The Journal of Supercomputing (2020) 76:7026–7046
https://doi.org/10.1007/s11227-019-03127-7

Robust parallel hybrid artificial bee colony algorithms for the multi-dimensional numerical optimization

Tansel Dokeroglu[1] · **Selen Pehlivan**[1,2] · **Bilgin Avenoglu**[1]

Published online: 9 January 2020

Abstract
This study proposes a set of new robust parallel hybrid metaheuristic algorithms based on artificial bee colony (ABC) and teaching learning-based optimization (TLBO) for the multi-dimensional numerical problems. The best practices of ABC and TLBO are implemented to provide robust algorithms on a distributed memory computation environment using MPI libraries. Island parallel versions of the proposed hybrid algorithm are observed to obtain much better results than those of sequential versions. Parallel pseudorandom number generators are used to provide diverse solution candidates to prevent stagnation into local optima. The performances of the proposed hybrid algorithms are compared with eight different metaheuristics algorithms of particle swarm optimization, differential evolution variants, ABC variants and evolutionary algorithm. The empirical results show that the new hybrid parallel algorithms are scalable and the best performing algorithms when compared to the state-of-the-art metaheuristics.

Keywords Hybrid · Artificial bee colony · TLBO · Parallel

1 Introduction

Metaheuristic algorithms can obtain (near)-optimal results if it is not possible to get the exact solutions of large intractable optimization problems [1, 2]. Most of these algorithms are inspired by nature, and they are known to be the state-of-the-art optimization tools to deal with NP-Hard problems. Since metaheuristic algorithms are an important area of interest, numerous studies have been reported and new notable studies emerge recently [3].

✉ Tansel Dokeroglu
tansel.dokeroglu@tedu.edu.tr

[1] Computer Engineering Department, TED University, Ankara, Turkey

[2] Department of Computer Science, Aalto University School of Science, Espoo, Finland

The background of individual metaheuristic algorithms is full of many significant success stories. They are believed to be the state-of-the-art algorithms for many optimization problems [4]. However, when they reach their computational limits, researchers aim to combine different metaheuristics and develop hybrid metaheuristic algorithms to solve harder optimization problems. According to the no free lunch (NFL) theorem [5], hybrid algorithms are able to solve a wider scope of NP-Hard problems in a better way. The main motivation for combining different metaheuristics is to complement the missing aspects of individual metaheuristics. However, choosing the correct metaheuristics to work in harmony and improve the optimization performance is not a straightforward task.

In this study, we introduce new hybrid artificial bee colony (ABC) algorithms combined with teaching learning-based optimization (TLBO) metaheuristic and investigate their performances for real-parameter optimization functions. Our main purpose is to deal with the issues of premature convergence, stagnation prevention, diverse search areas, well-balancing the exploration and exploitation phases and parameter tuning while providing the robustness effectively. These challenges are the most important research areas of the metaheuristic optimization methods. In this sense, parallel hybrid algorithms can provide good alternatives for the solution to these issues. We develop new island parallel versions of our hybrid algorithms and show that we can obtain better results than that of their sequential versions. Particularly, we propose three new ABC–TLBO hybrid algorithms, island parallel versions of ABC–TLBO algorithms and the island parallel version of the classical ABC algorithm.

Artificial bee colony (ABC) is a nature-inspired swarm metaheuristic introduced by Karaboga [6, 7]. The ABC mimics the foraging behavior of honey bees, and employed, onlooker and scout bees are simulated in the ABC metaheuristic [8]. Numerous optimization problems have been solved optimally by using the ABC algorithm [9]. Teaching learning-based optimization (TLBO) is a population-based metaheuristic that is proposed by Rao et al. [10]. There are learners and a trainer in the optimization environment. The TLBO promises to have only the algorithm-specific parameters, rapid convergence and easy implementation. Depending on the iterations, TLBO trains the individuals in the environment by a teacher or interaction of learners. The classical ABC algorithm uses employed and onlooker bees during the optimization process. In our proposed hybrid algorithms, TLBO improves the processes of the bees by using its training techniques. The details of the proposed algorithms are presented in Sect. 3.

In this study, we enhance the performance of the classical ABC metaheuristic by using the training techniques of TLBO. Island parallel hybrid versions of the proposed algorithms and the classical ABC algorithm are developed with MPI libraries. Robustness is provided by generating diverse initial search spaces for candidate solutions in a parallel computation environment. A pseudorandom number generator is proposed as a seeding mechanism of each processor that optimizes the selected set of benchmark functions.

In Sect. 2, related studies on metaheuristic algorithms that solve the multi-dimensional numerical function optimization are summarized. Section 3 introduces the proposed sequential and parallel hybrid ABC–TLBO algorithms. Section 4 presents

the evaluation of our experimental results. Concluding remarks and future work are provided in the last section.

2 Related work

In this part of the paper, we review studies related to ABC, TLBO, hybrid algorithms and parallel meta-heuristic algorithms.

Two comprehensive surveys on ABC optimization techniques and its applications are presented by Karaboga et al. [8, 11]. Among these techniques, Karaboda and Basturk propose an ABC algorithm for optimizing multivariable functions [12, 13] and compare its performance with that of two algorithms, namely genetic algorithm (GA) and particle swarm optimization (PSO). The results show that the proposed ABC algorithm outperforms the other two methods. Later, Karaboga and Basturk design a basic variant of the ABC algorithm for multi-dimensional numerical problems and this has been shown to achieve competitive results on high-dimensional optimization problems with differential evolution (DE), PSO and evolutionary algorithm (EA) [14]. In another study, Akay and Karaboga modify the classical ABC algorithm to improve convergence rate and examine three control parameters, the modification rate, the scaling factor and the *limit*, for real-parameter optimization [15]. Kıran and Fındık [16] claim that the classical ABC has the slow convergence drawback. Instead of updating more design parameters, the authors exploit directional information added to ABC algorithms. Performance evaluation is conducted on the numerical benchmark functions to compare with the classical ABC. Gao and Liu [17] observe that ABC is poor at exploitation, even though it is good at exploration. Inspired by the DE method, the authors develop a new process based on a bee that explores for the best of the previous solutions. Introducing a new selection probability mechanism and a search technique, the performance of the new algorithm outperforms the other state-of-the-art algorithms. For further improvement in the exploitation ability, Gao et al. propose a hybrid approach combining DE and gbest-guided ABC (GABC) techniques. The algorithm speeds up the convergence of ABC by utilizing more prior information from the previous searches [18]. The improvement in global convergence is provided by a chaotic opposition-based population initialization method for the generation of an initial population.

As an another variant of the ABC techniques, Du et al. [19] introduce search equations for improving the convergence speed and prematurity of ABC algorithm. It has been shown that the novel global search strategy and elite-guided ABC metaheuristic improve feature selection and data clustering results. Gómez and Rodríguez [20] propose multi-objective ABC for the optimization of resources in parallel systems. They compare the algorithm with deterministic and nondominated sorting genetic algorithm II (NSGA-II) resource selection algorithms and verify that better results are obtained in response time and power consumption.

