, Artificial bee colony algorithm for solving multi-objective optimal power flow problem, Electr. Power Energy Syst. 53 (2013) 219–230.
[2] A. Ahrari, M. Shariat-Panahi, A.A. Atai, GEM: a novel evolutionary optimization method with improved neighborhood search, Appl. Math. Comput. 210 (2009) 376–386.
[3] B. Akay, D. Karaboga, A modified Artificial Bee Colony algorithm for real-parameter optimization, Inform. Sci. 192 (2012) 120–142.
[4] R. Akbari, A. Mohammadi, K. Ziarati, A novel bee swarm optimization algorithm for numerical function optimization, Commun. Nonlin. Sci. Numer. Simul. 15 (2010) 3142–3155.
[5] D. Aydin, S. Ozyon, C. Yasar, T. Liao, Artificial bee colony algorithm with dynamic population size to combined economic and emission dispatch problem, Electr. Power Energy Syst. 54 (2014) 144–153.
[6] T. Bäck, D.B. Fogel, Z. Michalewicz, Handbook of Evolutionary Computations, Oxford University Press, 1996.
[7] A. Banharnsakun, T. Achalakul, B. Sirinaovakul, The best-so-far selection in Artificial Bee Colony algorithm, Appl. Soft Comput. 11 (2011) 2888–2901.
[8] J.C. Bansal, H. Sharma, S.S. Jadon, Artificial bee colony algorithm: a survey, Int. J. Advan. Intell. Parad. 5 (2013) 123–159.
[9] R.S. Barr, B.L. Golden, J.P. Kelly, M.G.C. Resende, W.R. Stewart Jr., Designing and reporting on computational experiments with heuristic methods, J. Metaheur. 1 (1995) 9–32.
[10] T. Bartz-Beielstein, Experimental Research in Evolutionary Computation: The New Experimentalism, Springer, 2006.
[11] S. Biswas, A. Chatterjee, S.K. Goswami, An artificial bee colony-least square algorithm for solving harmonic estimation problems, Appl. Soft Comput. 13 (2013) 2343–2355.
[12] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: overview and conceptual comparison, ACM Comput. Surv. 35 (2003) 268–308.
[13] A.L. Bolaji, A.T. Khader, M.A. Al-Betar, M.A. Awadallah, Artificial bee colony algorithm, its variants and applications: a survey, J. Theoret. Appl. Inform. Technol. 47 (2013) 434–459.
[14] K. Chandrasekaran, S. Hemamalini, S.P. Simon, N.P. Padhy, Thermal unit commitment using binary/real coded artificial bee colony algorithm, Electr. Power Syst. Res. 84 (2012) 109–119.
[15] P.-T. Chang, J.-H. Lee, A fuzzy DEA and knapsack formulation integrated model for project selection, Comp. Operat. Res. 39 (2012) 112–125.
[16] S.-M. Chen, A. Sarosh, Y.-F. Dong, Simulated annealing based artificial bee colony algorithm for global numerical optimization, Appl. Math. Comput. 219 (2012) 3575–3589.
[17] P. Civicioglu, Transforming geocentric cartesian coordinates to geodetic coordinates by using differential search algorithm, Comp. Geosci. 46 (2012) 229–247.
[18] W.G. Cochran, Sampling Techniques, John Wiley & Sons, New York, NY, 1977.
[19] C.C. Columbus, S.P. Simon, Profit based unit commitment: a parallel ABC approach using a workstation cluster, Comp. Electr. Eng. 38 (2012) 724–745.
[20] M. Črepinšek, S.-H. Liu, M. Mernik, Exploration and exploitation in evolutionary algorithms: a survey, ACM Comput. Surv. 45 (2013) 35:1–35:33.
[21] M. Črepinšek, S.-H. Liu, M. Mernik, Replication and comparison of computational experiments in applied evolutionary computing: common pitfalls and guidelines to avoid them, Appl. Soft Comput. 19 (2014) 161–170.
