Neural Computing and Applications
https://doi.org/10.1007/s00521-019-04533-x

ENGINEERING APPLICATIONS OF NEURAL NETWORKS 2018

A honeybees-inspired heuristic algorithm for numerical optimisation

Muharrem Düğenci[1] · Mehmet Emin Aydin[2]

Received: 11 January 2019 / Accepted: 5 October 2019

Abstract
Swarm intelligence is all about developing collective behaviours to solve complex, ill-structured and large-scale problems. Efficiency in collective behaviours depends on how to harmonise the individual contributors so that a complementary collective effort can be achieved to offer a useful solution. The main points in organising the harmony remain as managing the diversification and intensification actions appropriately, where the efficiency of collective behaviours depends on blending these two actions appropriately. In this paper, a hybrid bee algorithm is presented, which harmonises bee operators of two mainstream well-known swarm intelligence algorithms inspired of natural honeybee colonies. The parent algorithms have been overviewed with many respects, strengths and weaknesses are identified, first, and the hybrid version has been proposed, next. The efficiency of the hybrid algorithm is demonstrated in comparison with the parent algorithms in solving two types of numerical optimisation problems; (1) a set of well-known functional optimisation benchmark problems and (2) optimising the weights of a set of artificial neural network models trained for medical classification benchmark problems. The experimental results demonstrate the outperforming success of the proposed hybrid algorithm in comparison with two original/parent bee algorithms in solving both types of numerical optimisation benchmarks.

Keywords Swarm intelligence · Numerical optimisation · Bee-inspired algorithms · Diversification and intensification · Training feed-forward neural networks

1 Introduction

Collective intelligence is one of the approaches commonly found useful for problem-solving in the modern times. This is motivated by the fact that collective effort pays off better than individual effort in the real life and has been bought in by computer science researchers and implemented in various problem-solving approaches. Swarm intelligence is known to be a family of collective problem-solving frameworks such as ant colony optimisation, particle swarm optimisation and artificial bee colonies imposing use of population of solutions, here-forth called swarm of individuals. The main benefit of population-based meta-heuristic approaches, particularly swarm intelligence algorithms, is that the algorithms nicely harmonise local search activities around various neighbourhoods without guaranteeing to cover the whole search space. Therein, the local search is devised, to a certain extent, to intensify the search and enhancement activities are facilitated to diversify the search for managing change among neighbourhoods.

Diversification plays a crucial role to arrange visiting unseen regions of the search space as efficiently as possible so that the search effort for optimum solution would not be trapped in locality and be able to keep enough energy for further search. On the other hand, intensification is required to make the search algorithm as focus as possible so that any particular local region would not remain under-examined. A balanced/well-featured search algorithm harmonises the actions required for both diversification and intensification, which is required for effective and efficient search. In fact, individual solution-driven search algorithms conduct more intensified search, while population-driven algorithms are more diversifying by their nature.

✉ Mehmet Emin Aydin
mehmet.aydin@uwe.ac.uk

Muharrem Düğenci
mdugenci@karabuk.edu.tr

[1] Department of Industrial Engineering, Karabuk University, Karabuk, Turkey

[2] Department of Computer Science and Creative Technologies, University of the West of England, Bristol, UK

Published online: 16 October 2019

Hence, swarm intelligence algorithms do require intensification of the search in local regions as they deliver very diverse search by default. This feature applies to the algorithms developed inspired of the collective behaviour of honeybees, where a number of bees algorithm (BA) [25] and artificial bee colony (ABC) [12] variants have been redesigned to manage/handle such a harmony among various search actions. In fact, variants of both BA and ABC algorithms are devised mainly for these purposes, where a variety of difficult problems can be solved with a more generalised search that well-featured with diverse and focus search activities, adequately [4, 22, 34]. However, it is observed that the existing mechanics of BA and ABC algorithms do not sufficiently support intensification, which drives us to further investigations.

The main aim of this paper is to propose a hybridising framework to merge the strong search capabilities of both BA and ABC to seek for higher efficiency in problem-solving. Both algorithms, BA and ABC, have been reviewed first to identify the strengths and weaknesses with respect to intensification in search. Next, revisions for better granularity level in search are proposed for removing the weakness identified in both of BA and ABC, accordingly, where the revised versions demonstrated that they offer better granularity level in search steps to facilitate further intensification. Finally, a hybridisation approach is introduced to merge the strengths of both into a new algorithm for further intensification and improved diversification. In this respect, the contributions of this paper can be listed as follows:

- Reviewing the properties of both standard BA and ABC algorithms with respect to diversification and intensification of the search.
- Improve the intensification properties of both algorithms with appropriate revisions,
- Device a hybridisation approach/framework in which the strengths of the algorithms are aligned to help improve diversification of the search.

The rest of this paper is organised as follows: Sect. 2 introduces swarm intelligence algorithms inspired of natural honeybee colonies, Sect. 3 introduces related works and identifies how this work differs from the related ones, while Sect. 4 introduces the proposed approach including revisions envisaged for the parent algorithms (BA and ABC) and the proposed new hybrid algorithm. Section 5 includes a comprehensive experimental study to test the performance of all algorithms, while Sect. 6 provides the conclusions.

2 Swarm Intelligence and honeybees-inspired algorithms

Swarm intelligence is one of the cutting edge soft computing technologies used for solving various optimisation problems in more efficient ways. This is because the approaches and frameworks proposed are adaptive, flexible and robust in the way that the algorithms handle the problems using various techniques of collectivism. Collective effort by each individual within the swarms is managed by sharing the information regarding search activities towards the common targets. That helps divers the search by its nature.

Figure 1 sketches the search idea metaphorically delivered by honeybees, where a typical function, which has multiple optima points, is explored through for the optimum points, while the search conducted within a neighbourhood by a team of bees is also spotted out to reflect the idea and its implementation. This metaphoric idea has been borrowed from honeybees and their way of collective search by mainly two algorithms, as explained above. Both of the algorithms are detailed in the following subsections.

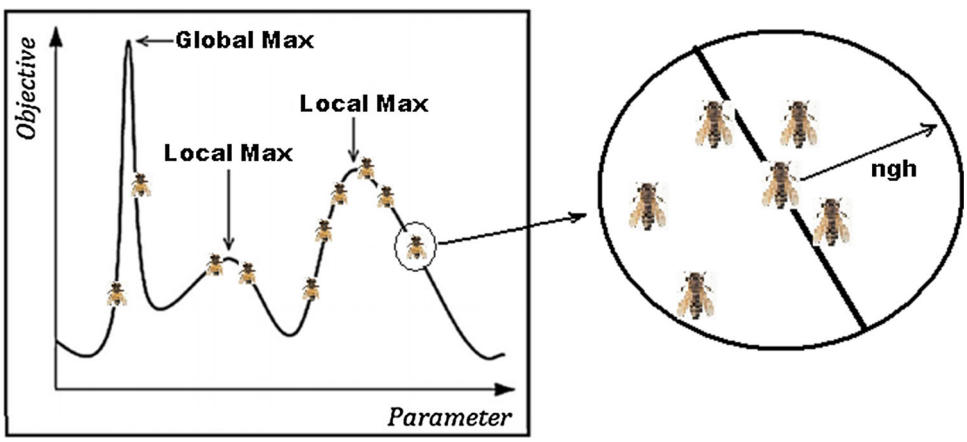

Fig. 1 Search metaphorically delivered by honeybees

2.1 Bees algorithm (BA)

Bees algorithm is one of the mainstream swarm intelligence algorithms inspired of natural honeybee colonies introduced by Pham and his associates [25, 33]. It looks like a typical population-based optimisation algorithm in which solutions are considered as individual bees and are evaluated based on the fitness function-like evaluation rules, which are usually of simple objective functions. The algorithm imposes a search procedure inspired of food/nectar exploration process by honeybees within the nature. An elitist approach is followed to search through the most fruitful regions of the search space so that the optimum or a useful near-optimum can be found as fast as possible without causing further complexities. This algorithm has not only been used for solving numerical optimisation problems, e.g. benchmark functions, neural network training, etc. but also been considered for solving a variety of combinatorial optimisation problems [19, 34].

