Diversifying search in Bee algorithms for numerical optimisation

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

[1] Karabuk University, Department of Industrial Engineering, Karabuk, Turkey, mdugenci@karabuk.edu.tr,
[2] University of the West of England, Department of Computer Science and Creative Technologies, Bristol, UK
mehmet.aydin@uwe.ac.uk

Abstract. Swarm intelligence offers useful instruments for developing collective behaviours to solve complex, ill-structured and large-scale problems. Efficiency in collective behaviours depends on how to harmonise the individual contributions so that a complementary collective effort can be achieved to offer a useful solution. The harmonisation helps blend diversification and intensification suitably towards efficient collective behaviours. In this study, two renown honeybees-inspired algorithms were analysed with respect to the balance of diversification and intensification and a hybrid algorithm is proposed to improve the efficiency accordingly. The proposed hybrid algorithm was tested with solving well-known highly dimensional numerical optimisation (benchmark) problems. Consequently, the proposed hybrid algorithm has demonstrated outperforming the two original bee algorithms in solving hard numerical optimisation benchmarks.

Keywords: swarm intelligence, numerical optimisation, Bee-inspired algorithms, diversification and intensification

1 Introduction

Swarm intelligence is known to be a family of approaches used for collective intelligence in problem solving. Swarm intelligence frameworks such as ant colony, particle swarm, artificial bee colonies algorithms impose use of population of solutions, here-forth called swarm of individuals. The main benefit of population-based metaheuristic 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 extend, for intensifying 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 remain under-examined. A balanced/well-featured search algorithm should harmonise the actions 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. 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 inspiring of honeybees, where a number of "bee algorithm (BA)" [24] and "artificial bee colony (ABC)" [13] variants have been proposed to manage/handle such a harmony among various search actions. In fact, various hybrid algorithms are devised mainly for this purposes, where a verity of difficult problems can be solved with a more generalised search that well-featured with diverse and focus search activities, adequately [21], [3] [30]. 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 study is investigate how to improve search in original bee-inspired algorithms through diversification and intensification. The properties in both BA and ABC algorithms are analysed first, and then efficiency through diversification and intensification is sought, next. Further details are provided in the following sections.

2 Relevant Works

Although the main aim of this study is not to solve the benchmark functions more precisely, it is worthwhile to mention that a variety of studies have considered the same benchmark functions considered in this study for testing the performance of their versions of BA and ABC algorithms. The work reported in [21] is inspiring, which imposes an orthogonal initialisation for their hybrid algorithm, where it uses 60 dimensional problems at most, and underperformed by large in comparison to our results. The results produced by [10] for the same set of benchmarks are not tabulated so as to be compared with our results, where the level of dimensions is clearly lower than ours. The authors of [19] 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 then mean statistics in some cases. The inventors of ABC [15] have first published their results with ABC algorithm for the same set of benchmark numerical functions solving them in relatively lower dimensions. However, [14] by the same research team presents the success of the same algorithm providing extensive details of their comparative study, where they solved around 50 benchmarks including our benchmarking problems. The algorithms seems implemented very successfully for dimensions up to 30 noting that many other ABC implementations could not hit that level of success. The authors of [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, [24] introduces their BAs algorithm with solving the same set of benchmarking numerical function with rather very lower dimensions, e.g. up to D=10. Likewise, [31] have also attempted to solve a number of benchmark functions including those considered in this study with up to 10 dimensions at most. The study by [12] has improved BAs algorithm with a pre-processing of particular initialisation algorithm and gained better results than both of [24] and [31] 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. The authors of [8], [22], [29] and [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, [11], [26] and [1] 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, 6, 9, 17, 25]). These studies have mostly compared their result with those produced by [28] in which a comprehensive study is extensively reported on solving a number of numerical optimisation benchmarks. Most recently, [4] and [5] have implemented bee-inspired algorithms for a number of functions including the above mentioned ones, but, considering dimensions under 100 while we consider dimensions of 150.

3 Honeybee-inspired Algorithms

3.1 Bees Algorithm (BA)

Bees algorithm is another mainstream swarm intelligence algorithm inspired of natural honeybee colonies introduced by Pham and his associates [24],[31]. 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. 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 complications. 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 [18],[30].

