IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 21 March 2023, accepted 27 April 2023, date of publication 5 May 2023, date of current version 17 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3273298

RESEARCH ARTICLE

PCOBL: A Novel Opposition-Based Learning Strategy to Improve Metaheuristics Exploration and Exploitation for Solving Global Optimization Problems

TAPAS SI[1], (Member, IEEE), DEBOLINA BHATTACHARYA[2], SOMEN NAYAK[3], (Member, IEEE), PÉRICLES B. C. MIRANDA[4], UTPAL NANDI[5], SAURAV MALLIK[6], (Member, IEEE), UJJWAL MAULIK[7], (Fellow, IEEE), AND HONG QIN[8]

[1]Department of Computer Science & Engineering, University of Engineering & Management, Jaipur, Rajasthan 303807, India
[2]Department of Computer Science & Engineering, Sarala Birla University, Ranchi, Jharkhand 835103, India
[3]Department of Computer Application, University of Engineering & Management, Jaipur, Rajasthan 303807, India
[4]Departamento de Computação (DC), Universidade Federal Rural de Pernambuco (UFRPE), Recife 52171-900, Brazil
[5]Department of Computer Science, Vidyasagar University, Midnapore, Paschim Medinipur, West Bengal 721102, India
[6]Department of Environmental Health, Harvard T. H. Chan School of Public Health, Boston, MA 02115, USA
[7]Department of Computer Science and Engineering, Jadavpur University, Kolkata 700032, India
[8]Department of Computer Science and Engineering, The University of Tennessee at Chattanooga, Chattanooga, TN 37403, USA

Corresponding authors: Saurav Mallik (sauravmtech2@gmail.com) and Hong Qin (hong-qin@utc.edu)

This work was supported in part by the USA NSF Award under Grant 1761839 and Grant 2200138, and in part by the USA National Academy of Medicine under the Catalyst Award.

ABSTRACT Meta-heuristics are commonly applied to solve various global optimization problems. In order to make the meta-heuristics performing a global search, balancing their exploration and exploration ability is still an open avenue. This manuscript proposes a novel Opposition-based learning scheme, called "PCOBL" (Partial Centroid Opposition-based Learning), inspired by the partial centroid. PCOBL aims to improve meta-heuristics performance through maintaining an effective balance between the exploration and exploitation. PCOBL was incorporated in three different meta-heuristics, and a comparative study was conducted on 28 CEC2013 benchmark problems with 30, 50, and 100 dimensions. In addition, we assessed the PCOBL in the IEEE CEC2011 real-world problems. The empirical results demonstrate that PCOBL balances the exploration and exploitation ability of the meta-heuristics, positively impacting their performance and making them outperform the state-of-the-art algorithms in terms of best-error runs and convergence in most of the optimization problems. Moreover, the computational cost analysis illustrated that the inclusion of PCOBL in the meta-heuristic algorithm has a low impact on its efficiency.

INDEX TERMS Opposition-based learning, optimization, swarm intelligence, meta-heuristic.

I. INTRODUCTION

Global optimization approaches are commonly used to solve optimization problems [1]. In the last decade, a variety of meta-heuristics with different inspirations have been proposed. However, they get trapped in local optima if the operators used for exploration and exploitation do not work efficiently [2]. To avoid such limitations, several works have incorporated Opposition-based Learning (OBL) in the meta-heuristics' procedure to promote exploration and exploitation during the search process [3]. OBL proposes to obtain complementary candidates from a set of solutions. Hence, through the generation of opposite solutions, the intent is to improve the search process' performance concerning coverage of the search space, accuracy, and convergence [4]. Tizhoosh et al. [5] performed the first work that applied OBL to a metaheuristic, proposing the Opposition Genetic Algorithm (OGA). OGA was assessed in benchmark

The associate editor coordinating the review of this manuscript and approving it for publication was Ilaria Boscolo Galazzo.

continuous optimization functions, and the results revealed that the OBL strategy contributed to the metaheuristics' convergence. This work inspired the formulation of new OBL approaches, as well as their applications in different metaheuristics, such as Differential Evolution (DE) [6], [7], [8], [9], Particle Swarm Optimization(PSO) [10], [11], [12], [13], [14], Harmony Search (HS) [15], [16], [17],Simulated Annealing (SA) [18],Salp Swarm algorithms (SSA) [19], [20], [21] and SCA [2], [22].

Aiming to contribute to the area, the scope of this work expands the use of OBL in order to improve the performance of metaheuristics. Herein, we propose a novel OBL scheme, named PCOBL, aiming at a more effective balance between exploration and exploitation during the search process, yet achieving more accurate solutions.

To assess the impact of the PCOBL on the metaheuristics' performance, we incorporated PCOBL in three different approaches: Sine-Cosine Algorithm (SCA), Whale Optimization Algorithm (WOA) [23] and Arithmetic Optimization Algorithm (AOA) [24]. These algorithms set with PCOBL were compared with their original versions, and other two OBL schemes.