Let $\mathcal{X}$ be a population of solutions, which is considered to be the bee colony, and let $\mathbf{x}_i = \{x_{i,j} | i = 1, \ldots, N; j = 1, \ldots, D\}$ represent solution i within this population, which is also called an individual bee as a member of colony/swarm, where N denotes the size of bee colony, $N = |\mathcal{X}|$, and D is the size of input set. Suppose also that $F(\mathbf{x}_i)$ is a function defined ($f_i : \mathbf{x}_i \rightarrow \mathbb{R}$) to measure the quality/fitness of solution $\mathbf{x}_i$. The initial population/swarm of bees is generated using $x_{i,j} = x_{i,\min} + \rho * (x_{i,\max} - x_{i,\min})$, where $x_{i,j}$ is a data point for jth input of $\mathbf{x}_i$ solution initialised to be a random value within the range of $[x_{i,\min}, x_{i,\max}]$ normalised with the random number of ρ.

After generating the initial swarm, each individual bee is evaluated using the fitness function created based on the main objective of the problem tackled. The bees are, then, classified based on their performance/fitness; a set of elite bees, $\mathcal{E}$, where $\mathbf{y}_e \in \mathcal{E}$ and $\mathbf{y}_e = \{y_{e,j} | e = 1, \ldots, |\mathcal{E}|; j \in D\}$, a set of moderate search bees, $\mathcal{M}$, where $\mathbf{z}_m \in \mathcal{M}$ and $\mathbf{z}_m = \{z_{m,j} | m = 1, \ldots, |\mathcal{M}|; j \in D\}$, and a set of employee bees, $\mathcal{I}$, where $\mathbf{x}_k \in \mathcal{I}$ and $\mathbf{x}_k = \{x_{k,j} | k = 1, \ldots, N - (|\mathcal{E}| - |\mathcal{M}|); j \in D\}$. Therefore, $\mathcal{X} = \mathcal{E} \bigcup \mathcal{M} \bigcup \mathcal{I}$, where $N_e = |\mathcal{E}|, N_m = |\mathcal{M}|$, and $|\mathcal{I}| = N - (|\mathcal{E}| - |\mathcal{M}|)$. In order for moving to the next generation, $\mathcal{E} \in \mathcal{X}$ and $\mathcal{M} \in \mathcal{X}$ are preserved ahead and the rest of the population, which are employee bees, are scraped.

The next step of producing the next generation is to deploy supporting bees, which are not created initially, but later while breeding the new generation in order for supporting each elite, $\mathbf{y}_e$, and moderate, $\mathbf{z}_m$, bees within the neighbourhood of each. Each individual elite bee, $\mathbf{y}_e$, is supported with a team of bees to further explore within its neighbourhood. This extends the size of elite bees' set from N_e to $N_e \times \beta$ while the moderate search bees are also supported in the same way, but with different predefined supporting teams of bees. This also increases the size of moderate bee set to $N_m \times \gamma$, where β and γ are predetermined fixed numbers, to identify how many bees to be recruited in the neighbourhood of each elite and moderate bee, respectively. The supporting bees, which are deployed in the search regions of elite and moderate bees, are created with the rule of $x_{i,j} = x_{i,j} + \rho * \delta$, where ρ is a random number generated within the range of $(-1, 1)$ and δ is another predetermined fixed value to be the step size of change in any input of a solution/a bee. This rule can be specified for each of the bee types as follows: (1) supporting bees for elite bees with $y_{i,j} = y_{i,j} + \rho * \delta$, while for moderate search bees with $z_{i,j} = z_{i,j} + \rho * \delta$. Once support teams of bees are deployed within corresponding search regions, the majority of the swarm of the next generation becomes complete. The remaining small portion of the new colony (around 20%) is randomly generated in the way of the initial random population.

Once the elite bees, moderate search bees and the others are identified, and the predefined number of supporting bees is sent to each neighbourhood of both types of these bees. This procedure is repeated until a predetermined stopping criterion is met.

2.2 Artificial bee colony algorithm (ABC)

Artificial Bee Colony (ABC) is another very popular swarm intelligence algorithm developed inspiring of the collective behaviours of honeybee colonies. Karaboga [12] has first initiated this algorithm to solve numerical optimisation problems [14] and then extended the applications with various combinatorial optimisation ones [16, 24]. ABC imposes considering individual solutions as sources of food (nectar) for honey bees, and searching around each solution is named to be collective activities of various types of bees. There are mainly two bee types envisaged; employed and unemployed, where unemployed bees can be in two types; Onlooker and Scout bees. A set of search activities is organised around the nectar sources by recruiting various types of bees in various configurations.

Let $\overrightarrow{x_i}$ be a solution, defined as an input vector of $\mathcal{D}$ size considered as a source of nectar. A population of $\mathcal{N}$ different sources are initially generated using $x_{ij} = x_{i,\min} + \rho * (x_{i,\max} - x_{i.\min})$, where $i = 1, .., \mathcal{N}; j = 1, .., \mathcal{D}; x_{i,\min}$ and $x_{i.\max}$ are minimum and maximum values of ith input of $\overrightarrow{x_i}$ source. Once the whole population of the sources is generated completely, and then, the nectar level of each source is determined to identify the quality of each, which becomes the fitness value of each solution. Following this step, the employed bees start operating on each source to search for sources with better quality using

$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{ik})$, where $\overrightarrow{v_i}$ is the new source found following an interaction between $\overrightarrow{x_i}$ and $\overrightarrow{x_k}$, which is a randomly selected known source among many within the colony of the generation. The difference calculated between the two sources is normalised with a randomly generated $\phi_{ij} \in (-a, a)$. After the new source identified, a decision is made whether or not to adopt the new source to replace the original one. The ultimate fitness of a typical source decision is calculated using:

$$F(\overrightarrow{x_i}) = \begin{cases} \dfrac{1}{1+f(\overrightarrow{x_i})}, & f(\overrightarrow{x_i}) \geq 0 \\ 1 + |f(x)|, & \text{otherwise} \end{cases}$$

Onlooker bees start operations following complete by employed bees. The main role of onlooker bees is to monitor the employed bees and taking the search further using a probabilistic process, where a probability of p_i is calculated using $p_i = \frac{F(\overrightarrow{x_i})}{\sum_{i=1}^{\mathcal{N}} F(\overrightarrow{x_i})}$ for each individual candidate source and a roulette-wheel selection rule is used to choose a solution for further explorations. The neighbourhood of a chosen source is conducted with $v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{ik})$ similar to employed bees. A small size memory is associated with each further investigated source if any progress is achieved or not. A counter for each investigated source is created and run up to a predefined threshold. If no progress accomplished, then the source is removed from the colony.

Scout bees, then, follow onlookers to diversify the colony, randomly inserting new sources using the initial rule of source generation: $x_{ij} = x_{i,\min} + \rho * (x_{i,\max} - x_{i.\min})$. This generational process is repeated until a certain level of satisfaction is reached. As part of the above-mentioned process, each individual solution/source can be included in the next generation via either of the following cases: (1) a source would remain without any change, (2) an employed bee would generate a new solution, (3) an onlooker bee may bring a new solution, (4) a source would be found by both employed or onlooker bees, or (5) an investigated source is replaced with a new source as a result of non-improvement decision. It is a fact that each solution is attempted for improvement at least once, which would be investigated with more attempts if its fitness remains high.