Let $\mathcal{X}$ be a population of solutions, which is considered as a bee colony and let $\mathbf{x}_i = \{x_{i,j} | i = 1, .., N; j = 1, .., D\}$ 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 the colony, $N = |\mathcal{X}|$ and D is the size of input

set. Suppose 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 j^{th} input of solution $\mathbf{x}_i$ 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 $\mathbf{x}_e \in \mathcal{E}$ and $\mathbf{x}_e = \{x_{e,j} | e = 1, .., |\mathcal{E}|; j \in D\}$ a set of moderate search bees, $\mathcal{M}$ where $\mathbf{x}_m \in \mathcal{M}$ and $\mathbf{x}_m = \{x_{m,j} | m = 1, .., |\mathcal{M}|; j \in D\}$ and a set of employee bees, $\mathcal{J}$ where $\mathbf{x}_k \in \mathcal{J}$ and $\mathbf{x}_k = \{x_{k,j} | k = 1, .., N - |\mathcal{E} - \mathcal{M}|; j \in D\}$. Therefore, $\mathcal{X} = \mathcal{E} \cup \mathcal{M} \cup \mathcal{J}$, where $N_e = |\mathcal{E}|$, $N_m = |\mathcal{M}|$ and $|\mathcal{J} = N - (|\mathcal{E}| - |\mathcal{M}|)$. 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 new generation imposes to create and deploy more supporting bees in the neighbourhood of each elite, $\mathbf{x}_e$ and moderate, $\mathbf{x}_m$, bees. Each individual elite bee, $\mathbf{x}_e$, is supported with a team of bees to further explore within its neighbourhood. This extends the set of elite bees from N_e to $N_e \times \beta$, while the moderate search bees, $\mathbf{x}_m$, are also supported in the same way, but with different predefined supporting team 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 ones, 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. 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. This procedure is repeated until a predetermined stopping criterion is met.

3.2 ABC algorithm

Artificial Bee Colony (ABC) is another very popular swarm intelligence algorithm developed inspiring of the collective behaviours of honeybee colonies. Karaboga [13] has first initiated this algorithm to solve numerical optimisation problems and extended with [15] and then applied to various combinatorial optimisation problems [16] [23]. 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 $\mathbf{x}_i$ be a solution, defined as an input vector of $\mathcal{N}$ size considered as a source of nectar. A population of different sources are initially generated using $x_{i,j} = x_{i,min} + \rho(x_{i,max} - x_{i,min})$, where $i = 1, .., \mathcal{N}; j = 1, .., \mathcal{D}$; $x_{i,min}$ and $x_{i,max}$ are minimum and maximum values of i^{th} input of $\mathbf{x}_i$ source. Once the whole population of the sources is generated completely, 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 $v_{i,j} = x_{i,j} + \phi(x_{i,j} - x_{i,k})$, where $\mathbf{v}_i$ is the new source found following an interaction between $\mathbf{x}_i$ and $\mathbf{k}_k$, 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 $\phi \in (-\alpha, \alpha)$. 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:

$$F(\mathbf{x}_i) = \begin{cases} \frac{1}{1+f(\mathbf{x}_i)} & f(\mathbf{x}_i) \geq 0 \\ 1 + |f(\mathbf{x}_i)| & \text{otherwise} \end{cases} \tag{1}$$

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 $p_i = \frac{F(\mathbf{x}_i)}{\sum_{i=1}^{N} F(\mathbf{x}_i)}$ for each individual candidate source and a roulette-wheel selection rule is used to make a choice of a solution for further explorations. The neighbourhood of a chosen source is conducted with $v_{i,j} = x_{i,j} + \phi(x_{i,j} - x_{i,k})$ 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:$x_{i,j} = x_{i,min} + \rho(x_{i,max} - x_{i,min})$. 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: (i) a source would remain without any change, (ii) an employed bee would generate a new solution, (iii) an onlooker bee may bring a new solution, (iv) a source would be found by both employed or onlooker bees, or (v) 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, would be investigated with more attempts if the its fitness remains high.

4 Algorithms Revised

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 the performances in solving numerical optimisation problems. Following structural and

experimental analysis, both of the algorithms 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 intensifies the search 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. Afterwards, that source 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 at the same value 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; (i) the sources taken out of population are evaluated not based on the fitness, but, improvability, which can lead to disregard the useful solutions, (ii) in addition, some useful and very well-improved solutions can be decommissioned from the population since their improvability is reduced to 0 according the criteria adopted. Both of these weaknesses can drive the algorithm towards very unfertile region of search space. In the following subsections both of the algorithms have been revised to re-arrange the diversification and intensification of the search.