Lim and Isa propose teaching and peer-learning PSO (TPLPSO) algorithm which is a novel variant of PSO improved with TLBO. In the teaching phase of the framework, the particle updates its velocity according to its historical best and the global best values to improve its fitness. If a particle fails to improve, it enters into the

peer-learning phase, where a guidance particle is selected. The search performance is evaluated on 20 benchmark functions and on a real-world problem. The TPLPSO shows comparable performance with state-of-the-art PSO algorithms and seven metaheuristic search techniques [21]. Zou et al. introduce a variant of TLBO, where each learner is trained using the group mean in the teaching phase, but not using the class mean. In the learning phase of the model, each learner applies random learning or the quantum-behaved learning strategy in the group. Experimental evaluation is performed on selected numerical benchmark functions, and each function is evaluated in 10, 30 and 50 dimensions. The results show that the proposed algorithm with a dynamic group strategy is an effective method for global optimization problems [22]. Dokeroglu [23] proposes a set of hybrid TLBO algorithms to solve the quadratic assignment problem. TLBO runs well in coordination with robust tabu Search engine while solving this NP-Hard problem.

Hybrid metaheuristic algorithms show significant improvements over classical versions of metaheuristic algorithms. It is perceived from recent studies that more efficient behavior and greater flexibility can be provided by hybrid metaheuristic algorithms [4]. The main goal of our hybrid algorithms is to make use of the unlike strategies of metaheuristics and benefit from synergy. This hybridization can provide good features to combine different metaheuristics and solve challenging problems instead of proposing new metaheuristics. In this sense, our algorithm contributes and introduces the first and only parallel ABC–TLBO algorithms in the literature.

3 Proposed robust hybrid ABC–TLBO algorithms

In this section, we present our proposed hybrid ABC–TLBO algorithms. First, a brief introduction about the ABC and TLBO metaheuristics is given. Then, the sequential and parallel versions of the proposed hybrid ABC–TLBO algorithms are presented, respectively.

3.1 Artificial bee colony metaheuristic

The ABC algorithm is inspired by the collective behavior of honey bee colonies and has been introduced for solving numerical optimization problems by Karaboga in 2005. It is one of the most cited new generation metaheuristics in the literature [6–8]. Many problems are solved and optimized by using the population-based ABC algorithm. The population of the ABC consists of bees that explore/exploit for the best/optimal food resources (solutions) of given problems [24]. A solution indicates a food resource, and the nectar amount of each resource represents the quality of each solution. There are three types of bees in the hive, namely employed, onlooker and scout bees. Employed bees look for a food source, come back to hive and share their information by dancing. When an employed bee finishes the collection of the nectar, it turns into a scout bee and looks for other food resources. Onlooker bees watch how the employed bees dance and choose food sources accordingly. Scout bees explore for food sources. Initial population is generated randomly in the first

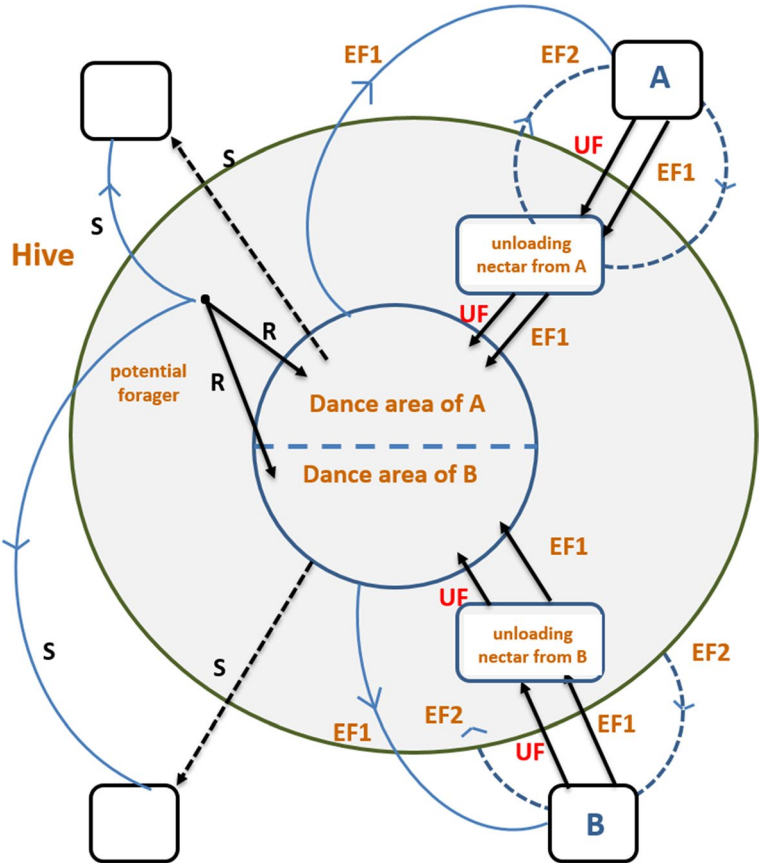

Fig. 1 The classical behavior of honeybees looking for nectar

phase of the ABC algorithm. Figure 1 presents the basic behavior of artificial bees. The optimization process is repeated by using a set of bees. A forager bee starts as an unemployed bee having no information about the food sources around the hive. An ordinary bee can be a scout bee (S) and explore the solution space, or it can watch the dance of other bees and search for new food sources, (R). The bee gathers the food, comes back to the hive and drops off the nectar. The bee can become a recruit nest mates (EF1), an uncommitted follower (UF) or go searching the food without recruiting after bees (EF2) (see Fig. 1 for the behavior of bees of ABC metaheuristic).

The pseudocode of the ABC algorithm is given in Algorithm 1. The execution is repeated as many as the number of iterations. This is the termination condition of the algorithm. At the first phase of each iteration, scout bees are sent to search the problem space randomly and they come back and share their information with other bees. Onlooker bees choose the best candidates for the exploitation phase of the algorithm. Employed bees go to the selected (previously explored solution) locations, and the

exploitation phase of the algorithm starts depending on the parameter settings of the ABC metaheuristic. After the termination of the iterations, the best result is reported as the solution.

Algorithm 1: The pseudocode of the ABC metaheuristic [6].

```
int i=0;
while (i++ < #max_iterations) do
    Scout bees search for food();
    Scout bees return to the hive and dance();
    Onlooker bees evaluate the food sources();
    Check previously visited food resources();
    Decide the best food resources();
    Employed bees travel to the food sources();
    Return to hive();
    Collect the solution in the hive();
```

3.2 Teaching learning-based optimization metaheuristic

TLBO is a population-based metaheuristic algorithm proposed by Rao et al. [10]. The population consists of a group of learners (sample solutions) and a teacher/ trainer in a classroom (population). The first phase of TLBO is the teacher phase, and the second phase is the learner phase. TLBO algorithm is a stochastic swarm intelligence algorithm. TLBO has an iterative evolution process that is similar to standard evolutionary algorithms. The lack of algorithm-specific parameters, rapid convergence and easy implementation of TLBO have attracted the attention of many researchers. This new method is applied to engineering design optimization problems easily [25, 26]. Zou et al. [27] provide a comprehensive survey of prominent TLBO variants and its recent applications and theoretical analysis. Detailed information about TLBO can be found in this survey.

The first phase of TLBO is the teacher phase, and the second phase is the learner phase. The population consists of a group of students/ learners (sample solutions) and a teacher/trainer in a classroom (population). Learners in the classroom can obtain knowledge through interaction with a teacher or their classmates. The best learner is employed as a teacher who is the most knowledgeable person in the population. Then, the teacher spreads information to learners. Through this simple training process, the teacher improves the quality of the learners and also the success of the class. When the improvement does not get better, a better quality teacher is assigned and a new training process can be initialized.