[22] G. Deng, Z. Xu, X. Gu, A discrete artificial bee colony algorithm for minimizing the total flow time in the blocking flow shop scheduling, Chin. J. Chem. Eng. 20 (2012) 1067–1073.
[23] J. Derrac, S. Garcia, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evolution. Comput. 1 (1) (2011) 3–18.
[24] A.E. Eiben, C. Schippers, On evolutionary exploration and exploitation, Fund. Inform. 35 (1998) 35–50.
[25] A.E. Eiben, M. Jelasity, A critical note on experimental research methodology in EC, in: Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002), 2002, pp. 582–587.
[26] A.E. Eiben, J.E. Smith, Introduction to Evolutionary Computing, Springer, 2008.
[27] M. El-Abd, Performance assessment of foraging algorithms vs. evolutionary algorithms, Inform. Sci. 182 (2012) 243–263.
[28] G. Fei, F. Feng-xia, D. Yan-fang, Q. Yi-bo, I. Balasingham, A novel non-Lyapunov approach through artificial bee colony algorithm for detecting unstable periodic orbits with high orders, Exp. Syst. Appl. 39 (2012) 12389–12397.
[29] F. Gao, F.-X. Fei, Q. Xu, Y.-F. Deng, Y.-B. Qi, I. Balasingham, A novel artificial bee colony algorithm with space contraction for unknown parameters identification and time-delays of chaotic systems, Appl. Math. Comput. 219 (2012) 552–568.
[30] W. Gao, S. Liu, Improved artificial bee colony algorithm for global optimization, Inform. Process. Lett. 111 (2011) 871–882.
[31] W. Gao, S. Liu, A modified artificial bee colony algorithm, Comp. Operat. Res. 39 (2012) 687–697.
[32] W. Gao, S. Liu, L. Huang, A global best artificial bee colony algorithm for global optimization, J. Comput. Appl. Math. 236 (2012) 2741–2753.
[33] S. Garcia, D. Molina, M. Lozano, F. Herrera, A study on the use of non-parametric tests for analysing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization, J. Metaheur. 15 (6) (2009) 617–644.
[34] H. Garg, M. Rani, S.P. Sharma, Predicting uncertain behavior of press unit in a paper industry using artificial bee colony and fuzzy Lambda–Tau methodology, Appl. Soft Comput. 13 (2013) 1869–1881.
[35] W.-C. Hong, Electric load forecasting by seasonal recurrent SVR (support vector regression) with chaotic artificial bee colony algorithm, Energy 36 (2011) 5568–5578.
[36] M.-H. Horng, Multilevel thresholding selection based on the artificial bee colony algorithm for image segmentation, Exp. Syst. Appl. 38 (2011) 13785–13791.
[37] T.-J. Hsieh, H.-F. Hsiao, W.-C. Yeh, Forecasting stock markets using wavelet transforms and recurrent neural networks: an integrated system based on artificial bee colony algorithm, Appl. Soft Comput. 11 (2011) 2510–2525.
[38] T.-J. Hsieh, H.-F. Hsiao, W.-C. Yeh, Mining financial distress trend data using penalty guided support vector machines based o nhybrid of particle swarm optimization and artificial bee colony algorithm, Neurocomputing 82 (2012) 196–206.
[39] T.-J. Hsieh, W.-C. Yeh, Penalty guided bees search for redundancy allocation problems with a mix of components in series–parallel systems, Comp. Operat. Res. 39 (2012) 2688–2704.
[40] C.-C. Hsu, H.-C. Chen, K.-K. Huang, Y.-M. Huang, A personalized auxiliary material recommendation system based on learning style on Facebook applying an artificial bee colony algorithm, Comp. Math. Appl. 64 (2012) 1506–1513.
[41] S.-J. Huang, X.-Z. Liu, Application of artificial bee colony-based optimization for fault section estimation in power systems, Electr. Power Energy Syst. 44 (2013) 210–218.