All algorithms were evaluated on 28 CEC2013 benchmark problems in terms of best-run errors, convergence, exploration, exploitation and computational time. Besides, we also investigated the proposed scheme in four real-world optimization problems from IEEE CEC 2011 competition [25]. Regarding the best-run errors analysis, the experiment results have illustrated that the PCOBSCA statistically outperforms other algorithms for most of the problems in the CEC2013 benchmark suite. Besides, we performed convergence, exploration, and exploitation analyses. The results showed that the PCOBSCA was able to find promising solutions through an incremental convergence, effectively balancing exploration and exploitation. In terms of exploration and exploitation trade-off, the PCOBSCA reached results statistically superior to OBSCA, and COBSCA, whereas it balances exploration and exploitation similarly to SCA. Similar to PCOBSCA, PCOBWOA statistically outperforms WOA, OBWOA, and COBWOA for most of the problems in CEC2013 suite. PCOAOA also statistically performs better than AOA, OBAOA, and COBAOA for most of the problems in CEC2013 suite. In solving the real-world optimization problems, PCOBSCA, PCOBWOA, and PCOBAOA perform statistically better than their competitive algorithms.

This paper is organized as follows: Section II introduces concepts of OBL, and presents two important OBL schemes for the understanding of the proposal. Section III presents related works. Section IV details the proposal and how to incorporate the PCOBL scheme into a metaheuristic. Section V presents the methodology, and section VI details the achieved results. Finally, Section VII gives the conclusion and presents future works.

II. OPPOSITION-BASED LEARNING

The primary opposition concept has been introduced to the area of Machine Learning in 2005 [5]. Since then, different OBL schemes have been successfully incorporated in metaheuristic algorithms for performance improvement. A detailed review on the use and evolution of OBL approaches can be found in [26].

Definition 1: Let x be a real number having ascertained in the interval $[a, b]$. The opposite number $\check{x}$ is defined as:

$$\check{x} = a + b - x. \tag{1}$$

For instance, if $x = 2$ is a point in the interval $[-5, 5]$, then the opposite of X is -2.

Definition 2: For each point $X(x_1, \ldots.., x_D)$ in the D dimensional space, x_i is defined in the interval $[a_i, b_i]$. There is a unique opposite point $\check{x}_i$ defined as follows:

$$\check{x}_i = a_i + b_i - x_i. \tag{2}$$

For example, if $X = (2, 1)$ is a 2D point in the interval $[-5, 5]^2$, the opposite point of X is $\check{X} = (-2, -1)$.

A. CENTROID OPPOSITION-BASED LEARNING

The COBC is a very popular strategy of OBL scheme, proposed by [27], which achieved remarkable success in DE algorithm in every competition of its kind. When it computes the centroid opposite points in the metaheuristic algorithm, the algorithm considers the entire population. Let $(X_1, \ldots, X_N)$ be N points in a D dimensional search space, which are carrying unit mass in the search space. Then the centroid of the body can be defined as follows:

$$M = \frac{X_1 + X_2 + X_3 + \ldots + X_N}{N} \tag{3}$$

The centroid point at the jth dimension can be calculated as follows:

$$M_j = \frac{1}{N} \sum_{i=1}^{N} x_{i,j} \tag{4}$$

Having defined the centroid of a discrete uniform body as M, the opposite-point $\check{X}_i$ of a point X_i of the body is calculated as follows:

$$\check{X}_i = 2 \times M - X_i \tag{5}$$

The centroid approach can be employed to optimization problems, usually achieving better performance than the min-max method. The estimated boundary, which is based on the generated sample points, is calculated as $[X_{min}, X_{max}]$.

B. PARTIAL OPPOSITION-BASED LEARNING

POBL is a refined type of OBL scheme, commonly found in the literature of Adaptive Differential Evolution (ADE) algorithms [28]. Si and Dutta [29] applied POBL in PSO to train Artificial Neural Network (ANN) for medical data classification. The authors found that POBL improved the

exploration ability of PSO, achieving superior performance than its counterparts. In multi-dimensional search space, a complete opposite point contains opposite values of original values in each dimension in OBL. In a multi-dimensional space, the partial opposite point contains opposite numbers to the original numbers in some dimensions. If $P\check{X}^1$ is the set of partial opposite points of a D dimensional space, then the partial opposite points can be calculated as follows:

$$P\check{X}^1 = \begin{pmatrix} P\check{X}_1^1 \\ P\check{X}_2^1 \\ \cdots \\ P\check{X}_D^1 \end{pmatrix} = \begin{pmatrix} x_1 & \check{x}_2 & \check{x}_3 & . & . & \check{x}_D \\ \check{x}_1 & x_2 & \check{x}_3 & . & . & \check{x}_D \\ . & . & . & . & & \\ . & . & . & . & & \\ . & . & . & & . & \\ \check{x}_1 & \check{x}_2 & \check{x}_3 & . & . & \check{x}_D \end{pmatrix}. \tag{6}$$