4.1 Bees Algorithm Revised (rBA)

The main revision envisaged for BA based on the shortcomings discussed above is to make step-sizes more adjustable and fine-tuned. This is identified to deal with the fixed-valued (constant) δ within the update rule, $x_{i,j} = x_{i,j} + \rho\ \delta$, where $x_{i,j}$ is a single dimension of a complete solution and ρ is a random number within the range of [-1, 1]. The fixed-valued parameter, δ, makes the approach coarse-grained and causes the step-size of a move not being easily adjustable for finer precisions. That would take much longer time to approximate. In order to avoid this shortcoming the update rule is revised as follows: $x_{i,j} = x_{i,j} + \rho\ \delta\ x_{i,j}$, 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 $\delta\ x_{i,j}$ will be more adjustable and proportional to the range of (x_{min}, x_{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.

4.2 ABC Algorithm Revised (rABC)

Following the shortcomings discussed above, two revisions have been envisaged to improve standard ABC's performance; one is to collect all results from all employed and onlooker bees and then apply roulette-wheel selection instead of original practice, and 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.

4.3 Hybrid Bees Algorithm (Hybrid)

This hybridisation is managed based on the framework of BA algorithm with implementing not only bee operations from BA algorithm but also all other above-mentioned updating rules. This hybrid algorithm systematically harmonises/reuses the updating rules given with equations (2) - (6) for generating new solutions/bees as well as neighbours for the existing elite and fit bees, where equation (2) is used for generating the initial swarm and independently exploring for better nectar sources while equations (3) - (6) are used to send supporting bees around each elite bee. It is expected to help diversify search with use of more optional neighbourhood functions, systematically supplied into the algorithm. It helps intensify search within local region using finer-grained update rules.

$$\mathbf{x}_i = \mathbf{x}_{min} + \rho(\mathbf{x}_{max} - \mathbf{x}_{min}) \qquad \text{for} \quad \forall i \in N \tag{2}$$

$$\mathbf{x}_i = \mathbf{x}_i + \rho\ \delta \qquad \text{for} \quad \forall i \in N \quad \text{and} \quad \delta \in \mathbb{R} \tag{3}$$

$$\mathbf{v}_i = \mathbf{x}_i + \phi(\mathbf{x}_i - \mathbf{x}_k) \qquad \text{for} \quad \forall i, k \in N \tag{4}$$

$$\mathbf{x}_i = \mathbf{x}_i + \rho\ \delta\ \mathbf{x}_i \qquad \text{for} \quad \forall i \in N \text{and} \quad \delta \in [0, 1] \tag{5}$$

$$\mathbf{v}_i = \mathbf{x}_i + \phi(\mathbf{x}_i - \mathbf{x}_k) \qquad \text{for} \quad \forall i \in N \quad \text{and } k \in Q_1 \subset N \tag{6}$$

Equation (3), (4), (5), and (6) are the neighbourhood rules used, respectively, by the ordinary BA algorithm, the revised BA algorithm, ABC and revised ABC algorithms 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 supporting bees for each elite so that bees can be placed in the new generation. The moderate search bees use only equations (3) and (4) for generating their neighbouring solutions to complete number of supporting bees so as to place solutions in the next swarm while the independent bees explore with equation (2) 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.

5 Experimental Results

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. First of all, the

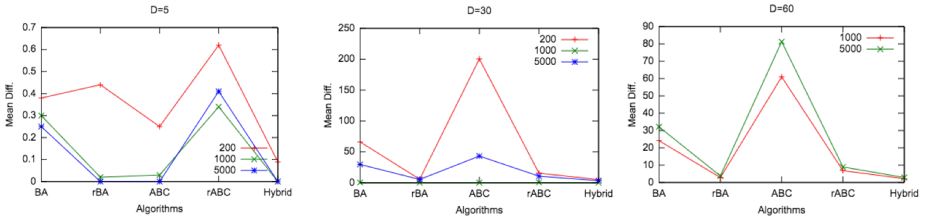
performance tests and analysis have been made using 6 numerical optimisation benchmarks, which are commonly used for the same purposes and given below. Obviously, all of these functions are multi-dimensional functions, which can also be considered as many-dimensional functions, where the tests have been conducted over their 5, 30, 60, 100 and 150 dimensions. The reason to opt with these dimensions is that the literature [21], [1, 2],[15],[31] reports solving these problems with similar dimensions, where 100 and 150 dimensions are exercised first time by this study. Two of the functions are know as uni-model, 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 [28].