2.3 Relevant works

A variety of works have considered honey bee-inspired algorithms for problem-solving purposes, attempting with a variety of the problems including numerical optimisation benchmarks. The benchmark functions provided in Table 1 are commonly used functions for performance evaluation purposes. Kong et al. [22] use numerical optimisation functions to test the success of their hybrid algorithm, which uses an orthogonal initialisation. However, 60-dimensional problems, at most, are considered, and mostly underperformed in comparison with our results. Hacibeyoglu et al. [9] produced the results for the same set of benchmarks, but did not tabulate them so as to be compared with our results, where the level of dimensions is clearly lower than our case. Kiran and Gunduz [21] have borrowed and embedded a crossover operator from genetic algorithms to solve the numerical benchmark functions, where they considered 50 dimensions at most and the results seem to be very fluctuating as standard deviations are higher than mean statistics in some cases. Karaboga and Basturk [14] have first published their ABC algorithm with the same set of benchmark numerical functions solving them in relatively lower dimensions. However, Karaboga and Akay [13] have presented the success of the same algorithm providing extensive details of their comparative study, where they solved around 50 benchmarks

Table 1 Benchmark functions commonly used for performance evaluation of algorithms

Test Function	Input range		Equation number
Sphere	(− 100,100)	$f_1(x) = \sum_{i=1}^{D} x_i^2$	(6)
Rosenbrock	(− 2.048,2.048)	$f_2(x) = \sum_{i=1}^{D} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	(7)
Ackley	(− 32.768,32.768)	$f_3(x) = -20e^{-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}} - e^{\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right)} + 20 - e$	(8)
Griewank	(− 600,600)	$f_4(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}) + 1$	(9)
Rastrigin	(− 5.12,5.12)	$f_5(x) = \sum_{i=1}^{D}\left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	(10)
Schwefel	(− 500,500)	$f_6(x) = \sum_{i=1}^{D}(x_i - \sin(\sqrt{\lvert x_i \rvert}))$	(11)

including our benchmarking problems. The algorithms seem implemented very successfully for dimensions up to 30, noting that many other ABC implementations could not hit that level of success. Kiran and Findik [20] present a directed/adaptive ABC algorithm solving the benchmarks with 10, 30 and 50 dimensions, where our results are competitive with them at this level while we solve the problems for much higher dimensions.

On the other hand, Pham et al. [25] introduce their BAs algorithm with solving the same set of benchmarking numerical function with rather very lower dimensions, e.g. up to $D = 10$. Likewise, Yuce et al. [33] have also attempted to solve a number of benchmark functions including those considered in this study with up to 10 dimensions at most. Hussein et al. [11] have improved BAs algorithm with a pre-processing of particular initialisation algorithm and gained better results than both of [25] and [33] in solving the same set of benchmarks with up to 60 dimensions, where our results apparently outperform for all functions except Schwefel.

A number of other metaheuristic and/or swarm intelligence algorithms have also attempted to solve the benchmarks we considered, recently. Based on the relevance that the same functions have been attempted, it is decided to include these studies in the review to help grasp the difficulty of the problems attended. Gong et al. [7], Liu et al. [23], Zhao and Tang [35] and Xin et al. [32] have published their results for the benchmark problems up to 30 dimensions using different variants of particle swarm optimisation, differential evolution and a particular algorithm so-called monkey algorithm. Their results are apparently either not better than, or remain competitive with ours. Likewise, Han et al. [10], Rahmani and Yusof [28] and Alam et al. [1, 3] have introduced their approaches for 30 and 50 dimensions, where our approach usually outperforms them or remain competitive. None of the following references have attempted dimensions larger than 50, but, the majority of them have only considered up to 30, while our approach outperform them in major [2, 5, 8, 17, 26]. These studies have mostly compared their result with those produced by Suganthan et al. [31] in which a comprehensive study is extensively reported on solving a number of numerical optimisation benchmarks.

The algorithms reviewed and cited above propose various approaches to improve the performance of bee-inspired algorithms either (1) with extending the algorithms through pre-processing or (2) introducing additional computational activities into any stage of the algorithms or (3) embedding local search procedures into the algorithms. To the best knowledge of the authors, none of the works related to this paper reviews the capabilities of the two mainstream honeybees-inspired algorithms (BA and ABC) and not propose hybridising the two to improve the fundamental properties of the search. This paper introduces an approach/framework to hybridise BA and ABC algorithms for improving the performances via further intensification and diversification in the search process, which can be made transferrable to any variant of the algorithms in this kind.

3 Proposed approach

The above-mentioned honeybees-inspired algorithms have been examined with respect to the balance between diversification and intensification of the search, and few ideas have been put together for the purpose of improving their performances in solving numerical optimisation problems.

Following the structural and experimental analysis, both of the algorithms, BE and ABC, introduced above have been found with strengths and weaknesses with respect to diversification and intensification of search process. Both ABC and BA algorithms include freshly generated random solutions into the new generations to a certain level, where diversification of the search is achieved in this way. In addition, BA algorithm intensifies the search on fruitful sources, where further search attempts are organised around highly fitted sources/solutions, which helps intensification further, while ABC uses memory-like mechanism to let scout bees intensify their search around certain sources for a number of attempts until it is understood that the source is dried out. Once a source is dried out, it is deleted from the population.

On the other hand, both algorithms conduct search with few shortcomings, which have been considered, in this study, as the grounds of improvement to enhance the capabilities of above-mentioned bee-inspired algorithms. In this regard, BA algorithm uses a parameter to normalise the step size, so-called environmental/neighbourhood factor and denoted with δ, in the previous sections. It is set to a fixed value at the initialisation stage and kept as it is to the end of the search. This makes granularity of the step size coarse-grained in approximating the optimum value, which drifts intensification away, and prevents the search to reach the optimum in most of the time. Another weakness of BA algorithm is the diminishing probability of having random solutions within the population, especially during the late stages of the search. This can escalate to disabling diversification at later stages. In the case of ABC, the weaknesses arise in two points; (1) the sources taken out of population are evaluated not based on the fitness, but, improvability, which can cause disregard of useful solutions, and (2) in addition to this, some useful and very well-improved solutions can be decommissioned from the population since their improvability is reduced to *0*

according to the criteria adopted. Both of these weaknesses can drive the algorithm towards very unfertile region of search space.

In the following two subsections, ideas are considered and discussed to enhance the capabilities of both of the bee-inspired algorithms following the above-mentioned structural assessments. These will be used as bee operators in the hybrid algorithm.

3.1 Intensification in bees algorithm (rBA)

The main revision envisaged for BA to improve intensification, based on the shortcomings discussed above, is to make step sizes more fine-grained. The granularity of step size can be adjusted through the fixed-valued (constant) δ in the update rule, $z_{i,j} = z_{i,j} + \rho * \delta$, where $z_{i,j}$ is a single dimension of a complete solution and ρ is a random number within the range of $[-1, 1]$. It is important to note that the parameter of δ is selected at the configuration stage the algorithms under consideration and kept constant through out of the entire search process in original BA. The parameter of δ corresponds to shrinking constant (sc) in [33], and to the parameter of h, so-called incremental size in [34] as part of BA variant embedded with a hill-climbing local search procedure in which the step size is calculated with slope-angle approach. The embedded hill-climbing procedure is not sufficiently explained to realise how the step size is calculated through the gradient and the incremental size parameter in [34].

This constant-valued parameter,δ as declared in original BA, manages the granularity level and leaves the approach rather coarse-grained, which causes the step size not to be easily adjustable in finer precisions and can take much longer time to approximate. In order to avoid this shortcoming, the update rule is revised as follows: $z_{i,j} = z_{i,j} + \rho * \delta * z_{i,j}$, where δ is made to be a rate within the range of [0,1], and can be adaptive, too. Therefore, the new step size calculated with $\delta * z_{i,j}$ will be more adjustable and proportional to the range of $(z_{\min}, z_{\max})$ with which the algorithm can approximate much faster than before, and more preciously. The update rule is applied to all types of bees recruited as part of the algorithm, while the rest of the algorithm remains as original.

3.2 Intensification in ABC algorithm (rABC)

In order to ease the difficulties in approximation using standard ABC, following the shortcomings identified and discussed above, two revisions have been envisaged to achieve ABC improvement; (1) one is to collect all results from all employed and onlooker bees and then apply roulette-wheel selection instead of original practice, and (2) the other revision is to adopt a rank-based selection rule for the next generation, where 25% of top ranked solution from entire existing solution set, $\mathcal{N} + \mathcal{E}$, where $\mathcal{N}$ denotes original bee colony and $\mathcal{E}$ is the number of generated solutions. The first revision widens the candidate set for roulette-wheel operator to select a more fruitful solution to enhance search while the second devises a greedier approach towards the top best solutions.