The superscript 1 in $P\check{X}^1$ indicates the degree of partial opposition. In each point, as it contains only one original number in one dimension, the illustrated partial opposite points are of order or degree one. So, a point in X in D-dimensional space has D partial opposite points. If (x_1, x_2, x_3) be a 3-dimensional point and $(\check{x_1}, \check{x_2}, \check{x_3})$ be its opposite point the partial opposite points are $P\check{X}_1^1 = (x_1, \check{x_2}, \check{x_3})$, $P\check{X}_2^1 = (\check{x_1}, x_2, \check{x_3})$, $P\check{X}_3^1 = (\check{x_1}, \check{x_2}, x_3)$. As an example, let $X = (-2, 1, 8)$ be a three-dimensional point in the search space range of $[-4, 9]^3$, so the opposite point $\check{X}$ of X is (7,4,-3). Then the set of partial opposite points of X with degree one should be:

$$P\check{X}^1 = \begin{pmatrix} -2 & 4 & -3 \\ 7 & 1 & -3 \\ 7 & 4 & 8 \end{pmatrix} \tag{7}$$

III. RELATED WORKS

Over the years, different approaches have been proposed to improve metaheuristics' performance by balancing exploration and exploitation using OBL. Huang et al. [30] developed an improved SCA applying the chaotic local search mechanism and Lévy flight operator. It showed better performance in solving multimodal problems than its competitive algorithms. Meshkat et al. [31] proposed a weighted update position mechanism to improve SCA's exploration and exploitation abilities. This modified algorithm employed the weighted position update mechanism instead of SCA's original position update rule. The weights are assigned to the search agents based on their fitness values. The modified algorithm performed better than SCA in solving numerical function optimization problems.

Qu et al. [32] used the search operator of neighbors and the greedy Lévy mutation strategy's involvement. First of all, the transmission control parameter and linear decreasing inertia weight were adopted to balance the exploration and exploitation of the search process. Secondly, optimal individuals are replaced by random individuals near the optimal individuals to help the search process in escaping the local optima. Thirdly, it used a greedy Lévy mutation strategy for the best individual to elaborate the local search efficiency of the algorithm. This algorithm performed better than its competitive algorithms in benchmark function optimization.

Gupta et al. [33] developed a modified SCA called MSCA. In this work, a nonlinear time-varying transmission control parameter was incorporated to balance the exploration and exploitation. Secondly, the classical equation for position updating was modified, and a mutation operator was involved in generating a new position to avoid collapse in the local optima. MSCA was applied in solving benchmark optimization problems and in multilayer perceptron training. MSCA performed better than basic SCA and its other competitive algorithms in solving the problems as mentioned earlier.

Gupta et al. [34] developed a memory-guided SCA (MG-SCA) in which each individual saved its personal best-found solution so far in its memory. During the search, guidance is decreased over time. At the beginning of the search process, it explores around the personal best solution, whereas it helps the exploitation later on with less guidance. MG-SCA was tested on IEEE CEC 2014 problems and four engineering optimization problems. MG-SCA performed better than basic SCA and its other competitive algorithms in solving the problems mentioned earlier.

Gupta et al. [35] developed a hybrid SCA called HSCA in which leading guidance and hybridization with simulated quenching algorithm are incorporated. This HSCA algorithm was used to solve CEC2014, CEC2017 optimization problems, four engineering optimization problems, and multilayer perceptron training. HSCA algorithm performed better than basic SCA and its other competitive algorithms in solving the problems as mentioned earlier.

Rehman et al. [36] proposed a Multi Sine-Cosine algorithm in which multiple swarm clusters are utilized for diversification and intensification of the search space to avoid the local optima problem. This modified SCA was tested on benchmark numerical functions, and it performed better than basic SCA.

Different from the previous works, Elaziz et al. [2] and Si et al. [22] were inspired in well succeeding studies that used OBL for metaheuristics [3] to improve SCA's performance. Elaziz et al. [2] proposed an opposition-based SCA (OBSCA) in which opposition-based learning (OBL) was used to make better exploration of the search space to achieve more efficient solutions. In this algorithm, the opposite position, i.e., the solution of each search agent, is computed using Eq. 2 in each iteration after updating the original position. Then best solutions are selected from the updated positions and opposite positions. OBSCA performed better than its competitive algorithms on benchmark function optimization and engineering problems.

Si et al. [22] developed an improved SCA algorithm, named COBSCA, by incorporating an OBL scheme named Centroid Opposition-based Learning (COBL) scheme. In COBSCA, COBL scheme was employed with a generation jumping probability 0.3. If a uniformly distributed random number in $(0, 1)$ is less than generation jumping probability, then opposite solutions are computed using COBL.

Otherwise, the positions of search agents are updated using SCA operators. COBSCA was applied to solve 28 IEEE CEC2013 benchmark problems, and it successfully outperformed SCA for most of the functions.