In the teacher phase, let M_i is the mean, T_i be the teacher at iteration i and T_i moves M_i toward its own level and the new mean becomes T_i designated as M_{new}. The new solution is modified with respect to the difference between the current and the new mean given by:

$$\text{Difference\_Mean}_i := r_i(M_{\text{new}} - T_F M_i) \tag{1}$$

where r_i is a random number in the range of [0, 1] and T_F is a *teaching factor* that decides how the mean value will be updated according to a teacher. The value of T_F is decided randomly in a heuristic step, and it can be either one or two with equal probability using $T_F = \text{round}[1 + \text{rand}(0, 1)\{2 - 1\}]$.

The current solution is changed according to this difference by:

$$X_{\text{new},i} := X_{\text{old},i} + \text{Difference\_Mean}_i \tag{2}$$

In the learner phase, students improve their knowledge by the input from the teacher and the interactions of classmates. A student/learner can randomly interact with other learners in the classroom and learn new things if the other classmate has a better knowledge level. A student is randomly selected from the classroom, and this individual trains other randomly selected classmates. If the new individual is better than the former one, they are replaced. The update step of two randomly selected learners, $X_i \neq X_j$, is given by:

$$\text{if}(X_i < X_j) \quad \text{then} \quad X_{\text{new},i} := X_{\text{old},i} + r_i(X_i - X_j) \tag{3}$$

$$\text{if}(X_i > X_j) \quad \text{if} \quad X_{\text{new},i} := X_{\text{old},i} + r_i(X_j - X_i) \tag{4}$$

3.3 Proposed hybrid ABC–TLBO algorithms

The main framework of the proposed algorithms is the same with that of ABC metaheuristic obtained from the Web site of ABC (Karaboga).[1] Two main phases of the ABC algorithm (onlooker bee search and employed bee search) are enhanced by the teacher and learner heuristics of the TLBO algorithm. We develop three different versions of ABC–TLBO according to the heuristics of the TLBO. In version 1, ABC–TLBO-1, the best bee is selected as a teacher and it trains the bees in the hive during the employed bees search phase and onlooker bees search phase. In version 2, ABC–TLBO-2, the bees interact with each other rather than the best bee in the employed bees search phase, whereas the best bee trains the others during the onlooker bees search phase. In version 3, ABC–TLBO-3, the best bee trains in the employed bees search phase, whereas the other bees interact with each other during the onlooker bees search phase.

Table 1 gives the configuration of the proposed hybrid ABC–TLBO algorithms. Figure 2 presents the flowchart of the proposed algorithm ABC–TLBO. During the onlooker bee and employed bee search stages of the ABC algorithm, the techniques of TLBO are configured as reported in Table 1.

[1] https://abc.erciyes.edu.tr/.

Table 1 The configuration of the proposed hybrid ABC–TLBO algorithms

Algorithm	Employed bee process	Onlooker bee process
ABC–TLBO-1	Teacher	Teacher
ABC–TLBO-2	Learner	Teacher
ABC–TLBO-3	Teacher	Learner

The best bee is selected as the teacher or the bees interact with each other during the search process of employed and onlooker bees

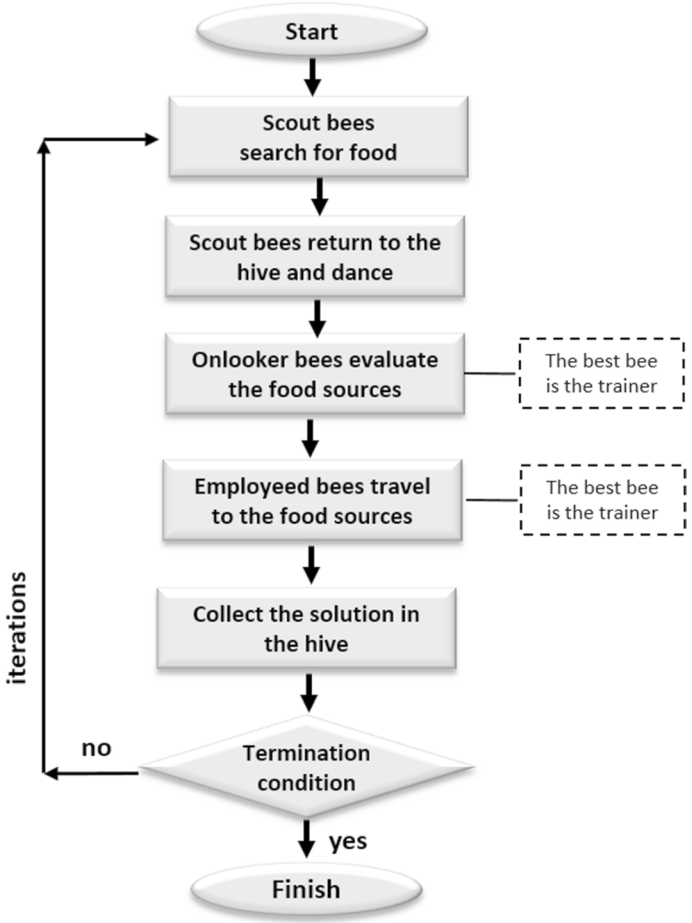

Fig. 2 The flowchart of the proposed ABC–TLBO algorithm

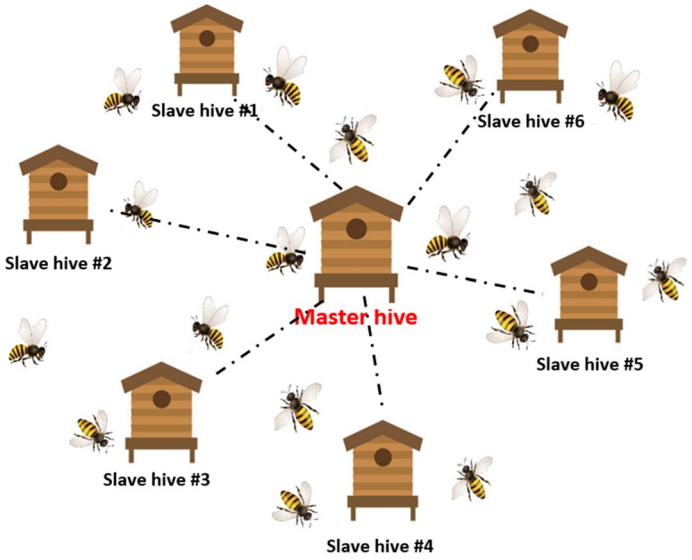

Fig. 3 The communication structure of hives between master and slaves

3.4 Proposed island parallel versions of ABC–TLBO algorithms

According to the evaluations of Manfrin et al. [28] about parallel ant colony optimization algorithms on traveling salesman problem, the performances of metaheuristic algorithms can be improved significantly with parallel implementations. In particular, the island parallel versions of the metaheuristics are reported to be more efficient than the other parallel algorithms in terms of the computation time deviation from the optimal solutions and low cost of communication. Therefore, in this part of our study, we propose island parallel versions of the ABC–TLBO algorithms (see above) for the solution of the problem. Figure 3 shows the communication structure of many hives in a parallel computation environment. The slave processors (hives) initialize their populations and individually optimize the solution of the problem without any communication until the end of the iterations. The slaves use a diversified pseudorandom number generator that produces a different starting point at each hive, which is a very efficient way of exploring the global space of the problem. After finding the solution of the local populations, slaves send their results to the master processor. The master hive collects the best solutions (nectars) of the slave hives. Algorithm 2 gives the details of the proposed parallel P-ABC–TLBO algorithm. Figure 3 shows the structure of hives in the environment. See Fig. 4 for the generic flowchart of the proposed parallel P-ABC–TLBO algorithms.