1. Sphere function $f_1(x) = \sum_{i=1}^{D} x_i^2$
2. Rosenbrock function $f_2(x) = \sum_{i=1}^{D} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$
3. Ackley function $f_3(x) = -20e^{(\ -0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2})} - e^{(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi x_i))} + 20 - e$
4. Griewank function $f_4(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$
5. Rastrigin function $f_5(x) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$
6. Schwefel function $f_6(x) = \sum_{i=1}^{D} (x_i - \sin(\sqrt{|x_i|})$

Parametric design details configured as follows:- the swarm size for all algorithms is 100, number of elite, moderate, supporting and independent bees are set to 5, 20, 40 and 30, for BA and Hybrid algorithms respectively. Non-improvability threshold is fixed at 200 operations for ABC while the neighbourhood factor is 0.1 for BA and $0.1x$ for Hybrid. It can be observed that the revised versions of both BA and ABC algorithms have the same parametric values since the changes suggested is rather procedural than parametric. 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 Hybrid adopts both in a systematic use. In addition, Hybrid algorithm selects mate-bees from top quartile when operating with revised ABC.

The experimentation has been started with rather lower dimensions and gradually increased up in due course. The starting dimensions were 5 and 30 following the literature. All algorithms were run for 200, 1000 and 5000 iterations. Corresponding results are provided in [7]. Table 1 and 2 present the tabulated results for higher dimensions of the functions.

Table 1 presents the performances of all 5 algorithms for 60 dimensional benchmarks over 5000 iterations, where it is clear that both ordinary BA and ABC algorithms remain very underperforming in comparison to the revised versions and the hybrid algorithm. Meanwhile, rBA, rABC and Hybrid algorithms



(a) D=5 (b) D=30 (c) D=60

Fig. 1: Differences between optimum and results found by the algorithms

approximate to the optimum in four functions, but struggle in solving Rosenbrok and Schwefel functions, despite that their performance improves in Rosenbrok function. These results indicate that Schwefel function clearly requires far more attention to better approximate.

	D=60	**5000 Iterations**						
	Input Ranges	**Model**	**Optimum**	**BA**	**rBA**	**ABC**	**rABC**	**Hybrid**
Sphere	(-100, 100)	Uni	0.00	9.029	0.00	1.27E+04	3.90E-03	**2.80E-45**
Rosenbrock	(-2.048, 2.048)	Uni	0.00	792.867	58.882	2357.957	111.236	**54.965**
Ackley	(-32.768, 32.768)	Multi	0.00	19.512	3.24E-14	15.093	2.27E+00	**3.95E-14**
Griewank	(-600, 600)	Multi	0.00	792.867	58.882	2357.957	111.236	**0.00**
Rastrigin	(-5.12, 5.12)	Multi	0.00	792.867	58.882	2357.957	111.236	**0.00**
Schwefel	(-500, 500)	Multi	-25,138.974	-14,996.00	-12,579.00	-8,554.00	**-19,540.00**	-14,726.00

Table 1: Experimental results by all 5 bee algorithms with 5000 iterations for 60-D benchmarks

Figure (1a), (1b), (1c) plot the differences between known optimum values and the achieved results averaged over all benchmark problems against the number of iterations attempted, where the plots are categorised in dimensions. Figure (1a) and (1b) include the results for 200 iterations, while Figure (1c) does not include since 200 iterations remain too short for growing dimensions. 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 then the original algorithms in the same overall point of view, where rBA remains as the first runner after Hybrid. It is also observed that ABC performs much better when dimensions are lower. However, rABC, the revised ABC, is one of the competitors with Hybrid regardless of the growing dimensions.