Long et al. [37] proposed a Refraction learning-based WOA (RLWOA) for high-dimensional optimization problems and parameter estimation of the photovoltaic model. RLWOA adopted a modified conversion parameter update rule to balance exploration and exploitation and Refraction learning strategy to help the population get out from the local optima. RLWOA performed better than standard WOA and WOA variants.

Li et al. [38] proposed an improved WOA known as CQA-WOA in which chaotic maps and quadratic OBL is used to generate initial population with better fitness, and an adaptive variable speed adjustment factor is used to make a good balance between exploration and exploitation. CQAWOA is tested on 20 benchmark numerical functions and three constrained engineering optimization problems, and it showed better performance than standard WOA and other competitive algorithms.

Alamri et al. [39] proposed an opposition-based WOA (OWOA) algorithm in which OBL scheme is employed after the position update of the search agents using the operators of WOA. OWOA tested on ten benchmark function optimization problems, and the results are compared with PSO, and DE.

Chauhan et al. [40] developed a mutation-based arithmetic optimization algorithm (m-AOA) in which a mutation operator provides a balance between exploration and exploitation during the search. m-AOA was tested on 23 benchmark function optimization problems, and it performed better than basic AOA.

An advanced AOA, nAOA, was developed by Agushaka et al. [41] and nAOA used the natural logarithm and exponential operators to generate high-density values to enhance the exploration ability of AOA. nAOA was tested on 30 benchmark functions and three engineering design benchmarks, and it performed better than basic AOA and the other nine competitive algorithms.

Izci et al. [42] proposed an improved AOA by incorporating a modified OBL (mOBL) scheme with a probability. In this mOBL-AOA, the first basic operations of AOA are applied to update the position of search agents, and then mOBL is applied with a probability. mOBL-AOA is tested on benchmark functions of Rosenbrock, Schwefel 2.22, Schwefel, Step, Penalized and Ackley. mOBL-AOA performs better than SCA and original AOA in optimizing the functions as mentioned earlier.

As it can be seen, although OBL has been used successfully in different metaheuristics, the application of OBL schemes has not been extensively studied. According to our knowledge, to date, there are few OBL schemes developed until now, such as classical OBL, and COBL incorporated in SCA, the classical OBL incorporated in WOA, and a modified OBL is incorporated in AOA. The contribution of this paper is two-fold: we proposed a new opposition-based learning scheme by hybridizing COBL and POBL schemes, named PCOBL, aiming to use both schemes' potential. Besides, the PCOBL is incorporated in the three different metaheuristics to enhance their performance. The impact PCOBL has on them is investigated in terms of best-error runs, exploration, exploitation, and computation cost. This novel OBL scheme is detailed next.

IV. PROPOSAL

In this work, we propose a novel OBL scheme, the Partial Centroid OBL (PCOBL), inspired in the COBC and POBL. Herein, section IV-A details how the proposed PCOBL works and is designed to balance exploration and exploitation. Besides, section IV-B presents how metaheuristics can be set with the PCOBL, and the impacted procedures with this combination.

A. PARTIAL CENTROID OPPOSITION-BASED LEARNING

In this work, a new OBL scheme termed Partial Centroid Opposition-based Learning (PCOBL) is proposed. Partial opposite point contains some original numbers in some dimensions and basic opposite numbers in the remaining dimensions. In PCOBL, the scheme is the same as POBL, except it uses opposite numbers generated by the COBL scheme instead of the basic OBL scheme. Like other OBL schemes, PCOBL also enhances the chance of obtaining a better quality of solutions by generating a set of partial centroid opposite solutions. Like the original solution, opposite solutions have the chance to be optimal or near-optimal solutions. Let X be the original solution and $\check{X}_1$, $\check{X}_2, \ldots, \check{X}_l$ be the l number of partial centroid opposite solutions, then the best solution is obtained by using $\min(f(X), f(\check{X}_1), f(\check{X}_2), \ldots, f(\check{X}_l))$ where $f(\cdot)$ is the objective function to be minimized. When PCOBL is incorporated in the *generation jumping phase* of the metaheuristics, better solutions are generated from the existing solutions using PCOBL in one generation. These generated solutions are updated using the update operators of the metaheuristics in the next generations. On the other hand, new solutions are generated using the update operators of the metaheuristics in one generation. From these newly generated solutions, better solutions are obtained by generating opposite solutions using PCOBL in the next generation. PCOBL is designed to have a more effective exploration and exploitation ability in the search space than COBL scheme. With the incorporation of PCOBL in SCA, WOA, and AOA, we intend to produce a more effective balance or *trade-off* between exploration and exploitation and consequently obtain better solutions. Like POBL, many PCOBL points with different degrees can be generated for an original point. The number of partial centroid opposite solutions that can be generated from a given centroid opposite solution is ${}^{D}C_m$ where m is the degree or order of the partial centroid opposite solutions. Therefore, for all degrees $m = 1, 2, \ldots, D$, the total number of partial centroid opposite solutions can be generated from the centroid opposite solutions of an original solution is $|P\check{X}| = \sum_{j=1}^{D-1} C_j = (2^D - 2)$.