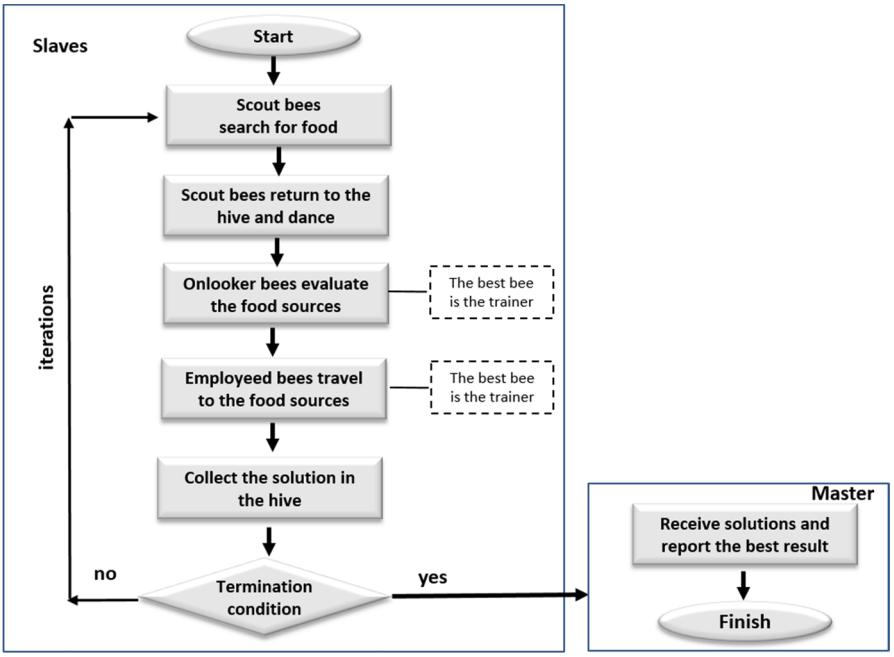

Fig. 4 The flowchart of the proposed parallel P-ABC–TLBO algorithm

Algorithm 2: Generic pseudocode of the P-ABC-TLBO algorithms.

```
1  if (I am a slave hive) then
2      Construct an empty database for food resources();
3      int i=0;
4      while (i++ < #max_iterations) do
5          Scout bees search for food();
6          Scout bees return to the hive and dance();
7          Onlooker bees evaluate the food sources();
8          Check previously visited food resources();
9          Decide the best food resources();
10         Employed bees travel to the food sources();
11         Return to hive();
12         Collect the solution in the hive();
13         Send the best result to the master node();
14 if (I am the master hive) then
15     Receive the solutions from the slaves();
16     Report the best solution();
```

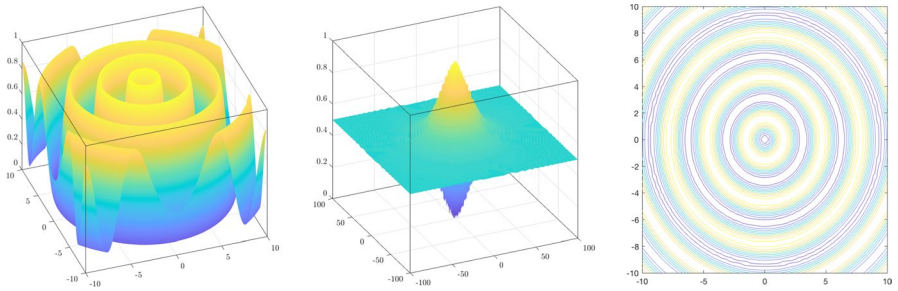

Fig. 5 Schaffer function surface plots (for ranges [− 10, 10] and [− 100, 100]) and contour lines

4 Performance evaluation of the experimental results

We give details of our experimental setup, benchmark problem instances, solution quality of the results, the execution time of our algorithms, comparison with state-of-the-art algorithms, speedup and scalability performance of the proposed parallel P-ABC–TLBO algorithms.

4.1 Experimental setup and problem instances

Our experiments are performed on a high-performance cluster (HPC) computer, HP ProLiant DL585 G7, that has AMD Opteron 6212 CPU running at 2.6 GHz and having 8 cores. CPU has 64-bit computing capacity and AMD SR5690 chipset. The server uses 128 GB PC3-10600 RAM and 1.5 TB hard disk. The software comprises: a Scientific Linux v4.5 64-bit operating system, Open MPI v1.2.4 and C++ programming language.

In order to evaluate the performance of the proposed algorithms, five well-known functions are selected among continuous benchmark functions used for testing the optimization algorithms [29]. These functions are low-dimensional Schaffer and Sphere and high-dimensional Griewank, Rastrigin and Rosenbrock functions [30]. The selected benchmark functions are tested 30 times, and their best/average results and execution times are reported.

The Schaffer function, $f_1(\vec{x})$, is a member of continuous multimodal function family defined in a two-dimensional (2D) space, and it is evaluated on the square $x, y \in [-100, 100]$ (see Fig. 5). The function has a large number of local minima and a global optimum at $x^* = (0, 0)$. The Sphere function, $f_2(\vec{x})$, is convex and unimodal (see Fig. 6). It is defined in a five-dimensional (5D) space and evaluated within the search region defined by [− 100, 100]. The global optimum of the function is at $x^* = (0, \ldots, 0)$. The Griewank function, $f_3(\vec{x})$, is multimodal with many local optima, which are distributed regularly (see Fig. 7). The function is evaluated on a 50-dimensional (50D) search space and on the hypercube $x_i \in [-600, 600]$. Although the general view of the function suggests convex function, zoom in view suggests the existence of numerous local minima. The function has

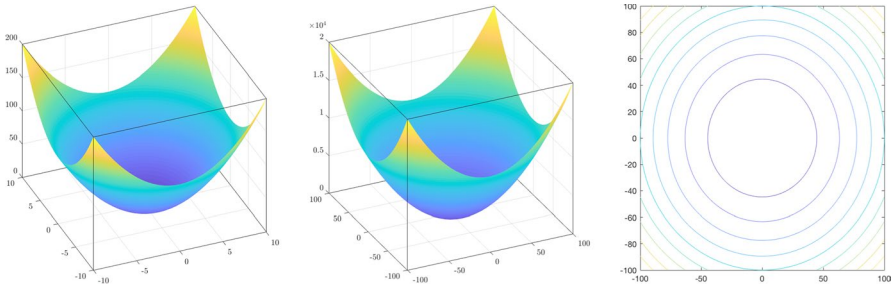

Fig. 6 Sphere function surface plots (for ranges [− 10, 10] and [− 100, 100]) and contour lines, respectively