3.3 Proposed hybrid algorithm (Hybrid)

This algorithm attempts at merging the strengths of both of above-mentioned mainstream bee algorithms for better search performance. It is a hybrid algorithm in which the hybridisation process is managed based on the framework of BA algorithm with implementing not only the bee operators from BA algorithm but also all other above-mentioned algorithms. The details of proposed Hybrid algorithm are provided in Fig. 2. As can be seen, the main structure of the algorithm is inherited from BE algorithm and devised differently using a set of rules given below as the set of equations; Eqs. (1)–(5). Each rule is a bee operator used by one of the above-mentioned algorithms. The Hybrid algorithm implies a systematic harmony/reuse of Eqs. (1)–(5), adopted as bee operators used for generating new solutions/bees as well as neighbours for the existing elite and fit bees. We should note that Eq. (1) is used for generating the initial swarm and independent bees exploring for better nectar sources while Eqs. (2)–(5) are used to send supporting bees around each elite bee.

$$\overrightarrow{x_i} = \overrightarrow{x}_{\min} + \overrightarrow{\rho} * \left(\overrightarrow{x}_{\max} - \overrightarrow{x}_{\min}\right) \quad \text{for } \forall i \in N \tag{1}$$

$$\overrightarrow{x_i} = \overrightarrow{x_i} + \overrightarrow{\rho} * \delta \quad \text{for } \forall i \in N \quad \text{and} \quad \delta \in \mathbb{R} \tag{2}$$

$$\overrightarrow{v_i} = \overrightarrow{x_i} + \overrightarrow{\phi_i}\left(\overrightarrow{x_i} - \overrightarrow{x_k}\right) \quad k \in N \quad \text{and} \quad \text{for } \forall i \in N \tag{3}$$

$$\overrightarrow{x_i} = \overrightarrow{x_i} + \overrightarrow{\rho} * \delta * \overrightarrow{x_i} \quad \text{for } \forall i \in N \quad \text{and} \quad \delta \in [0, 1] \tag{4}$$

$$\overrightarrow{v_i} = \overrightarrow{x_i} + \overrightarrow{\phi_i}\left(\overrightarrow{x_i} - \overrightarrow{x_k}\right) \quad k \in \{Q_1 ofN\} \quad \text{and} \quad \text{for } \forall i \in N \tag{5}$$

1. *Initialise* the Population/Swarm, $\mathcal{X}$, using **Eq**: (**1**);
2. *Evaluate* the individuals/bees;
3. *Classify* bees into Elite, $\mathcal{E}$ and Moderate, $\mathcal{M}$;
4. *Conduct* search:
   a. Around each elite bee, $\mathbf{y_e} \in \mathcal{E}$, with $\boldsymbol{\beta}$ more bees, *using* randomly selected rule from **Eq: (1)-(5);**
   b. Around each moderate bee, $\mathbf{z_m} \in \mathcal{M}$ with $\boldsymbol{\gamma}$ more bees, *using* randomly selected rule from **Eq: (2)-(5)**;
   c. Generate $|\mathcal{J}|$ independent bees, using **Eq**: (**1**);
5. *Evaluate* the complete swarm;
6. *Identify* the best solution so far, $\mathbf{x}^*$;
7. If stopping criterion not met, *Go* to Step 3;
8. *Stop*.

Fig. 2 Pseudocode for Hybrid bee algorithm

Equations (2), (3), (4) and (5) are the neighbourhood rules used, respectively, by the ordinary BA algorithm, the revised BA algorithm (rBA), ABC and revised ABC algorithms (rABC) to explore around a local nectar source, which means a local region of the search space in optimisation context. The hybrid algorithm randomly selects one of these rules to generate a neighbouring solution of a particular elite solution, each time, to complete up β supporting bees for each elite so that $N_e \times \beta$ bees can be placed in the new generation. The moderate search bees randomly use Eq. (2) to (5) for generating their neighbouring solutions to complete γ number of supporting bees so as to place $N_m \times \gamma$ solutions in the next swarm while the independent bees explore with Eq. (1) for further generations of randomly searched nectars. The rest of algorithmic mechanics of this hybrid algorithm works in the same way as the ordinary bee algorithm does until a certain satisfactory level is achieved as indicated in the pseudocode provided in Fig. 2.

It is important to note that this frame work can be flexible with additional and different types of bee operators to be employed as part of Step 4 for populating the new generation of the swarm. In the following experimental study, it is demonstrated that random selection of bee operators among the set of operators/rules given with Eqs. (1)–(5) can pay off better in comparison with the original algorithms take part of Hybrid. It is also possible that a bespoke selection policy can be adopted instead of randomly selecting the operators for generating bees and the neighbouring solution.

4 Experimental evaluations

The following section introduces a major experimental study to demonstrate the performance of above-mentioned well-known bee algorithms and the revisions envisaged to enhance the capabilities via performances.

4.1 Functional optimisation

The first bit of this experimental evaluation is made with functional optimisation benchmarks. The functions considered for testing purposes are multidimensional functions, which can also be considered as many-dimensional functions, where the tests have been conducted over their 5, 30, 60, 100 and 150 dimensions. The reason to opt with these dimensions is that the literature [1, 14, 22, 33] reports solving these problems with similar dimensions, where 100 and 150 dimensions are exercised first time in this study. Two of the functions are known as uni-model (labelled as (6) and (7) in Table 1), which means that they have only single optimum points, while the other four are multi-model functions meaning that they can have multiple optimum points. These are all well-known and challenging benchmark functions used to test optimisation algorithms across the literature of this field. An extensive study on a number of numerical optimisation benchmarks including those considered below is reported in [31].

The parametric design details of the algorithms are provided in Table 2, where the parameters of the main three algorithms tabulated. It should be noted that the revised versions of both BA and ABC algorithms, rBA and rABC, have the same parametric values as the original ones since they suggest more procedural rather than parametric changes. As a matter of fact, the neighbourhood structures of the algorithms, which is also a procedural difference, are indicated as follows: all algorithms use fixed-sized local neighbourhood, while BA has a rank-based random selection, ABC uses roulette-wheel selection and, in fact, Hybrid adopts both in a systematic use. In addition, Hybrid algorithm selects mate bees from top quartile when operating with revised ABC.

The initial populations/swarms have been randomly generated with rather different seeds, and the experimentations for each function with each variant of algorithm are repeated 30 times. The experimentation has started with rather lower dimensions and gradually increased up in due course. The first set of experiments has been carried out with a dimension of 5 and 30 for all benchmarks, just to be in-line with the existing literature. The results are tabulated in Table 3, where the results are recorded in two statistics, mean and standard deviation, with the five algorithms against each benchmark function. As part of investigating the number of iterations, preliminary experiments with 200 and 1000 iterations have been conducted as plotted in Fig. 3a, b. As suggested, the averaged differences between the optimum and the results found are plotted for each algorithm. Further performance details of the algorithms are provided accordingly within Table 3 with 5000 iterations for both 5-D and 30-D benchmark problems.

Table 2 Parametric details of the algorithmic configurations

		BA	ABC	Hybrid
Population/swarm size	N	100	100	100
Number of elite bees	N_e	5	–	5
Number of moderate search bees	N_m	20	–	20
Number of bees supporting elite bees	β	40		40
Number of Independent bees	$\lvert\mathcal{I}\rvert$	30	–	30
Neighbourhood factor	δ	0.1	–	$0.1 * \overrightarrow{x_i}$
Non-improvability threshold	L	–	200	–

Table 3 Experimental results by all five bee algorithms with 5000 iterations for 5-D and 30-D benchmark functions

Functions	Optimum	BA		rBA		ABC		rABC		HYBRID	
		Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
D = 5											
Sphere	0.000	**0.000**	**0.000**	**0.000**	**0.000**	0.012	0.005	**0.000**	**0.000**	**0.000**	**0.000**
Rosenbrock	0.000	**0.000**	**0.000**	0.376	1.058	1.157	0.568	2.766	1.043	0.705	0.270
Ackley	0.000	**0.000**	**0.000**	**0.000**	**0.000**	0.322	0.104	**0.000**	**0.000**	**0.000**	**0.000**
Griewank	0.000	0.027	0.008	0.124	0.089	0.864	0.376	**0.000**	**0.000**	**0.010**	**0.024**
Rastrigin	0.000	**0.000**	**0.000**	2.791	1.601	2.762	0.512	**0.000**	**0.000**	**0.000**	**0.000**
Schwefel	− 2094.915	− **2094.914**	**0.000**	− 1874.53	150.786	− 2085.44	32.131	− 1793.648	63.512	− 1941.62	118.433
D = 30											
Sphere	0.000	**0.000**	**0.000**	**0.000**	**0.000**	4.119	0.302	**0.000**	**0.000**	**0.000**	**0.000**
Rosenbrock	0.000	21.626	0.086	**19.414**	**5.418**	517.168	50.104	28.710	0.304	25.244	0.709
Ackley	0.000	**0.000**	**0.000**	0.159	0.587	11.500	7.092	**0.000**	**0.000**	**0.000**	**0.000**
Griewank	0.000	**0.000**	**0.000**	0.012	0.014	0.217	0.022	**0.000**	**0.000**	**0.000**	**0.000**
Rastrigin	0.000	201.545	10.485	95.042	63.927	222.550	10.042	**0.000**	**0.000**	0.239	1.170
Schwefel	− 12,569.49	− 5485.77	291.939	− **8590.56**	**682.295**	− **8698.14**	**268.407**	− 4540.758	328.703	− 7729.98	739.943