[42] M. Husseinzadeh Kashan, N. Nahavandi, A. Husseinzadeh Kashan, DisABC: a new artificial bee colony algorithm for binary optimization, Appl. Soft Comput. 12 (2012) 342–352.
[43] R. Irani, R. Nasimi, Application of artificial bee colony-based neural network in bottom hole pressure prediction in underbalanced drilling, J. Petrol. Sci. Eng. 78 (2011) 6–12.
[44] H.T. Jadhav, R. Roy, Gbest guided artificial bee colony algorithm for environmental/economic dispatch considering wind power, Exp. Syst. Appl. 40 (2013) 6385–6399.
[45] F. Kang, J. Li, Q. Xu, Structural inverse analysis by hybrid simplex artificial bee colony algorithms, Comp. Struct. 87 (2009) 861–870.
[46] F. Kang, J. Li, Z. Ma, Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions, Inform. Sci. 181 (2011) 3508–3531.
[47] F. Kang, J. Li, H. Li, Artificial bee colony algorithm and pattern search hybridized for global optimization, Appl. Soft Comput. 13 (2013) 1781–1791.

[48] D. Karaboga, An Idea Based on Honey Bee Swarm for Numerical Optimization. Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.
[49] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Appl. Soft Comput. 8 (2008) 687–697.
[50] D. Karaboga, B. Akay, A comparative study of artificial bee colony algorithm, Appl. Math. Comput. 214 (2009) 108–132.
[51] D. Karaboga, B. Akay, A survey: algorithms simulating bee swarm intelligence, Artif. Intell. Rev. 31 (2009) 61–85.
[52] D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga, A comprehensive survey: artificial bee colony (ABC) algorithm and applications, Artif. Intell. Rev. 42 (2014) 21–57.
[53] N. Karaboga, F. Latifoglu, Adaptive filtering noisy trans cranial Doppler signal by using artificial bee colony algorithm, Eng. Appl. Artif. Intell. 26 (2013) 677–684.
[54] N. Karaboga, F. Latifoglu, Elimination of noise on transcranial Doppler signal using IIR filters designed with artificial bee colony – ABC-algorithm, Dig. Sig. Process. 223 (2013) 1051–1058.
[55] Y.S. Kavian, A. Rashedi, A. Mahani, Z. Ghassemlooy, Routing and wavelength assignment in optical networks using Artificial Bee Colony algorithm, Optik 124 (2013) 1243–1249.
[56] B. Kitchenham, P. Brereton, A systematic review of systematic review process research in software engineering, Inform. Softw. Technol. 55 (2013) 2049–2075.
[57] T. Krink, B. Filipič, G.B. Fogel, R. Thomsen, Noisy optimization problems a particular challenge for differential evolution?, in: Proceedings of 2004 Congress on Evolutionary Computation, 2004, pp. 332–339.
[58] G. Li, P. Niu, X. Xiao, Development and investigation of efficient artificial bee colony algorithm for numerical function optimization, Appl. Soft Comput. 12 (2012) 320–332.
[59] Y. Li, Y. Wang, B. Li, A hybrid artificial bee colony assisted differential evolution algorithm for optimal reactive power flow, Electr. Power Energy Syst. 52 (2013) 25–33.
[60] X. Liao, J. Zhou, R. Zhang, Y. Zhang, An adaptive artificial bee colony algorithm for long-term economic dispatch in cascaded hydropower systems, Electr. Power Energy Syst. 43 (2012) 1340–1345.
[61] T. Liao, D. Aydin, T. Stützle, Artificial bee colonies for continuous optimization: experimental analysis and improvements, Swarm Intell. 7 (2013) 327–356.
[62] M. Ma, J. Liang, M. Guo, Y. Fan, Y. Yin, SAR image segmentation based on artificial bee colony algorithm, Appl. Soft Comput. 11 (2011) 5205–5214.