Algorithm 1 PCOBL Pseudocode

Input: $X = \{x_1, x_2, \ldots, x_d\}$, its centroid opposite solutions, $\check{X} = \{\check{x}_1^c, \check{x}_2^c, \ldots, \check{x}_d^c\}$, number of partial opposite solutions k
Output: $\{P\check{X}^c\}$

```
dimension=d ← size(X)
for i ← 1 to k do
    P_X̌_i^c = X̌^c
    degree ← ⌈rand(0, 1) × (d − 1)⌉
    for j ← 1 to degree do
        l = ⌈d × rand(0, 1)⌉
        PX̌_il = X_l
return {PX̌^c}
```

Algorithm 2 Population-Based Metaheuristic's Pseudocode Set With PCOBL

```
Initialize the population (X) of size N
Evaluate the fitness
Calculate the opposite solutions X̌^c of X using COBL scheme and their fitness
Select elite N solutions from the set {X, X̌^c}
while (termination criterion) do
    if rand(0, 1) < P_gj then
        Calculate the COBL solutions X̌_i^c of each search agent
        Calculate the fitness of COBL solutions
        Calculate the PCOBL solutions PX̌_i^c of each search agent using Algorithm 1
        Calculate the fitness of PCOBL solutions
        Select the elite N solutions from the set {X, X̌^c, PX̌^c}
    else
        Update the position of the individuals using the operators of metaheuristic
        Evaluate the objective function for each individual
        Update the best position, i.e., best solution (P = X*) found so far
return the best solution found so far.
```

$|P\acute{X}| = {}^D C_1 + {}^D C_2 + \cdots + {}^D C_m + \cdots + {}^D C_{D-1} = 2^D - 2$ as ${}^D C_1 + {}^D C_2 + \cdots + {}^D C_m + \cdots + {}^D C_D = 2^D - 1$. ${}^D C_D = 1$ is the number of D degree partial opposite solution, nothing but the original solution. Therefore, the total number of centroid opposite solution and partial centroid opposite solutions of an original solution X is $(|P\acute{X}| + |\acute{X}|) = (2^D - 2 + 1) = (2^D - 1)$.

Let $X = < x_1, x_2, x_3 >$ be a 3-dimensional point and $\check{X} = < \check{x}_1, \check{x}_2, \check{x}_3 >$ be its centroid opposite point. The partial centroid opposite points of degree 1 are $P\check{X}_1^1 = < x_1, \check{x}_2, \check{x}_3 >$, $P\check{X}_2^1 = < \check{x}_1, x_2, \check{x}_3 >$ and $P\check{X}_3^1 = < \check{x}_1, \check{x}_2, x_3 >$.

TABLE 1. IEEE CEC2013 Benchmarks Problems.

Type	No.	Name	Global Optima ($\mathcal{F}^*$)
Unimodal Functions	1	Sphere Function	−1400
	2	Rotated High Conditioned Elliptic Function	−1300
	3	Rotated Bent Cigar Function	−1200
	4	Rotated Discus Function	−1100
	5	Different Powers Function	−1000
Basic Multimodal Functions	6	Rotated Rosenbrocks Function	−900
	7	Rotated Schaffers F7 Function	−800
	8	Rotated Ackleys Function	−700
	9	Rotated Ackleys Function	−600
	10	Rotated Griewanks Function	−500
	11	Rastrigins Function	−400
	12	Rotated Rastrigins Function	−300
	13	Non-Continuous Rotated Rastrigins Function	−200
	14	Schwefel's Function	−100
	15	Rotated Schwefel's Function	100
	16	Rotated Katsuura Function	200
	17	Lunacek Bi-Rastrigin Function	300
	18	Rotated Lunacek Bi-Rastrigin Function	400
	19	Expanded Griewanks plus Rosenbrocks Function	500
	20	Expanded Scaffers F6 Function	600
Composition Functions	21	Composition Function 1 (Rotated)	700
	22	Composition Function 2 (Unrotated)	800
	23	Composition Function 3 (Rotated)	900
	24	Composition Function 4 (Rotated)	1000
	25	Composition Function 5 (Rotated)	1100
	26	Composition Function 6 (Rotated)	1200
	27	Composition Function 7 (Rotated)	1300
	28	Composition Function 8 (Rotated)	1400

TABLE 2. Parameter Settings of SCA, OBSCA, COBSCA, and PCOBSCA.

Algorithm	Population Size (N)	a	b	P_j
SCA	30	2	-	-
OBSCA	30	2	-	-
COBSCA	30	2	-	0.3
PCOBSCA	30	2	-	0.3

TABLE 3. Parameter Settings of WOA, OBWOA, COBWOA, and PCOBWOA.