The bold values indicate significance of the outperforming results

Figure 3a, b present the differences between known optimum values and the achieved results averaged overall benchmark problems categorised dimensions and the number of iterations taken. Figure 3a, b include the results for 200 iterations. All three figures clearly suggest that Hybrid algorithm outperforms all others and its approximation goes closer to 0. On the other hand, revised algorithms perform better than the original algorithms in the same overall point of view, where rBA remains as the first runner algorithm after Hybrid. It is also observed that ABC performs much better when dimension is lower, but performs not as good as the other rivals with growing dimension. However, rABC, the revised ABC, is one of the competitors with Hybrid regardless of the growing dimensions.

As indicated above, the experimental results reported in Table 3 are the performance of five algorithms for each of the benchmark problems.

Sphere function is easily solved by almost all algorithms with five dimensions over 5000 iterations, while the function with 30 dimensions becomes a bit challenging, where all four algorithms except ABC find the optimum, and Hybrid hits the optimum with an ignorable difference after 200 iterations, while the other significantly remain distant. After 1000 iterations, BA and ABC only stay struggling, but the other three solve the problem with exact solution. ABC only remains a little bit distant after 5000 iterations while the rest solve it exactly.

Rosenbrock function is one of two functions found challenging in this research. None of the algorithms have found the optimum while the best with five dimensions is by BA and with 30 dimensions by rABC. Algorithms' performances improve with an increase the number of iterations to 5000; however, the optimum is still not achieved, although BA and rABC perform better for 5 and 30 dimensions, respectively, and Hybrid always follows as the second best.

The best approximation for Ackley function is made by Hybrid, while BA and rABC remain competing with Hybrid to reach the exact optimum; however, both remain in a very ignorable distance. It is observed that Hybrid performs the best after 200 iterations for 5 dimensions cases, but BA and rABC compete with Hybrid in other cases of 30 dimensions.

Griewank function is best approximated by rABC and Hybrid algorithms, even as early as 200 iterations in both dimensions of 5 and 30. The other algorithms approximate to the optimum level after 5000 iterations, noting that rBA and ABC remain a little bit distant to the optimum. rABC solves Rastrigin function to optimum in both dimensions (5 and 30) after 200 iterations, while the other algorithms struggle to approximate even after 5000 iterations. It is important to indicate that this function is attended by Kong

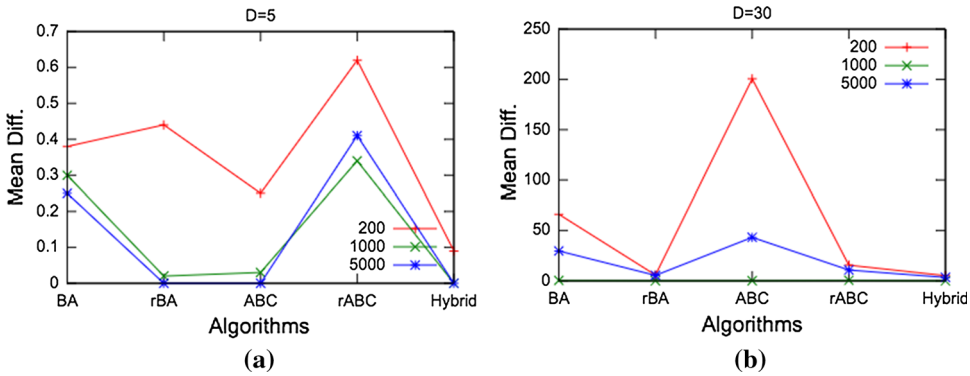

Fig. 3 Differences between the optimum and the results found by the algorithms for **a** $D = 5$, **b** $D = 30$

et al. [22] with 5 and 10 dimensions only. Schwefel function remains as the most challenging benchmark since BA and ABC solve it with 5 dimensions after 200 iterations, but none of the algorithms managed solving the problems to the optimum with higher dimensions even after 5000 iterations, where initial swarms/populations escalate to very different results, each time.

Table 4 presents the performances of all five algorithms for 60-dimensional benchmarks after 5000 iterations, where it is clear that both ordinary BA and ABC algorithms remain very underperforming in comparison with the revised versions and the hybrid algorithm, although their performance improves with more iterations as indicated in the bottom (5000 iteration cases) section of the table. On the other hand, rBA, rABC and Hybrid approximate to the optimum in four functions after 1000 iterations, while struggle in solving Rosenbrock and Schwefel functions, despite that their performance improves in Rosenbrock function. These results indicate that Schwefel function clearly requires far more attention to better approximate. It is also notable that the results after 5000 iterations deviate from the optimum with very ignorable level.

The results in Tables 5 and 6 include the performance of BA, ABC and the Hybrid algorithms only, since beyond $D = 60$, the revised algorithms seem to underperform following their original versions. However, Hybrid remains competitive and solving four functions to optimum out of the six benchmarks. Tables 5 and 6 presents the results of Hybrid in comparison with original BA and ABC for $D = 100$ and $D = 150$ running the algorithms for 5000 iterations. Hybrid solves Sphere, Ackley, Griewank and Rastrigin functions to optimum for all dimensions including 60-D, 100-D and 150-D. However, the algorithms, BA, ABC and the revised versions of these two, remain behind this level of achievement with growing dimensions. The algorithms other than Hybrid seem falling in a local optimum around 20 while solving Ackley function for dimensions of 100 and 150. Rosenbrock function is the second challenging benchmark among all, where the approximation of Hybrid remains just below 100 for 100-D and below 150 for 150-D cases. Clearly, Schwefel function is the most challenging one since the approximation of all algorithms stays far apart of the expected optimum. This hints that Schwefel function requires particular attention. Both BA and ABC improve their performances with an increase in iteration numbers, but the level of improvement, apparently, remains rather weaker. That means that the approximation of both algorithms approach to the ultimate level of achievement, and beyond this level of iterations a significant improvement is not expected.

Both Tables 5 and 6 include the results gained after 10,000 iterations for both large-dimensional cases, i.e. 100-D and/or 150-D. The results do not include any surprise on that beyond a certain number of iterations; the achievement is not improving significantly, where Hybrid evidently out performs both of its competitors. In fact, both of BA and ABC algorithms approximate very roughly, while Hybrid approaches to the optimum values except the cases of Rosenbrock and Schwefel functions.