[63] T. Malakar, S.K. Goswami, Active and reactive dispatch with minimum control movements, Electr. Power Energy Syst. 44 (2013) 78–87.
[64] V.J. Manoj, E. Elias, Artificial bee colony algorithm for the design of multiplier-less nonuniform filter bank transmultiplexer, Inform. Sci. 192 (2012) 193–203.
[65] M. Manuel, E. Elias, Design of frequency response masking FIR filter in the Canonic Signed digit space using modified artificial bee colony algorithm, Eng. Appl. Artif. Intell. 26 (2013) 660–668.
[66] E. Mezura-Montes, O. Cetina-Dominguez, Empirical analysis of a modified artificial bee colony for constrained numerical optimization, Appl. Math. Comput. 218 (2012) 10943–10973.
[67] E. Miandoabchi, F. Daneshzand, W.Y. Szeto, R. Zanjirani Farahani, Multi-objective discrete urban road network design, Comp. Operat. Res. 40 (2013) 2429–2449.
[68] V. Mohanraj, M. Chandrasekaran, J. Senthilkumar, S. Arumugam, Y. Suresh, Ontology driven bees foraging approach based self adaptive online recommendation system, J. Syst. Softw. 85 (2012) 2439–2450.
[69] E. Moreira Bernardino, A. Moreira Bernardino, J.M. Sanchez-Perez, J.A. Gomez-Pulido, M.A. Vega-Rodriguez, Solving large-scale SONET network design problems using bee-inspired algorithms, Opt. Switch. Network. 9 (2012) 97–117.
[70] E. Moreira Bernardino, A. Moreira Bernardino, J.M. Sanchez-Perez, J.A. Gomez-Pulido, M.A. Vega-Rodriguez, Swarm optimisation algorithms applied to large balanced communication networks, J. Netw. Comp. Appl. 36 (2013) 504–522.
[71] P. Moscato, On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. Technical Report, California Institute of Technology, Concurrent Computation Program 158-79, 1989.
[72] A. Mozaffari, M. Gorji-Bandpy, T.B. Gorji, Optimal design of constraint engineering systems: application of mutable smart bee algorithm, Int. J. Bio-Insp. Comput. 4 (2012) 167–180.
[73] R. Mukherjee, S. Chakraborty, S. Samanta, Selection of wire electrical discharge machining process parameters using non-traditional optimization algorithms, Appl. Soft Comput. 12 (2012) 2506–2516.
[74] M. Nikolić, D. Teodorović, Empirical study of the Bee Colony Optimization (BCO) algorithm, Exp. Syst. Appl. 40 (2013) 4609–4620.
[75] I.M.S. de Oliveira, R. Schirru, Swarm intelligence of artificial bees applied to In-core fuel management optimization, Ann. Nucl. Energy 38 (2011) 1039–1045.
[76] S. Ozyon, D. Aydin, Incremental artificial bee colony with local search to economic dispatch problem with ramp rate limits and prohibited operating zones, Energy Convers. Manage. 65 (2013) 397–407.
[77] Q.-K. Pan, M.F. Tasgetiren, P.N. Suganthan, T.J. Chua, A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem, Inform. Sci. 181 (2011) 2455–2468.
[78] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic mapping studies in software engineering, in: Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE'08), 2008, pp. 71–80.
[79] R.V. Rao, P.J. Pawar, Parameter optimization of a multi-pass milling process using non-traditional optimization algorithms, Appl. Soft Comput. 10 (2010) 445–456.
[80] R.V. Rao, V.K. Patel, Optimization of mechanical draft counter flow wet-cooling tower using artificial bee colony algorithm, Energy Convers. Manage. 52 (2011) 2611–2622.
[81] R.V. Rao, V.K. Patel, An elitist teaching–learning-based optimization algorithm for solving complex constrained optimization problems, Int. J. Indust. Eng. 3 (2013) 535–560.