The results in Table 2 include the performance of BA, ABC and the Hybrid algorithms only, since beyond D=60, the revised algorithms (rBA, and rABC) seem significantly outperforming their original versions. Hybrid remains competitive and solves four functions to optimum out of the six benchmarks. The presented results include the performance of Hybrid in comparison to original BA and ABC for dimensions of D=100 and D=150 running the algorithms over 5000 iterations. We should note that the results of 5000 iterations are only included in the tables due to the space constraints. Although not included in the tables, Hybrid solves Sphere, Ackley, Grienwak and Rastrigin functions to optimum after 1000 iterations for all dimensions including 60-D, 100-D and 150-D. However, the algorithms, BA, ABC and the revised versions of these two, re-

	D=100	**5000 Iterations**				
	Input Ranges	**Model**	**Optimum**	**BA**	**ABC**	**Hybrid**
Sphere	(-100, 100)	Uni	0.00	61.08	72,499.42	**0.00**
Rosenbrock	(-2.048, 2.048)	Uni	0.00	162.66	10,679.20	**92.52**
Ackley	(-32.768, 32.768)	Multi	0.00	19.49	19.05	**3.99E-15**
Griewank	(-600, 600)	Multi	0.00	0.90	768.32	**0.00**
Rastrigin	(-5.12, 5.12)	Multi	0.00	764.34	1,265.31	**0.00**
Schwefel	(-500, 500)	Multi	-41,898.29	-26,057.94	-9,255.25	**-26,537.00**
	D=150	**5000 Iterations**				
	Input Ranges	**Model**	**Optimum**	**BA**	**ABC**	**Hybrid**
Sphere	(-100, 100)	Uni	0.00	0.02	221,512.73	**1.10E-44**
Rosenbrock	(-2.048, 2.048)	Uni	0.00	142.89	32,620.37	**140.68**
Ackley	(-32.768, 32.768)	Multi	0.00	19.18	20.47	**1.47E-15**
Griewank	(-600, 600)	Multi	0.00	2,212.12	1,995.48	**0.00**
Rastrigin	(-5.12, 5.12)	Multi	0.00	868.31	2,106.19	**0.00**
Schwefel	(-500, 500)	Multi	-62,847.435	-35,174.20	-12,060.97	**-38,568.57**

Table 2: Experimental results by all 5 bee algorithms with 3 levels of iterations for 100-D and 150-D benchmark functions

main behind this level of achievement with growing dimensions. All algorithms except Hybrid seem falling in a local optimum around 20.0 while solving Ackley function for dimensions of 100 and 150. Rosenbrock function is the second challenging benchmark among all, where the approximation of Hybrid remain 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. A slight improvement is observed from the performance of both BA and ABC with growing iterations from 1000 to 5000, and 5000 to 10000, but, the level of improvement remains rather weaker. This suggests 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.

Figure 2a, 2b, 2c, and 2d, present the overall performance of BA, ABC and Hybrid algorithms indicated for the dimensions of 100 and 150. Similar to the previous cases depicted in Figure 1, 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. It is observed that, as suggested by Figure 2a, and 2c, ABC significantly underperforms in comparison to both BA and Hybrid, while Figure 2b, and 2d compare the performance of BA with Hybrid, where Hybrid significantly outperforms BA. The performance of BA improves with increasing number of iterations while Hybrid looks approximated to a steady state as suggested by Figure 2b and 2d noting that the averaged difference by Hybrid looks more substantial in 100-D cases than 150-D ones.

6 Conclusions

In this study, a comprehensive investigation is conducted to review the capabilities of Ba and ABC algorithms with respect to diversification and intensification in their search conduct. Both frameworks have been comparatively tested in solv-