In order to visualise and overview the overall performances, Fig. 4a–f presents the overall performance indication of BA, ABC and Hybrid algorithms for the dimensions of 100 and 150. Similar to the previous cases depicted in Fig. 3a, b, the results of all the algorithms for all functions have been further processed to calculate the differences between the optima and the results produced and then averaged accordingly. Figure 4a, c, e plot the averaged differences for 100-D, while Fig. 4b, d, f plots for 150-D. It is observed that, as suggested in Fig. 4a–f, BE significantly underperforms in comparison with both ABC and Hybrid, while ABC does runs up after Hybrid, where Hybrid significantly outperforms both rivals. The performance of both ABC and Hybrid improves with an increase in the number of iterations, while BA does not improve significantly with the growing number of iterations for both cases of D-100 and D-150. As seen, Fig. 4c, d presents the performance excluding Schwefel function, which is the most challenging one, while Fig. 4e, f shows the performance excluding both challenging functions: Rosenbrock and Schwefel. It is observed that Hybrid solves all

Table 4 Experimental results by all five bee algorithms with 5000 iterations for 60-D benchmarks

Functions	Optimum	BA		rBA		ABC		rABC		HYBRID	
		Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
$D = 60$											
Sphere	0.00	8755.69	1074.61	**0.00**	**0.00**	20.23	1.09	**0.00**	**0.00**	**0.00**	**0.00**
Rosenbrock	0.00	2077.09	305.53	260.85	493.85	3524.30	268.87	58.63	0.41	**56.60**	**5.28**
Ackley	0.00	13.02	0.55	2.34	1.51	19.76	0.11	**0.00**	**0.00**	**0.00**	**0.00**
Griewank	0.00	78.99	10.74	0.02	0.09	0.56	0.04	**0.00**	**0.00**	**0.00**	**0.00**
Rastrigin	0.00	614.33	17.39	305.97	162.41	620.64	22.16	**0.00**	**0.00**	**0.73**	**3.57**
Schwefel	− 25,139.00	− 7563.69	390.96	− 14,886.10	3842.62	− **15,021.71**	**369.85**	− 6299.01	357.44	− **16,202.95**	**1100.75**

The bold values indicate significance of the outperforming results

problems to optimum except Schwefel, while ABC approximates better excluding Rosenbrock and Schwefel.

4.2 Conclusions for functional optimisation

Functional optimisation provided in the previous subsection helps demonstrate how to implement the proposed Hybrid algorithm for functional optimisation benchmarks and prove that the proposed algorithm outperforms two mainstream honeybees-inspired algorithms. Tables 3, 4, 5 and 6 present the comparative results in both mean and standard deviation statistics, where the outperforming achievement of Hybrid can be observed. The standard deviations show the steadiness of the mean results, where the results by Hybrid and ABC seem not much fluctuating while BA results fluctuate, meaning that BA is not fit enough to tackle the algorithms. The overall performance of all three algorithms is plotted in Fig. 4a–f to visualise the outperforming success by Hybrid. Also, the marginal achievement is plotted excluding the challenging benchmark problems, where BA does not improve, but both Hybrid and ABC improve with exclusion of challenging benchmarks as well as with the growing number of iterations.

4.3 Neural network training

In this section, the Hybrid algorithm is tested with another numerical optimisation case, which is used for optimising the weights of feed-forward neural network models used in classification problems.

Training NN models with search algorithms is not a very new concept, but is under consideration for further improvements with more powerful algorithms. Both ABC and BA algorithms are, as mentioned above, relatively new swarm intelligence algorithms attract further research efforts as it proves success in various areas of applications, especially in numerical optimisation problems. Dugenci and Aydin [6] have recently demonstrated that the algorithm is capable of solving high-dimensioned complex numerical functions. On this basis, BA-based training is devised for optimising the set of connection weights of feed-forward NN models, so that they can predict and classify more precisely and successfully [18, 29, 30]. In order to achieve this, first an ANN model should be configured, and then, the set of weights can be retrieved. A typical feed-forward NN model with two inputs, few hidden nodes and one output is displayed in Fig. 5, where the connection weights labelled with honeybees and the nodes are signified with honeycombs. The total number of connections is the size of a solution state, which is subjected to optimisation process.

Table 5 Experimental results for 100-D and 150-D cases with iterations of 5000

	Optimum	BA		ABC		HYBRID	
		Mean	SD	Mean	SD	Mean	SD
$D = 100$							
Sphere	0.00	82666.223	7305.573	62.287	2.926	**0.000**	**0.000**
Rosenbrock	0.00	11196.241	1082.235	12562.895	793.275	**96.391**	**0.926**
Ackley	0.00	18.980	0.216	20.237	0.059	**0.000**	**0.000**
Griewank	0.00	736.722	48.323	0.926	0.021	**0.000**	**0.000**
Rastrigin	0.00	1239.407	21.171	1198.287	32.504	**0.000**	**0.000**
Schwefel	− 41898.29	− 9764.026	432.783	− **24552.068**	**716.374**	− **25045.133**	**1529.901**
$D = 150$							
Sphere	0.00	212934.022	13039.447	158.882	7.217	**0.000**	**0.000**
Rosenbrock	0.00	29945.963	1879.910	28818.579	1189.044	**146.507**	**0.906**
Ackley	0.00	20.627	0.055	20.477	0.055	**0.000**	**0.000**
Griewank	0.00	1943.258	94.105	31.436	8.770	**0.000**	**0.000**
Rastrigin	0.00	2067.049	35.547	1979.598	35.096	**0.663**	**3.249**
Schwefel	− 62847.44	− 12162.446	527.191	− **36306.258**	**769.194**	− **37402.093**	**1285.558**

The bold values indicate significance of the outperforming results

Table 6 Experimental results for 100-D and 150-D cases with iterations of 10000

D = 100	Optimum	BA		ABC		HYBRID	
		Mean	SD	Mean	SD	Mean	SD
Sphere	0.00	73741.840	7178.359	55.730	2.159	**0.000**	**0.000**
Rosenbrock	0.00	10614.572	1015.702	11224.417	701.626	**96.059**	**1.046**
Ackley	0.00	18.759	0.230	20.179	0.071	**0.000**	**0.000**
Griewank	0.00	665.434	55.750	0.757	0.028	**0.000**	**0.000**
Rastrigin	0.00	1219.130	34.059	1172.028	40.949	**0.000**	**0.000**
Schwefel	− 41898.29	− 10078.371	513.046	− **24766.810**	**625.801**	− **25436.107**	**1410.357**
D = 150	Optimum	BA		ABC		HYBRID	
		Mean	SD	Mean	SD	Mean	SD
Sphere	0.00	202242.026	13243.606	135.488	5.650	**0.000**	**0.000**
Rosenbrock	0.00	28221.044	2056.926	26233.178	1316.636	**146.622**	**0.812**
Ackley	0.00	20.542	0.102	20.457	0.033	**0.000**	**0.000**
Griewank	0.00	1814.381	91.636	0.945	0.017	**0.000**	**0.000**
Rastrigin	0.00	2044.323	28.582	1940.707	48.985	**0.286**	**1.401**
Schwefel	− 62847.44	− 12233.336	530.825	− **36806.223**	**982.819**	− **37797.404**	**1353.187**

The bold values indicate significance of the outperforming results

Given the circumstances, a feed-forward NN model with one hidden layer has I number of input nodes, H number of hidden nodes and O number of output nodes. The number of connections constituted between input and hidden layer is $(I+1)H$, while the number of connections required between hidden layer and output layer is $(H+1)O$, where in each level 1 bias node is also considered as part of feed-forward ANN to facilitate learning more smoothly. The set of weights between I and H is $w_i^{I-H} = \left\{ w_{i,j}^{I-H} | j = 1, \ldots, (I+1)H \right\}$, while the weight set for connections between H and O is $w_i^{H-O} = \left\{ w_{i,j}^{H-O} | j = 1, \ldots, (H+1)O \right\}$. The ultimate set of weights is $w_i = \left\{ w_i^{I-H} \cup w_i^{H-O} \right\}$ with the size of $|w_i| = (I+1)H + (H+1)O$. For example, given the model in Fig. 5, there are two input neurons,three hidden