[82] M.M. Rashidi, N. Galanis, F. Nazari, A. Basiri Parsa, L. Shamekhi, Parametric analysis and optimization of regenerative Clausius and organic Rankine cycles with two feedwater heaters using artificial bees colony and artificial neural network, Energy 36 (2011) 5728–5740.
[83] F.J. Rodriguez, M. Lozano, C. Garcia-Martinez, J.D. Gonzalez-Barrera, An artificial bee colony algorithm for the maximally diverse grouping problem, Inform. Sci. 230 (2013) 183–196.
[84] S. Saadi, A. Guessoum, M. Bettayeb, ABC optimized neural network model for image deblurring with its FPGA implementation, Microprocess. Microsyst. 37 (2013) 52–64.
[85] S.L. Sabat, S.K. Udgata, A. Abraham, Artificial bee colony algorithm for small signal model parameter extraction of MESFET, Eng. Appl. Artif. Intell. 23 (2010) 689–694.
[86] O. Safarzadeh, A. Zolfaghari, A. Norouzi, H. Minuchehr, Loading pattern optimization of PWR reactors using Artificial Bee Colony, Ann. Nucl. Energy 38 (2011) 2218–2226.
[87] S. Samanta, S. Chakraborty, Parametric optimization of some non-traditional machining processes using artificial bee colony algorithm, Eng. Appl. Artif. Intell. 24 (2011) 946–957.
[88] A. Singh, An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem, Appl. Soft Comput. 9 (2009) 625–631.
[89] T.S.D. Singh, A. Chatterjee, MMSE design of nonlinear Volterra equalizers using artificial bee colony algorithm, Measurement 46 (2013) 210–219.
[90] M. Sonmez, Artificial bee colony algorithm for optimization of truss structures, Appl. Soft Comput. 11 (2011) 2406–2418.
[91] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.

[92] Z.-G. Su, P.-H. Wang, J. Shen, Y.-F. Zhang, L. Chen, Convenient T–S fuzzy model with enhanced performance using a novel swarm intelligent fuzzy clustering technique, J. Process Control 22 (2012) 108–124.
[93] Z.-G. Su, P.-H. Wang, J. Shen, Y.-G. Li, Y.-F. Zhang, E.-J. Hu, Automatic fuzzy partitioning approach using Variable string length Artificial Bee Colony (VABC) algorithm, Appl. Soft Comput. 12 (2012) 3421–3441.
[94] G. Sun, G. Li, Q. Li, Variable fidelity design based surrogate and artificial bee colony algorithm for sheet metal forming process, Finite Elem. Anal. Des. 59 (2012) 76–90.
[95] H. Sun, H. Lus, R. Betti, Identification of structural models using a modified Artificial Bee Colony algorithm, Comp. Struct. 116 (2013) 59–74.
[96] S. Sundar, A. Singh, A swarm intelligence approach to the quadratic minimum spanning tree problem, Inform. Sci. 180 (2010) 3182–3191.
[97] W.Y. Szeto, Y. Wu, S.C. Ho, An artificial bee colony algorithm for the capacitated vehicle routing problem, Euro. J. Operat. Res. 215 (2011) 126–135.
[98] A.A. Taleizadeh, S.T.A. Niaki, H.-M. Wee, Joint single vendor–single buyer supply chain problem with stochastic demand and fuzzy lead-time, Knowl.-Based Syst. 48 (2013) 1–9.
[99] P. Tapkan, L. Ozbakir, A. Baykasoglu, Modeling and solving constrained two-sided assembly line balancing problem via bee algorithms, Appl. Soft Comput. 12 (2012) 3343–3355.
[100] M.F. Tasgetiren, Q.-K. Pan, P.N. Suganthan, A.H.-L. Chen, A discrete artificial bee colony algorithm for the total flow time minimization in permutation flow shops, Inform. Sci. 181 (2011) 3459–3475.