Algorithm	Population Size (N)	P_j
WOA	30	-
OBWOA	30	0.3
COBWOA	30	0.3
PCOBWOA	30	0.3

TABLE 4. Parameter Settings of AOA, OBAOA, COBAOA, and PCOBAOA.

Algorithm	Population Size (N)	α	μ	P_j
AOA	30	5	0.5	-
OBAOA	30	5	0.5	0.3
COBAOA	30	5	0.5	0.3
PCOBAOA	30	5	0.5	0.3

The partial opposite points of an original point help in both exploration or global search and exploitation or local search in the search space. Here, an analytical study has been made using Euclidean distance measure to prove how partial

TABLE 5. Mean and standard deviation (in parenthesis) of best-error-runs over 51 independent runs of PCOBSCA, SCA, OBSCA, and COBSCA for 30D problems.

F	PCOBSCA	SCA	h	OBSCA	h	COBSCA	h
1	**32.86** (59.16)	29361.47 (5208.62)	1	262.54 (238.77)	1	262.54 (238.77)	1
2	**16278933.62** (40300431.11)	139843050.7 (8002621.38)	1	820722486.6 (263096692.7)	1	38018327.31 (24304687)	1
3	**565790847.5** (672718852.4)	31232642409 (11807399727)	1	2.15052E+17 (7.33936E+17)	1	6201005714 (4182891294)	1
4	46635.9408 (8690.64)	**34861.02** (5854.33)	-1	71639.39 (8017.430)	1	41948.09 (5846.05)	-1
5	**412.85** (273.97)	2185.38 (653.04)	1	17596.13 (6245.95)	1	1201.35 (504.84)	1
6	**82.28** (23.51)	657.91 (204.46)	1	4158.53 (1029.88)	1	123.96 (39.33)	1
7	**23.94** (9.32)	171.18 (36.58)	1	174797.81 (321234.58)	1	68.62 (17.08)	1
8	20.97 (0.04)	**20.93** (0.05)	-1	21.38 (0.08)	1	20.95 (0.05)	-1
9	**14.24** (2.99)	39.12 (1.40)	1	35.79 (1.66)	1	19.95 (3.13)	1
10	**31.12** (21.48)	1527.29 (307.82)	1	4773.87 (864.97)	1	119.48 (81.16)	1
11	**28.52** (10.90)	356.24 (34.92)	1	704.62 (80.20)	1	94.28 (22.70)	1
12	**107.65** (41.84)	380.82 (33.10)	1	654.75 (80.09)	1	137.13 (29.26)	1
13	**110.59** (42.54)	377.43 (33.05)	1	683.80 (74.29)	1	155.25 (28.92)	1
14	**1176.84** (414.69)	7120.18 (370.20)	1	6917.05 (359.86)	1	4614.72 (1088.55)	1
15	**4382.27** (1955.11)	7356.49 (354.61)	1	6668.10 (442.93)	1	6272.45 (954.32)	1
16	2.59 (0.30)	2.56 (0.27)	0	3.66 (0.70)	1	**2.55** (0.30)	0
17	**135.23** (16.79)	485.59 (37.23)	1	834.88 (68.04)	1	175.49 (18.96)	1
18	**197.66** (12.95)	474.88 (43.87)	1	832.53 (68.44)	1	212.88 (15.59)	1
19	**15.52** (12.82)	2909.15 (2107.83)	1	501959.29 (195714.14)	1	574.16 (1219.69)	1
20	**12.18** (0.41)	13.95 (0.42)	1	15 (0)	1	14.13 (0.33)	1
21	**341.72** (75.68)	1843.23 (179.06)	1	2420.17 (69.68)	1	596.47 (181.62)	1
22	**927.82** (352.15)	7522.51 (483.59)	1	7071.35 (470.44)	1	4630.31 (1668.02)	1
23	**3640.42** (1775.22)	7810.22 (314.97)	1	7571.41 (536.36)	1	7274.92 (907.67)	1
24	**216.85** (5.78)	316.90 (6.49)	1	362.65 (12.67)	1	240.46 (7.54)	1
25	**264.25** (7.62)	328.26 (4.66)	1	390.64 (13.64)	1	285.36 (7.25)	1
26	**200.53** (0.26)	211.88 (5.38)	1	243.50 (15.05)	1	202.18 (3.88)	1
27	**505.20** (82.88)	1359.85 (43.59)	1	1569.41 (143.11)	1	707.26 (111.23)	1
28	**599.28** (126.81)	2591.28 (193.69)	1	6018.54 (605.92)	1	934.26 (209.87)	1

opposite points help in both exploration and exploitation. The Euclidean distance $d(\mathcal{X}, \acute{\mathcal{X}})$ is calculated as follows:

$$d(X, \check{X}) = \sqrt{(x_1 - \check{x}_1)^2 + (x_2 - \check{x}_2)^2 + (x_3 - \check{x}_3)^2} \quad (8)$$

$d(X, \check{X})$ of the centroid opposite point from the original point is maximum. The Euclidean distance of the opposite point $P\check{X}_1^1$ from the original point is as follows:

$$d(X, P\acute{X}_1^1) = \sqrt{(x_1 - \check{x}_1)^2 + (x_2 - \check{x}_2)^2 + (x_3 - \check{x}_3)^2} = \sqrt{(x_2 - \check{x}_2)^2 + (x_3 - \check{x}_3)^2} \quad \left[\because x_1 = \check{x}_1\right] \quad (9)$$

It is obviously that $d(X, \check{X}) > d(\mathcal{X}, \mathcal{P}\acute{\mathcal{X}}_1^1)$. It can be re-written in more generalized form as $d(\mathcal{X}, \acute{\mathcal{X}}) > d(X, P\check{X}^1)$.