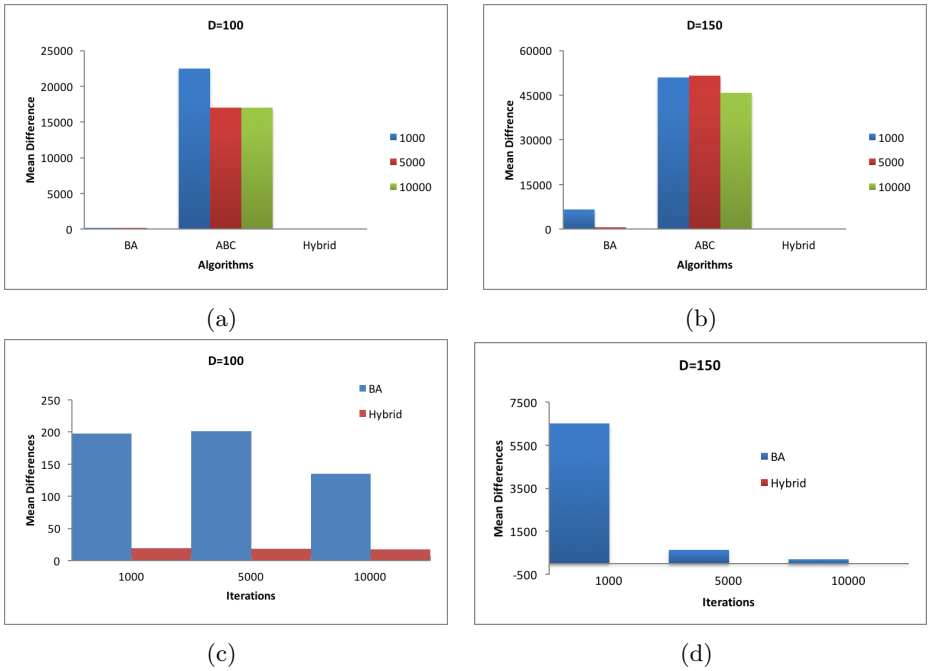


Fig. 2: Averaged overall achievements by BA, ABC and Hybrid; (a) comparative results for 100-D cases (b) comparative results by all three for 150-D cases, (c) comparative results by BA and Hybrid only, for 100-D cases, (d) comparative results by BA and Hybrid only, for 150-D cases

ing very high-dimensional numerical optimisation benchmarks. Revisions have been proposed for each algorithm for performance enhancement purposes. The results clearly suggested that revised versions of both BA and ABC (rBA and rABC) outperformed the original algorithms by large. Furthermore, a hybrid algorithm based on the original and their revised versions has been developed and tested, accordingly, resulting that the hybrid algorithm significantly improves the performance in solving very high-dimensional numerical optimisation benchmarks. 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 hybrid version is going to be tested for combinatorial optimisation problems as the next step of this research.

References

1. Alam, M. S., M. Md Islam, and K. Murase. Artificial bee colony algorithm with improved explorations for numerical function optimization. Intelligent Data Engineering and Automated Learning-IDEAL 2012. Natal, Brazil: Springer Berlin Heidelberg, 2012. 1-8.
2. Alam, M. S., Md. M. Islam, and X. Yao. Recurring two-stage evolutionary programming: A novel approach for numerical optimizaiton. IEEE Transactions on System, Man, and Cybernetics Part B: Cybernetics 41, no. 5 (2011): 1352-1365.
3. Aydin, M. E. Coordinating metaheuristic agents with swarm intelligence. Journal of Intelligent Manufacturing (Springer) 23, no. 4 (2012): 991-999.
4. Aydoğdu, I., Akin, A., and Saka, M. P., "Design optimization of real world steel space frames using artificial bee colony algorithm with Levy flight distribution." Advances in Engineering Software 92 (2016): 1-14.