[101] M.F. Tasgetiren, Q.-K. Pan, P.N. Suganthan, A. Oner, A discrete artificial bee colony algorithm for the no-idle permutation flowshop scheduling problem with the total tardiness criterion, Appl. Math. Model. 37 (2013) 6758–6779.
[102] J.-P. Tien, T.-H.S. Li, Hybrid Taguchi-chaos of multilevel immune and the artificial bee colony algorithm for parameter identification of chaotic systems, Comp. Math. Appl. 64 (2012) 1108–1119.
[103] G. Waghmare, Comments on a note on teaching–learning-based optimisation algorithm, Inform. Sci. 229 (2013) 159–169.
[104] X. Wang, X. Xie, T.C.E. Cheng, A modified artificial bee colony algorithm for order acceptance in two-machine flow shops, Int. J. Product. Econ. 141 (2013) 14–23.
[105] B. Wu, C. Qian, W. Ni, S. Fan, Hybrid harmony search and artificial bee colony algorithm for global optimization problems, Comp. Math. Appl. 64 (2012) 2621–2634.
[106] Z. Xian, J. Xie, Y. Wang, Representative artificial bee colony algorithms: a survey, in: Proceedings of 2nd International Conference on Logistics, Informatics and Service Science, 2012, pp. 1419–1424.
[107] W.-L. Xiang, M.-Q. An, An efficient and robust artificial bee colony algorithm for numerical optimization, Comp. Operat. Res. 0 (2013) 1256–1265.
[108] C. Xu, H. Duan, Artificial bee colony (ABC) optimized edge potential function (EPF) approach to target recognition for low-altitude aircraft, Patt. Recog. Lett. 31 (2010) 1759–1772.
[109] C. Xu, H. Duan, F. Liu, Chaotic artificial bee colony approach to Uninhabited Combat Air Vehicle (UCAV) path planning, Aerosp. Sci. Technol. 14 (2010) 535–541.
[110] X. Yan, Y. Zhu, W. Zou, L. Wang, A new approach for data clustering using hybrid artificial bee colony algorithm, Neurocomputing 97 (2012) 241–250.
[111] W.-C. Yeh, T.-J. Hsieh, Solving reliability redundancy allocation problems using an artificial bee colony algorithm, Comp. Operat. Res. 38 (2011) 1465–1473.
[112] A.R. Yildiz, Optimization of cutting parameters in multi-pass turning using artificial bee colony-based approach, Inform. Sci. 220 (2013) 399–407.
[113] C. Zhang, D. Ouyang, J. Ning, An artificial bee colony approach for clustering, Exp. Syst. Appl. 37 (2010) 4761–4767.
[114] H. Zhang, Y. Zhu, W. Zou, X. Yan, A hybrid multi-objective artificial bee colony algorithm for burdening optimization of copper strip production, Appl. Math. Model. 36 (2012) 2578–2591.
[115] Z. Zhang, J. Lin, Y. Shi, Application of artificial bee colony algorithm to maximum likelihood DOA Estimation, J. Bionic Eng. 10 (2013) 100–109.
[116] R. Zhang, S. Song, C. Wu, A hybrid artificial bee colony algorithm for the job shop scheduling problem, Int. J. Product. Econ. 141 (2013) 167–178.
[117] W. Zhang, N. Wang, S. Yang, Hybrid artificial bee colony algorithm for parameter estimation of proton exchange membrane fuel cell, Int. J. Hydro. Energy 38 (2013) 5796–5806.
[118] G. Zhu, S. Kwong, Gbest-guided artificial bee colony algorithm for numerical function optimization, Appl. Math. Comput. 217 (2010) 3166–3173.
[119] K. Ziarati, R. Akbari, V. Zeighami, On the performance of bee algorithms for resource-constrained project scheduling problem, Appl. Soft Comput. 11 (2011) 3720–3733.