TABLE 6. Win, loss, and ties based on Wilcoxon Signed Rank Test statistics for 30D, 50D, and 100D problems in PCOBSCA versus SCA, OBSCA, and COBSCA.

D		SCA	OBSCA	COBSCA
30	Win	25	28	25
	Loss	2	0	2
	Tie	1	0	1
50	Win	24	28	24
	Loss	2	0	2
	Tie	2	0	2
100	Win	24	26	24
	Loss	2	1	1
	Tie	2	1	3

Similarly, it can be said that $d(X, \check{PX}^1) > d(X, \check{PX}^2) > \ldots > d(X, \check{PX}^{(D-1)})$. As the order of the partial centroid opposite points is increased, the distance from the original point is decreased. The partial centroid opposite points with lower-order help in exploration or global search whereas partial centroid opposite points with higher-order help in exploitation or local search in the search space.

In Algorithm 1, an algorithm for the generation of k number of PCOBL solutions with random degrees is given. In this algorithm, inputs are the original solution, its centroid opposite solution, and the number of partial opposite solutions to be generated. In step 3, the first centroid opposite solution $\check{X}^c$ is assigned to ith partial centroid opposite solutions $\{\check{PX}_i^c\}$. Then, the degree of partial centroid opposite solutions is computed in the range $(1, d-1)$ in step 4. In steps 5-7, the components X_l is assigned to the lth position of $\check{PX}_{il}$ where the index l is generated randomly in the range $(1, d)$ in step 6.

B. IMPROVED METAHEURISTIC WITH PCOBL

To incorporate the PCOBL into a metaheuristic, the original metaheuristic have to incorporate two procedures: the opposition-based initialization and generation jumping. Algorithm 2 is a pseudocode that exemplifies how to use PCOBL in a populational-based metaheuristic. In this algorithm, the first step is to initialize the population X. After evaluating the fitness of the search agents in step 2, the centroid opposite solutions $\check{X}^c$ of X are computed using COBL schemes, and fitness values are calculated in step 3. In step 4, the best N solutions are selected from $\{X, \check{X}^c\}$. Steps 6-16 are repeated until the termination criterion is met. Steps 7-11 are comprising of PCOBL execution if the condition in step 6 is true. In step 7, the opposite solutions using COBL are computed, and their fitness values are calculated in step 8. In step 9, the number of opposite solutions $\check{PX}^c$ for each search agent using PCOBL are computed, and their fitness values are calculated in step 10. In step 11, the best N solutions are selected from $\{X, \check{X}^c, \check{PX}^c\}$. If the condition in step 6 is false, steps 13-15, comprised of basic operations of the metaheuristic, are executed.

After the initialization of random positions for search agents in the opposition-based initialization phase, the opposite positions of agents are calculated using COBL scheme, and their fitness values are evaluated. The initial positions and opposite positions are sorted together according to the fitness values in ascending order. After that, the best N agents are selected. In the generation jumping phase, opposition-based computing is performed using a generation jumping probability P_{gj}. In this phase, the opposite positions of agents are generated using PCOBL scheme. Besides, the best N positions are selected from the original, and opposite positions as the same as in the initialization phase.

1) OPPOSITION-BASED INITIALIZATION

In general, metaheuristics use randomly initialized search agents for population initialization. Hence, their performance is not stable. If the initial population is generated close to the global optimum, then there is a better scope to achieve the better convergence speed of the algorithm and its ability to obtain the global optimal solution. As the opposition-based initialization provides a good starting point for the search process, the algorithm's performance improved. In this phase, first, jth element of position X_i of ith solution is generated as follows:

$$x_{ij} = x_j^{\min} + (x_j^{\max} - x_j^{\min}) \times rand(0, 1), \tag{10}$$

where $[x_j^{\min}, x_j^{\max}]$ is the search space range and $rand(0, 1)$ is the uniformly distributed random number in the range $(0, 1)$. After initialization, opposite positions OX of search agents are computed using the COBL scheme. Finally, the best N positions are selected from $\{X, OX\}$ according to the fitness values sorted in ascending order.