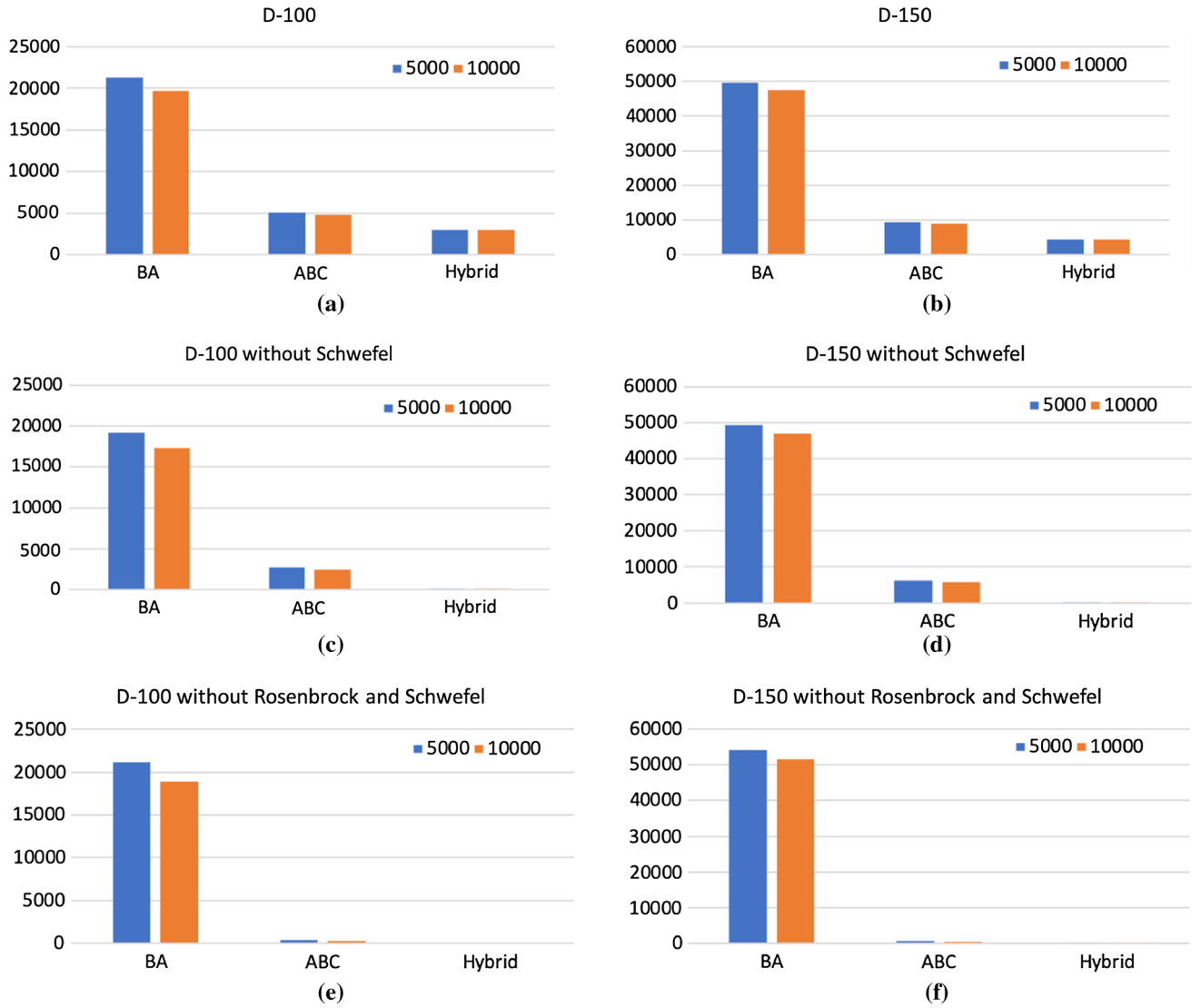

Fig. 4 Averaged overall achievements by BA, ABC and Hybrid; **a** all comparative results in D-100 cases, **b** all comparative results in D-150 cases, **c** comparative results for benchmarks except Schwefel in D-100 cases, **d** comparative results for benchmarks except Schwefel in D-150 cases, **e** comparative results for benchmarks except Rosenbrock and Schwefel in D-100 cases and **f** comparative results for benchmarks except Rosenbrock and Schwefel in D-150 cases

layer neurons and 1 output neuron with bias nodes in each layer of hidden and output level; hence, the total number of connection weights is $(2 + 1) * 3 + (3 + 1) * 1 = 12$. Therefore, a typical bee will represent the whole NN with a vector of 12 weights including nine w_i^{I-H} and four w_i^{H-O} values. The total number of weights will change accordingly if any of I or H or O changes.

The benchmark problems have been taken from UCI data collection, one of well-known medical data collection used for research purposes [27]. Table 7 tabulates the results including all data types with the sizes of both training and test sets alongside the configurations of neural nets, where the models have multiple and many outputs; each corresponds to one particular class to be identified. The NN model is expected to produce one output much higher than the others so as to consider that the highest output to be the class-identified subject to given input data. Table 7 introduces the data sets with the overall data size in the second column, which is divided into the training and test sets given in the third and fourth columns. The fifth column in the table provides the configurations of feed-forward NN models set up to classify corresponding data sets indicated the rows. The form of data is introduced in <I–H–O> , where I, H and O stand for the number of input nodes, hidden nodes and output nodes, respectively. The last two columns show the comparative results in CEP

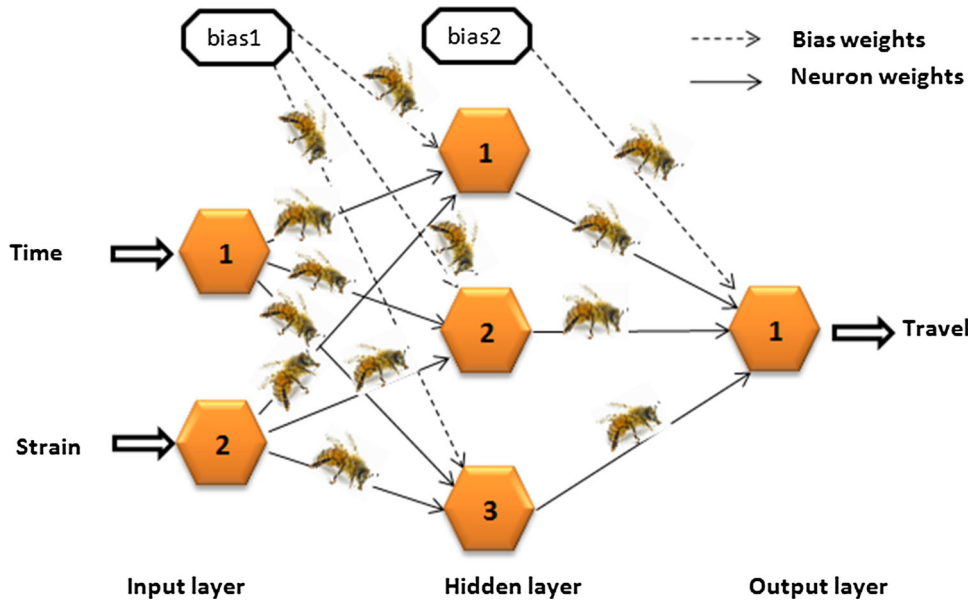

Fig. 5 Feed-forward NN model configured and trained with Hybrid BA

Table 7 Experimental results by Hybrid and ABC algorithms in mean of CEP and standard deviation in parenthesis for various medical data sets used for training and testing neural network models

Type of data sets	Size of data sets			NN-config	Results (CEP)	
	Data	Training	Test		Hybrid	ABC
Cancer	699	525	174	9-5-2	1.14	1.14
					(0.41)	(0.17)
Diabetes	768	576	192	8-6-2	20.31	24.84
					(5.31)	(2.65)
Heart	920	690	230	35-5-2	18.69	19.48
					(0.30)	(0.51)
Card	690	518	172	51-6-2	13.37	13.53
					(0.56)	(1.30)
Gene	3190	2300	890	120-6-3	16.86	29.50
					(2.49)	(5.37)
Glass	214	161	53	9-6-6	23.25	45.62
					(3.85)	(9.57)
Horse	368	300	68	58-6-2	16.15	28.63
					(2.35)	(7.85)
Soybean	683	513	170	82-6-19	21.17	38.63
					(2.13)	(6.45)
Thyroid	7200	3772	3428	21-6-3	2.66	6.95
					(0.39)	(1.23)

measure, mean and standard deviation in parenthesis, where the column labelled "Hybrid" presents the results by Hybrid algorithm while the one labelled with "ABC" contains the results by ABC algorithm which is taken from Karaboga and Ozturk [15].

As mentioned above, the base ABC algorithm considered in this comparison is introduced by Karaboga and Ozturk [15]. The performances are measured in an index called classification error percentage (CEP), which is the percentage of misclassification patterns over the total number of test patterns. The configurations of each NN model developed per data set are the same as those in the Karaboga and Ozturk [15]. As suggested in table (Table 7), the majority of the results by Hybrid are significantly better than those produced by ABC with respect to mean and standard deviation statistics, where the top four problem cases are slightly better with Hybrid, but last five cases are significantly better as suggested by rather lower standard

deviations and much higher CEP means. This demonstrates that the diversification and intensification operations handled as the result of hybridisation pay off and prove the superiority of Hybrid algorithm.