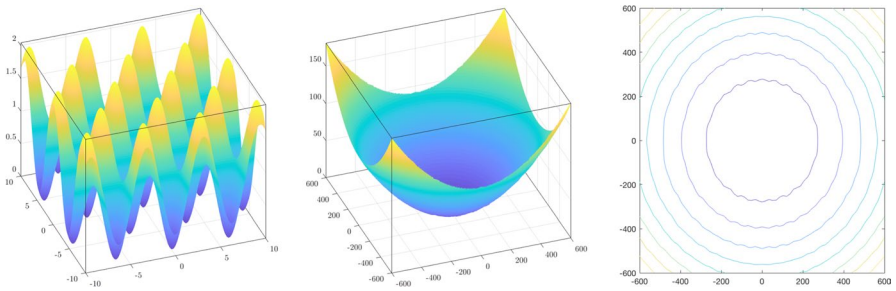

Fig. 7 Griewank function surface plots (for ranges [− 10, 10] and [− 100, 100]) and contour lines

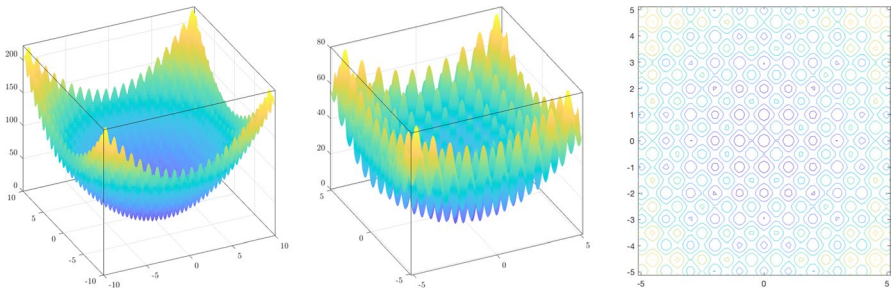

Fig. 8 Rastrigin function surface plots (for ranges [− 10, 10] and [− 5.12, 5.12]) and contour lines

one global optimum at $x^* = (0, \ldots, 0)$. The Rastrigin function, $f_4(\vec{x})$, is multimodal with regularly distributed local optima (see Fig. 8). The function is evaluated on a 50-dimensional (50D) search space and within the search region defined by hypercube $x_i \in [-5.12, 5.12]$. Similarly, the global optimum of the function is located at $x^* = (0, \ldots, 0)$. The Rosenbrock function, $f_5(\vec{x})$, is a multimodal function with a global minimum located at $x^* = (1, \ldots, 1)$ [31] (see Fig. 9 for two-dimensional unimodal version). The function is evaluated on a 50-dimensional

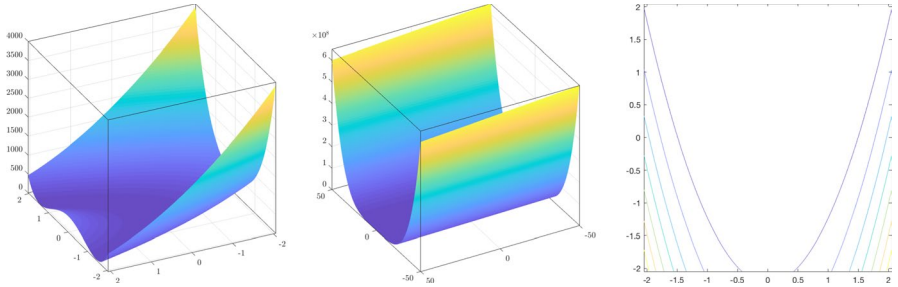

Fig. 9 Rosenbrock function surface plots (for ranges [− 2, 2] and [− 50, 50]) and contour lines

Table 2 Details of the benchmark functions used in the experiments

Name		Formula	Range
Schaffer	$f_1(\vec{x})$	$0.5 + \frac{\sin^2(\sqrt{x^2+y^2})-0.5}{(1+0.001(x^2+y^2))^2}$	$[-100, 100]^D$
Sphere	$f_2(\vec{x})$	$\sum_{i=1}^{n} x_i^2$	$[-100, 100]^D$
Griewank	$f_3(\vec{x})$	$\frac{1}{4000}\left(\sum_{i=1}^{n}(x_i-100)^2\right) - \left(\prod_{i=1}^{n}\cos(\frac{x_i-100}{\sqrt{i}})\right) + 1$	$[-600, 600]^D$
Rastrigin	$f_4(\vec{x})$	$\sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i) + 10)$	$[-5.12, 5.12]^D$
Rosenbrock	$f_5(\vec{x})$	$\sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	$[-50, 50]^D$

Table 3 Parameters used in the algorithms [14, 29]

DE		PSO		EA		ABC	
Pop. size	50	Pop. size	20	Pop. size	100	Colony size	100
CF	0.8	ω	$1.0 \rightarrow 0.7$	p_c	1.0	n_0	50% of the colony
f	0.5	$\varphi_{\min}$	0	p_m	0.3	n_e	50% of the colony
f		$\varphi_{\max}$	2.0	σ_m	0.01	n_s	1
				n	10	Limit	$n_e \times D$

(50D) search space within the search region defined by [− 50, 50]. Table 2 gives detailed information about the functions used in the experiments.

In Table 3, the parameters of the algorithms are presented [14, 29]. CF is the crossover factor of DE, f is the scaling factor, ω is the inertia weight, $\varphi_{\min}$, $\varphi_{\max}$ are the lower and upper bounds of the velocity rule weight, p_c is the crossover rate for EA, p_m is the mutation rate, σ_m is the mutation variance, n is the elite size, n_0 is the number of onlooker bees, n_e is the number of employed bees, n_s is the number of scout bees, D is the number of dimensions in the problem. In our proposed algorithms, the settings provided by Karaboga and Basturk [14] are used (see Table 3).

Table 4 presents the results of the experiments performed with algorithms DE [14], PSO [14], EA [14], ABC [14], directed-ABC [16], DE with gbest-guided

Table 4 The optimization results obtained by DE, PSO, EA, ABC, directed-ABC, DE with gbest-guided ABC (DGABC), (JADE), self-adaptive DE (SaDE) and our ABC–TLBO algorithms

Algorithm	$f_1(\vec{x})$	$f_2(\vec{x})$	$f_3(\vec{x})$	$f_4(\vec{x})$	$f_5(\vec{x})$
DE	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	35.3176 ± 0.27444
PSO	0.0045 ± 0.0009	2.5113E−8 ± 0	1.5490 ± 0.0669	13.1162 ± 1.4481	5142.45 ± 2929.47
EA	**0 ± 0**	**0 ± 0**	0.0062 ± 0.0013	32.667 ± 1.94017	79.818 ± 10.447
ABC	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**	0.069 ± 0.0372
Directed-ABC	**0 ± 0**	**0 ± 0**	2.59E−04	2.42E−01	1.02E+01
DGABC	**0 ± 0**	**0 ± 0**	**0 ± 0**	3.26E−11	**1.59E−03**
JADE	**0 ± 0**	**0 ± 0**	1.57E−08	1.33E−01	1.59E−01
SaDE	**0 ± 0**	**0 ± 0**	2.52E−09	2.43E+00	7.98E−02
ABC–TLBO	**0 ± 0**	**0 ± 0**	1.50E−08	**0 ± 0**	0.345 ± 0.0411

The best results of the experiments are reported in bold face

The results of the ABC–TLBO algorithms are the average of our proposed three algorithms