5. Cui, L., Li, G., Zhu, Z., Lin, Q., Wen, Z., Lu, N., Wong, K.C. and Chen, J., "A novel artificial bee colony algorithm with an adaptive population size for numerical function optimization", Information Sciences, 414 (2017): 53-67.
6. Dogan, B., and T. Olmez. A new metaheuristics for numerical function optimization: Vortex search algorithm, Information Science 293 (2015): 125-145.
7. Düğenci, M. "Honeybees-inspired heuristic algorithms for numerical optimisation", arXiv preprint (2015), arXiv:1504.05766.
8. Gong, W., Z. Cai, L. Jia, and H. Li. A generalized hybrid generation scheme of differential evolution for global numerical optimization. International Journal of Computational Intelligence and Applications 10 (2011): 35-65.
9. Guo, L., G.-G. Wang, A. H. Gandomi, A. H. Alavi, and H. Duan. A new improved krill herd algorithm for global numerical optimization. Neurocomputing 138 (2014): 392-402.
10. Hacbeyolu, M., B. Koer, and A. Arslan. Transfer Learning for Artificial Bee Colony Algorithm to optimize numerical functions. International Conference on Computer Engineering and Network Security (ICCENS'2012). Dubai, 2012.
11. Han, M., C. Liu, and J. Xing. An evolutionary membrane algorithm for global optimization problems. Information Sciences 276 (2014): 219-241.
12. Hussein, W. A., S. Sahran, and S. N. H. S. Abdullah. Patch-Levy-based initialization algorithm for Bees Algorithm. . Applied Soft Computing 23 (2014): 104-121.
13. Karaboga, D. An idea based on honey bee swarm for numerical optimisation. Technical Report, Computer Engineering Department, Erciyes University, Kayseri, Turkey, 2005.
14. Karaboga, D., and B. Akay. A comparative study of artificial bee colony algorithm. Applied Mathematics and Computation 214 (2009): 108-132.
15. Karaboga, D., and B. Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of Global Optimization 39, no. 3 (2007): 459-471.
16. Karaboga, D., B. Gorkemli, C. Ozturk, and N Karaboga. A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artificial Intelligence Review 42, no. 1 (2014): 21-57.
17. Kashan, A. H. A new metaheuristic for optimization: Optics inspired optimization (OIO). Computers and Operations Research 55 (2015): 99-125.
18. Keskin, T. E., M. Dugenci, and F. Kacaroglu. Prediction of water pollution using artificial neural networks in the study areas of Sivas, Karabuk and Bartin (Turkey). Environmental Earth Science, 2014.
19. Kiran, M. S., and M. Gunduz. A novel artificial bee colony-based algorithm for solving the numerical optimization problems. International Journal of Innovative Computing, Information and Control 8, no. 9 (September 2012): 6107-6121.
20. Kiran, M. S., and O. Findik. A directed artificial bee algorithm. Applied Soft Computing 26 (2015): 454-462.
21. Kong, X., S. Liu, Z. Ang, and L. Yong. Hybrid Artificial Bee Colony Algorith for Global Numerical Optimization. Journal of Computational Information Systems 8, no. 6 (2012): 2367-2374.
22. Liu, Y., B. Niu, and Y. Luo. Hybrid learning particle swarm optimizer with genetic disturbance. Neurocomuting 151 (2015): 1237-1247.
23. Pan, Q. K., M. F. Tasgetiren, P. N. Suganthan, and T J. Chua. A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. Information Sciences 181, no. 12 (2011): 2455-2468.
24. Pham, D. T., A. Ghanberzadeh, E. Koc, S. Otri, S. Rahim, and M. Zaidi. The bees algorithm - Anovel tool for complex optimisation. Intelligent Production Machines and Systems, 2006.
25. Piotrowski, A. P. Regardin the rankings of optimization heuristics based on artificially constructed functions. Information Sciences 297 (2015): 191-201.
26. Rahmani, R., and R. Yusof. A new simple, fast and efficient algorithm for global optimization over continuous search-space problems:Radial Movement Optimization. Applied Mathematics and Computation 248 (2014): 287-300.
27. Senyigit, E., M. Dugenci, M. E. Aydin, and M. Zeydan. Heuristic-based neural networks for stochastic dynamic lot sizing problem. Applied Soft Computing (Elsevier) 13, no. 3 (2013): 1331-1338.
28. Suganthan, P. N., et al. Problem definitions and evaluation criteria for CEC 2005 Special Session on real-parameter optimization. Technical Report, Computer Science, Nanyang Technological University, Singapore: KanGAL, IIT, Kanpur, 2005.
29. Xin, B., J. Chen, Z. H. Peng, and F Pan. An adaptive hybrid optimizer based on particle swarm and differential evolution for global optimization. Information Science (Science China) 53, no. 5 (May 2010): 980-989.
30. Yuce, B., D. T. Pham, M. S. Packianather, and E. Mastrocinque. An enhancement to the Bees Algorithm with slope angle computation and Hill Climbing Algorithm and its applications on scheduling and continuous-type optimisation problem. Production & Manufacturing Research 3, no. 1 (2015): 3-19.
31. Yuce, B., M. S. Packianather, E. Mastrocinque, D. T. Pham, and A. Lambiase. Honey bees inspired optimization method: the Bees Algorithm. Insects 4, no. 4 (2013): 646-662.
32. Zhao, R., and W. Tang. Monkey algorithm for global numerical optimization. Journal of Uncertain Systems 2, no. 3 (2008): 165-176.