5 Conclusions

In this paper, a hybrid bee-inspired algorithm is proposed with a comprehensive performance investigation through two numerical optimisation problem types; (1) functional optimisation benchmarks and (2) ANN training through optimising the weights of connection links of feed-forward NN models. Although a number of variants of both mainstream bee-inspired algorithms, BA and ABC, have been developed and used for a number of problem-solving purposes, there was not attempt to merge the strengths of both mainstream algorithms so that more efficient and robust problem-solving can be achieved. This paper presents a novel bee algorithm which hybridises both BA and ABC algorithms for better performance.

The properties of both algorithms are first reviewed to identify the strengths and weaknesses, and then, remedies are identified to cure the weaknesses, where revised versions of both algorithms, rBA and rABC, are developed. Afterwards, a framework is devised to harmonise and reuse the bee operators of all original and revised algorithms into the search process. It is demonstrated that the existing and improved capabilities of BA and ABC algorithms with respect to diversification and intensification are pulled in the Hybrid so that the strengths of all participating algorithms can be merged in Hybrid framework. The Hybrid framework has been comparatively tested with (1) solving very high-dimensional numerical optimisation benchmarks and (2) optimising the weights of feed-forward NN models develop for classification purposes. The experimental results clearly suggested that revised versions of both BA and ABC (rBA and rABC) improve the performance by large and more importantly the proposed Hybrid algorithm significantly performs better in comparisons with the original and revised versions of both algorithms, BA and ABC.

This achievement is attained with better harmony induced in the hybrid algorithm, where both of rBA and rABC provided better intensification and randomly and systematically use of operators helped achieve improved diversification. This does not limit the Hybrid framework to the set of equations proposed and used neither rules out any selection policy other than random selection. It means that further studies are needed to test the Hybrid framework with variety of bee operators and selection policies to identify the best configurations bespoke to the problems under investigation. In addition, the hybrid framework requires to be further investigated for combinatorial optimisation problems as the next step of this research.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Alam MS, Islam MM, Murase K (2012). Artificial bee colony algorithm with improved explorations for numerical function optimizationn. In: Intelligent data engineering and automated learning-IDEAL 2012. LNCS 7435, pp 1-8. Springer Berlin Heidelberg, Natal, Brazil
2. Alam MS, Islam MM, Yao X (2011) Recurring two-stage evolutionary programming: a novel approach for numerical optimizaiton. IEEE Trans Syst Man Cybern Part B Cybern 41(5):1352–1365
3. Alam MS, Islam MM, Yao X, Murase K (2012) Diversity guided evolutionary programming: a novel approach for continuous optimization. Appl Soft Comput 12:1693–1707
4. Aydin ME (2012) Coordinating metaheuristic agents with swarm intelligence. J Intell Manuf 23(4):991–999
5. Dogan B, Olmez T (2015) A new metaheuristics for numerical function optimization: vortex search algorithm. Inf Sci 293:125–145
6. Dugenci M, Aydin ME (2018) Diversifying search in bee algorithms for numerical optimisation. Lect Notes Artif Intell 11056:132–144
7. Gong W, Cai Z, Jia L, Li H (2011) A generalized hybrid generation scheme of differential evolution for global numerical optimization. Int J Comput Intell Appl 10:35–65
8. Guo L, Wang G-G, Gandomi AH, Alavi AH, Duan H (2014) A new improved krill herd algorithm for global numerical optimization. Neurocomputing 138:392–402
9. Hacıbeyoğlu M, Koçer B, Arslan A (2012) Transfer learning for artificial bee colony algorithm to optimize numerical functions. In: International conference on computer engineering and network security (ICCENS'2012), Dubai
10. Han M, Liu C, Xing J (2014) An evolutionary membrane algorithm for global optimization problems. Inf Sci 276:219–241
11. Hussein WA, Sahran S, Abdullah SN (2014) Patch-Levy-based initialization algorithm for bees algorithm. Appl Soft Comput 23:104–121
12. Karaboga D (2005) An idea based on honey bee swarm for numerical optimisation. Computer Engineering Department, Erciyes University, Kayseri
13. Karaboga D, Akay B (2009) A comparative study of artificial bee colony algorithm. Appl Math Comput 214:108–132
14. 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

15. Karaboga D, Ozturk C (2009) Neural networks tarining by artificial bee colony algorithm on pattern classification. Neural Network World 19:279–292
16. Karaboga D, Gorkemli B, Ozturk C, Karaboga N (2014) A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artif Intell Rev 42(1):21–57
17. Kashan AH (2015) A new metaheuristic for optimization: optics inspired optimization (OIO). Comput Oper Res 55:99–125
18. Keskin TE, Düğenci M, Kaçaroğlu F (2015) Prediction of water pollution sources using artificial neural networks in the study areas of Sivas, Karabük and Bartın (Turkey). Environ Earth Sci 73(9):5333–5347
19. Keskin TE, Dugenci M, Kacaroglu F (2014) Prediction of water pollution using artificial neural networks in the study areas of Sivas. Environmental Earth Science, Karabuk and Bartin (Turkey)
20. Kiran MS, Findik O (2015) A directed artificial bee algorithm. Appl Soft Comput 26:454–462
21. Kiran MS, Gunduz M (2012) A novel artificial bee colony-based algorithm for solving the numerical optimization problems. Int J Innov Comput Inf Control 8(9):6107–6121
22. Kong X, Liu S, Ang Z, Yong L (2012) Hybrid artificial bee colony algorithm for global numerical optimization. J Comput Inf Syst 8(6):2367–2374
23. Liu Y, Niu B, Luo Y (2015) Hybrid learning particle swarm optimizer with genetic disturbance. Neurocomputing 151:1237–1247
24. Pan QK, Tasgetiren MF, Suganthan PN, Chua TJ (2011) A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. Inf Sci 181(12):2455–2468
25. Pham DT, Ghanberzadeh A, Koc E, Otri S, Rahim S, Zaidi M (2006) The bees algorithm – a novel tool for complex optimisation. In: Pham DT, Eldukhri EE, Soroka AJ (eds) Intelligent production machines and systems. Springer, Berlin
26. Piotrowski AP (2015) Regardin the rankings of optimization heuristics based on artificially constructed functions. Inf Sci 297:191–201
27. Prechelt L (1994) PROBEN1 – a set of benchmarks and benchmarking rules for neural network training algorithms. Fakultat fur Informatik, Universitat Karlsruhe, Karlsruhe, Germany
28. Rahmani R, Yusof R (2014) A new simple, fast and efficient algorithm for global optimization over continuous search-space problems: radial movement optimization. Appl Math Comput 248:287–300
29. Sarangi P, Sahu A, Panda M (2014) Training a feed-forward neural network using artificial bee colony with back-propagation algorithm. Intell Comput Network Inform Adv Intell Syst Comput 243:511–519
30. Senyigit E, Dugenci M, Aydin ME, Zeydan M (2013) Heuristic-based neural networks for stochastic dynamic lot sizing problem. Appl Soft Comput 13(3):1331–1338
31. Suganthan,PN, Hansen N, Liang JJ, Deb K, Chen Y-P, Auger A et al (2005). Problem definitions and evaluation criteria for CEC 2005 Special Session on real-parameter optimization. Nanyang Technological University, Computer Science, Singapore. KanGAL, IIT, Kanpur
32. Xin B, Chen J, Peng ZH, Pan F (2010) An adaptive hybrid optimizer based on particle swarm and differential evolution for global optimization. Inf Sci 53(5):980–989
33. Yuce B, Packianather MS, Mastrocinque E, Pham DT, Lambiase A (2013) Honey bees inspired optimization method: the bees algorithm. Insects 4(4):646–662
34. Yuce B, Pham DT, Packianather MS, Mastrocinque E (2015) An enhancement to the bees algorithm with slope angle computation and hill climbing algorithm and its applications on scheduling and continuous-type optimisation problem. Prod Manuf Res 3(1):3–19
35. Zhao R, Tang W (2008) Monkey algorithm for global numerical optimization. J Uncertain Syst 2(3):165–176