Table 5 The average optimization results of the algorithms for the functions

Function	ABC	ABC–TLBO-V1	ABC–TLBO-V2	ABC–TLBO-V3
$f_1(\vec{x})$	7.40E−18	0.00E+00	0.00E+00	0.00E+00
$f_2(\vec{x})$	2.11E−17	1.56E−17	1.54E−17	1.70E−17
$f_3(\vec{x})$	8.51E−17	4.49E−08	1.48E−16	8.88E−17
$f_4(\vec{x})$	0.00E+00	0.00E+00	0.00E+00	0.00E+00
$f_5(\vec{x})$	6.90E−02	6.69E−01	2.70E−01	9.67E−02

ABC (DGABC) [18], JADE [32], self-adaptive DE (SaDE) [33] and the average of proposed ABC–TLBO. The values give the average optimization results of 30 different test runs. The results less than E-12 are reported as $\mathbf{0 \pm 0}$. All the algorithms (except PSO) find the optimal solutions for the function $f_1(\vec{x})$ and $f_2(\vec{x})$. For the function $f_3(\vec{x})$, DE, ABC and DGABC are the best performing algorithms. For the function $f_4(\vec{x})$, our algorithms are the best performing ones with DE and classical ABC. For the function $f_5(\vec{x})$, DGABC is the best performing one where ABC–TLBO is the fourth best performing algorithm among the eight state-of-the-art algorithms. When the overall performance of the algorithms is analyzed, DGABC, SaDE and ABC–TLBO are the best performing three algorithms.

When the proposed ABC and hybrid ABC–TLBO versions of the algorithms are analyzed, for the functions Schaffer and Sphere, the ABC is observed to be the worst performing one. For functions Griewank and Rosenbrock, the ABC is the best performing one. Figures 10, 11, 12 and 13 present the solution quality of algorithms ABC, ABC–TLBO-V1, ABC–TLBO-V2 and ABC–TLBO-V3 for the functions, Schaffer, Sphere, Griewank and Rosenbrock, respectively. The average solutions of the algorithms (with 30 runs) are reported in Table 5. For the function Rastrigin, all the algorithms report the optimal solutions.

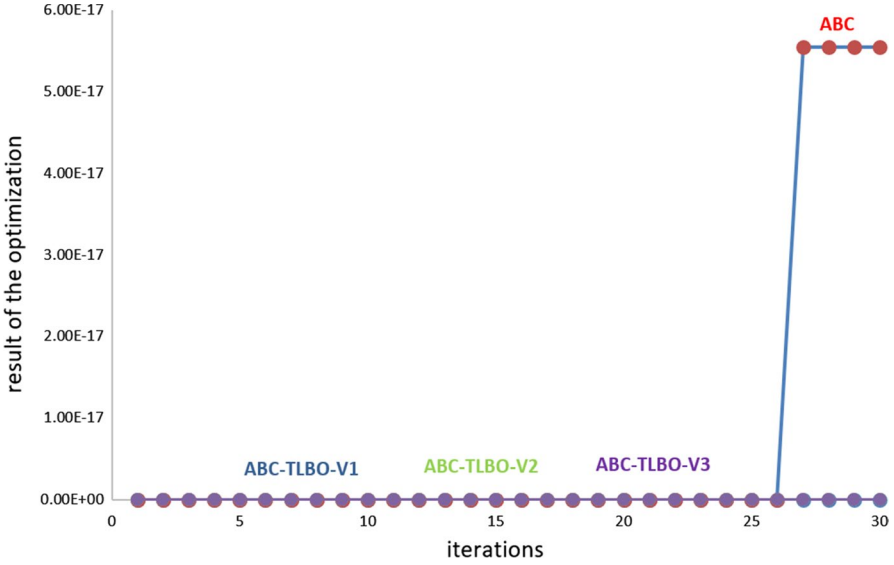

Fig. 10 The comparison of optimization results for Schaffer function $f_1(\vec{x})$

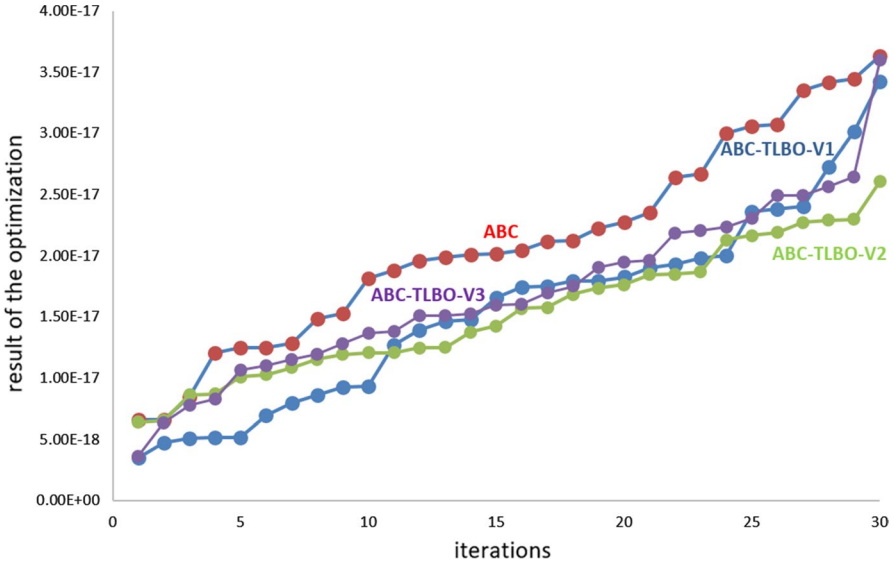

Fig. 11 The comparison of optimization results for Sphere function $f_2(\vec{x})$

In order to get better results by our proposed hybrid ABC–TLBO algorithms, we implement their island parallel versions. We observe much better results than those of sequential ABC and ABC–TLBO algorithms.

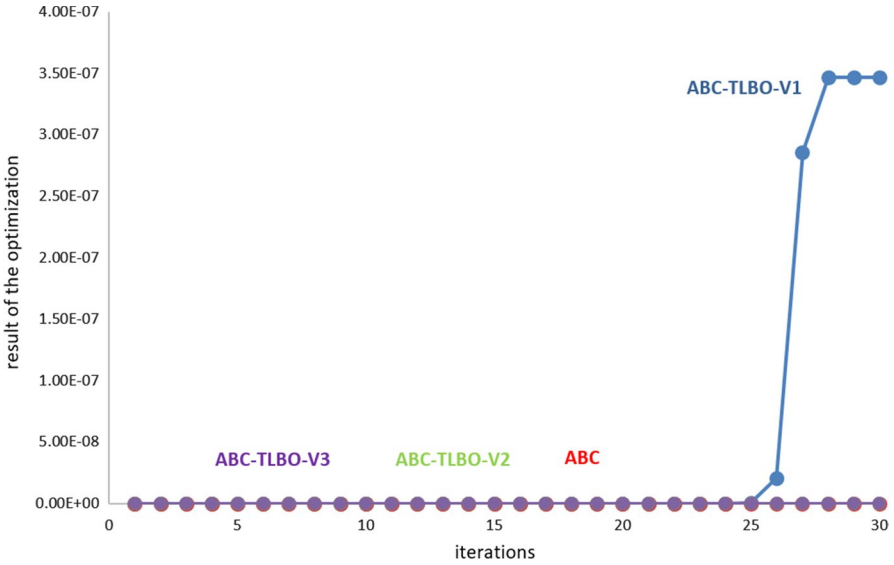

Fig. 12 The comparison of optimization results for Griewank function $f_3(\vec{x})$

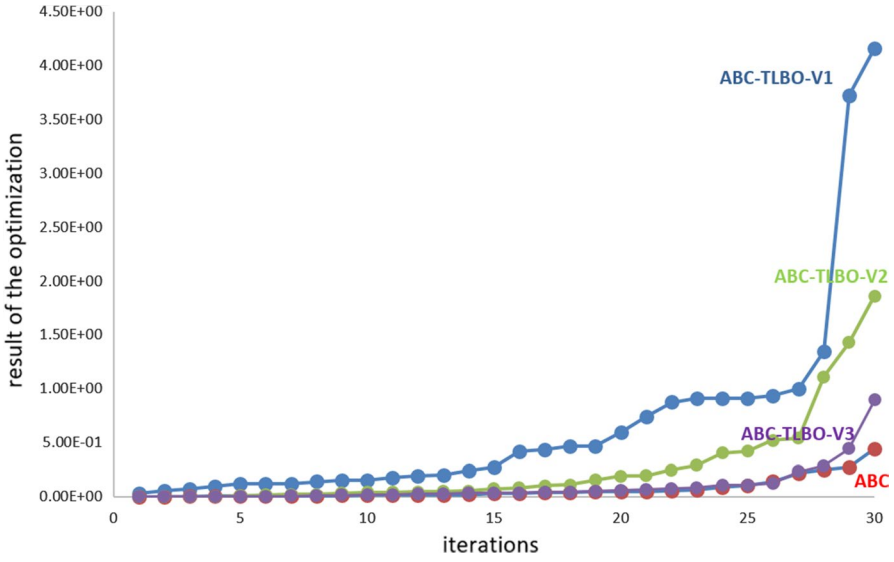

Fig. 13 The comparison of optimization results for Rosenbrock function $f_5(\vec{x})$

4.2 The effect of increasing the number of processors

In this part of our experiments, we observe the effect of increasing the number of processors during the optimization. Figure 14 presents the performance of the

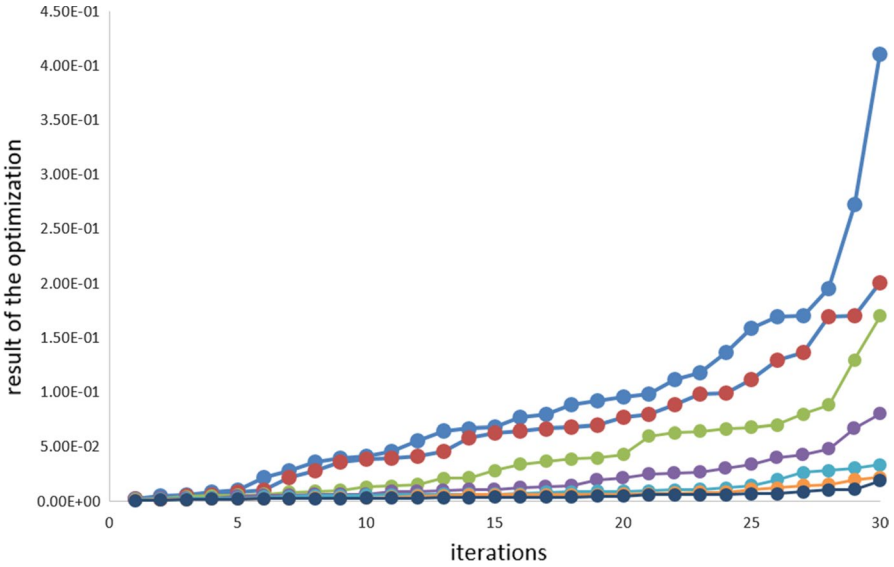

Fig. 14 The effect of increasing the number of processors on the Rosenbrock function with P-ABC–TLBO-V1 algorithm

P-ABC–TLBO-V1 algorithm on the Rosenbrock function which is the most challenging one among five functions. The number of processors is increased from 4 to 256 by doubling the number of processors gradually. From the top chart curve to the bottom, the figure shows the performance of the algorithm with 4 to 256 processors, respectively. With 4 and 256 processors, P-ABC–TLBO-V1 algorithm reports 9.22E−02 and 4.06E−03 values, respectively. This shows a 95.59% improvement in the optimization results of the algorithm.

The same performance is observed with all the proposed parallel algorithms. As we add more processors, the algorithms gradually increase their performances, which is one of the main contributions of our study. It is still possible to add more processors and obtain better results by the proposed parallel algorithms.

4.3 The performance of the proposed parallel algorithms

The performances of the proposed parallel algorithms P-ABC, P-ABC–TLBO-V1, P-ABC–TLBO-V2 and P-ABC–TLBO-V3 are presented in Tables 6, 7 and 8 for the functions Sphere, Griewank and Rosenbrock, respectively. The results show the average values over 30 runs with the increasing number of processors from 4 to 256 by doubling the number of processors gradually. The best performing algorithm is the P-ABC–TLBO-V2 for the Sphere function. For the Griewank function, P-ABC is the best proposed parallel algorithm, and for the Rosenbrock function, P-ABC–TLBO-V3 outperforms the other algorithms. For the Schaffer function, the

Table 6 The fitness values of the parallel algorithms for the Sphere function

Algorithm	4	8	16	32	64	128	256
P-ABC	9.96E−18	7.48E−18	6.08E−18	4.60E−18	3.37E−18	2.36E−18	1.91E−18
P-ABC–TLBO-V1	8.19E−18	6.48E−18	5.23E−18	4.07E−18	2.54E−18	1.86E−18	1.41E−18
P-ABC–TLBO-V2	**6.34E−20**	**3.30E−20**	**1.09E−20**	**7.21E−21**	**3.96E−21**	**2.06E−21**	**1.25E−21**
P-ABC–TLBO-V3	8.32E−18	6.82E−18	5.24E−18	3.77E−18	2.70E−18	2.16E−18	1.57E−18

The best results of the experiments are reported in bold face

The columns show the number of processors used during the optimization

Table 7 The fitness values of the parallel algorithms for the Griewank function. The columns show the number of processors used during the optimization

Algorithm	4	8	16	32	64	128	256
P-ABC	**1.85E−17**	**3.83E−18**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
P-ABC–TLBO-V1	1.26E−16	6.13E−17	3.45E−17	7.93E−18	0.00E+00	0.00E+00	0.00E+00
P-ABC–TLBO-V2	4.07E−17	1.91E−17	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
P-ABC–TLBO-V3	2.96E−17	3.83E−18	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00

The best results of the experiments are reported in bold face

Table 8 The fitness values of the parallel algorithms for the Rosenbrock function. The columns show the number of processors used during the optimization

Algorithm	4	8	16	32	64	128	256
P-ABC	2.59E−02	1.06E−02	4.28E−03	2.80E−03	1.53E−03	9.26E−04	6.11E−04
P-ABC–TLBO-V1	9.22E−02	6.31E−02	3.59E−02	1.80E−02	9.16E−03	6.02E−03	4.06E−03
P-ABC–TLBO-V2	1.25E−02	7.02E−03	2.53E−03	**1.10E−03**	7.84E−04	4.07E−04	2.15E−04
P-ABC–TLBO-V3	**1.14E−02**	**6.25E−03**	**2.51E−03**	1.29E−03	**5.13E−04**	**3.27E−04**	**1.81E−04**

The best results of the experiments are reported in bold face

ABC is the only algorithm that cannot report the optimal results. The P-ABC reports the optimal values by using 4 processors for this function.

The overall results of the island parallel hybrid algorithms are much better than the state-of-the-art algorithms. Of course, the parallel island versions of the other metaheuristic algorithms can be implemented easily and compared with the algorithms proposed in this study. It can be another area of research. Our main goal is to show the efficiency of the parallel island algorithms for these multi-dimensional numerical optimization problems.

4.4 Speedup and scalability analysis of the algorithms

The speedup and scalability are crucial criteria for parallel algorithms. Therefore, we analyze the executions time of the algorithms according to the increasing number

of processors. The execution times of the functions f_1 through f_5 are 0.649, 0.323, 0.504, 0.424 and 39.4 s for 30 runs, respectively. The execution times are almost similar for the sequential versions of ABC–TLBO algorithms. Since the behavior of island parallel algorithms does not need to send many messages, only a 10% execution time overhead is observed when the algorithm is run with 256 processors. Therefore, an almost linear speedup is provided by the proposed algorithms.

5 Conclusions and future work

In this study, we propose robust hybrid parallel ABC–TLBO algorithms for the optimization of the multi-dimensional numeric problems. Metaheuristic algorithms can obtain good results when it is not possible to solve NP-Hard problems exactly. Beyond that, it is applicable to enhance their intelligence by combining them with other metaheuristics and using parallel computation techniques. Robustness can be provided effectively by using these advanced methods. Mainly, our aim in this study is to draw the attention of researchers to parallel hybrid metaheuristic algorithms.

The performances of our proposed algorithms are compared with that of PSO, DE variants, ABC variants and EA for multi-dimensional and multimodal numeric problems. The experimental evaluations verify that the proposed algorithms perform better than the state-of-the-art algorithms. Island parallel versions of the ABC–TLBO algorithms can perform better than their sequential (one-processor) versions. It has been shown that their performances are getting increased with the addition of new processors, while the execution time of the parallel algorithms is observed to be scalable.

As future work, we believe that the studies on hybrid metaheuristic algorithms will continue to exist and hyperheuristic algorithms will be an important research area. With the arrival of new metaheuristics, their hybridization will gain more interest from researchers. Every new metaheuristic can be a good opportunity for novel parallel hybrid metaheuristics. Therefore, we intend to combine the techniques of recent metaheuristics, gray wolf, whale optimization and ABC to provide better parallel hybrid/hyperheuristic algorithms. The other numeric minimization problems (Alpine, Ackley, Penalized 1, Penalized 2, levy, Quartic and Weierstrass) can also be solved by using our proposed hybrid parallel metaheuristic algorithms.

References

1. Neumann F, Witt C (2010) Combinatorial optimization and computational complexity. In: Neumann F, Witt C (eds) Bioinspired computation in combinatorial optimization. Springer, Berlin, pp 9–19
2. Leiserson CE, Rivest RL, Cormen TH, Stein C (2001) Introduction to algorithms, vol 6. MIT Press, Cambridge
3. Yang XS (2010) Nature-inspired metaheuristic algorithms. Luniver Press, Beckington
4. Blum C, Puchinger J, Raidl GR, Roli A (2011) Hybrid metaheuristics in combinatorial optimization: a survey. Appl Soft Comput 11(6):4135–4151

5. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
6. Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Technical report-tr06, Erciyes University, Engineering Faculty, Computer Engineering Department, vol 200, pp 1–10
7. Basturk B (2006) An artificial bee colony (ABC) algorithm for numeric function optimization. In: IEEE Swarm Intelligence Symposium, Indianapolis, IN, USA, p 2006
8. Karaboga D, Gorkemli B, Ozturk C, Karaboga N (2014) A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artifi Intell Rev 42(1):21–57
9. Khader AT, Al-betar MA, Mohammed AA (2013) Artificial bee colony algorithm, its variants and applications: a survey. J Theor Appl Inf Technol 47(2):434–459
10. Rao RV, Savsani VJ, Vakharia DP (2011) Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput Aided Des 43(3):303–315
11. Karaboga D, Akay B (2009) A survey: algorithms simulating bee swarm intelligence. Artif Intell Rev 31(1–4):61–85
12. 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
13. Dokeroglu T, Sevinc E, Kucukyilmaz T, Cosar A (2019) A survey on new generation metaheuristic algorithms. Comput Ind Eng 137:106040
14. Karaboga D, Basturk B (2008) On the performance of artificial bee colony (ABC) algorithm. Appl Soft Comput 8(1):687–697
15. Akay B, Karaboga D (2012) A modified artificial bee colony algorithm for real-parameter optimization. Inf Sci 192:120–142
16. Kıran MS, Fındık O (2015) A directed artificial bee colony algorithm. Appl Soft Comput 26:454–462
17. Gao WF, Liu SY (2012) A modified artificial bee colony algorithm. Comput Oper Res 39(3):687–697
18. Gao WF, Huang LL, Wang J, Liu SY, Qin CD (2016) Enhanced artificial bee colony algorithm through differential evolution. Appl Soft Comput 48:137–150
19. Du Z, Han D, Li KC (2019) Improving the performance of feature selection and data clustering with novel global search and elite-guided artificial bee colony algorithm. J Supercomput 75:1–38
20. Gomez-Martín C, Vega-Rodríguez MA (2018) Optimization of resources in parallel systems using a multiobjective artificial bee colony algorithm. J Supercomput 74(8):4019–4036
21. Lim WH, Isa NAM (2014) Teaching and peer-learning particle swarm optimization. Appl Soft Comput 18:39–58
22. Zou F, Wang L, Hei X, Chen D, Yang D (2014) Teaching–learning-based optimization with dynamic group strategy for global optimization. Inf Sci 273:112–131
23. Dokeroglu T (2015) Hybrid teaching-learning-based optimization algorithms for the Quadratic Assignment Problem. Comput Ind Eng 85:86–101
24. Dokeroglu T, Sevinc E, Cosar A (2019) Artificial bee colony optimization for the quadratic assignment problem. Appl Soft Comput 76:595–606
25. Rao RV, Savsani VJ, Vakharia DP (2012) Teaching-learning-based optimization: an optimization method for continuous non-linear large scale problems. Inf Sci 183(1):1–15
26. Rao R, Patel V (2012) An elitist teaching–learning-based optimization algorithm for solving complex constrained optimization problems. Int J Ind Eng Comput 3(4):535–560
27. Zou F, Chen D, Xu Q (2019) A survey of teaching-learning-based optimization. Neurocomputing 335:366–383
28. Manfrin M, Birattari M, Stutzle T, Dorigo M (2006) Parallel ant colony optimization for the traveling salesman problem. In: Dorigo M, Gambardella LM, Birattari M, Martinoli A, Poli R, Stützle T (eds) International workshop on ant colony optimization and swarm intelligence. Springer, Berlin, pp 224–234
29. Krink T, Filipic B, Fogel GB (2004) Noisy optimization problems-a particular challenge for differential evolution? In: Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753). IEEE, vol 1, pp 332–339
30. Coello CAC, Lamont GB, Van Veldhuizen DA (2007) Evolutionary algorithms for solving multiobjective problems, vol 5. Springer, New York, pp 79–104
31. Shang YW, Qiu YH (2006) A note on the extended Rosenbrock function. Evol Comput 14(1):119–126

32. Zhang J, Sanderson AC (2009) JADE: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 13(5):945–958
33. Qin AK, Huang VL, Suganthan PN (2008) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evol Comput 13(2):398–417