2) GENERATION JUMPING

In the generation jumping phase, the opposite solution $\check{X}^c$ of each search agent is computed using the COBL scheme. Next, three PCOBL solutions $(\check{PX}^c)$ with random degree (integer value) in the range $[1, d-1]$ are generated for each agent. Finally, the N elite solutions are selected from the set $\{X, \check{X}^c, \check{PX}^c\}$. The incorporation PCOBL scheme in the metaheuristic accelerates its exploration ability which significantly improves its performance.

3) TIME COMPLEXITY ANALYSIS

In Algorithm 1, step 1 takes $\mathcal{O}(1)$. Steps 3 and 4 take $\mathcal{O}(d)$ and $\mathcal{O}(1)$ respectively. Both steps 6 and 7 take $\mathcal{O}(1)$. Hence, Steps 5-6 take $\mathcal{O}(1) \times degree = \mathcal{O}(degree)$. Therefore, steps 2-7 take $\mathcal{O}(k.d)$ as $d > degree$ i.e., $\mathcal{O}(d) > \mathcal{O}(degree)$. Putting all together, the time complexity of PCOBL is $\mathcal{O}(k.d)$. The time complexity of SCA, WOA, and AOA is $\mathcal{O}(T.N.d)$ where T is the maximum number of iterations and N is the population size. In Algorithm 2, the steps 1-3 take $\mathcal{O}(N.d)$, $\mathcal{O}(N)$, and $\mathcal{O}(N.d)$ respectively. Step 4 takes $\mathcal{O}(N.\log_2 N)$ if the Quick Sort algorithm is used for sorting. Steps 7, 8, and 9 take $\mathcal{O}(N.d)$, $\mathcal{O}(N)$, and $\mathcal{O}(k.d)$ respectively. Step 10 takes $\mathcal{O}(N)$. Step 11 takes $\mathcal{O}(N.\log_2 N)$ if the Quick Sort

TABLE 7. Mean and standard deviation (in parenthesis) of best-error-runs over 51 independent runs of PCOBWOA, WOA, OBWOA, and COBWOA for 30D problems.

F	PCOBOWA	WOA	h	OBWOA	h	COBWOA	h
1	297.96 (517.65)	**0.39** (0.17)	-1	8.57 (7.47)	-1	513.28 (740.05)	0
2	**18925486.99** (7136689.18)	30858312.96 (11347186.69)	1	32250371.72 (14114118.04)	1	32492625.85 (13009249.63)	1
3	**6176773628** (4279800466)	13884706497 (9621502869)	1	11889974292 (6250846488)	1	15144839235 (6947476600)	1
4	**31871.52** (9227.80)	58369.13 (21269.63)	1	50813.96 (10352.22)	1	31724.42 (8693.94)	0
5	106.86 (82.53)	**86.96** (13.96)	0	114.44 (25.70)	1	192.54 (166.53)	1
6	150.45 (63.46)	**107.82** (37.34)	-1	124.47 (27.03)	0	172.84 (47.45)	1
7	**98.33** (23.73)	798.34 (3270.40)	1	168.28 (47.11)	1	149.96 (58.30)	1
8	21.28 (0.08)	**20.94** (0.06)	-1	21.20 (0.10)	-1	21.09 (0.10)	-1
9	**26.77** (3.15)	37.03 (2.69)	1	34.25 (3.15)	1	31.95 (2.99)	1
10	201.19 (190.95)	**44.78** (23.32)	-1	65.59 (28.03)	-1	253.16 (166.78)	0
11	**303.60** (75.10)	499.82 (96.68)	1	435.18 (66.05)	1	418.49 (88.33)	1
12	**313.47** (61.73)	486.47 (106.64)	1	463.80 (94.28)	1	445.52 (87.35)	1
13	**299.10** (57.27)	527.51 (115.67)	1	349.87 (56.94)	1	357.89 (61.57)	1
14	**3311.99** (721.09)	5460.73 (856.49)	1	4495.74 (795.60)	1	3981.70 (755.75)	1
15	**3882.95** (701.54)	5371.77 (792.64)	1	5212.60 (812.09)	1	4408.77 (735.03)	1
16	1.57 (0.97)	1.77 (0.41)	1	2.45 (0.85)	1	**1.40** (0.54)	0
17	**254.50** (62.28)	592.89 (121.96)	1	461.33 (79.59)	1	347.78 (89.28)	1
18	**322.07** (49.70)	601.73 (125.10)	1	411.29 (68.55)	1	376.08 (84.42)	1
19	54.95 (152.93)	58.89 (19.21)	1	**23.40** (4.15)	0	58.67 (199.24)	1
20	13.65 (1.01)	14.70 (0.23)	1	13.72 (0.59)	0	**13.29** (0.88)	-1
21	566.63 (258.18)	**336.59** (82.47)	-1	517.91 (310.85)	0	797.34 (388.56)	1
22	**4089.85** (893.90)	5660.86 (1047.10)	1	5596.53 (953.49)	1	5257.23 (955.88)	1
23	**5351.35** (949.71)	6579.86 (899.86)	1	6037.26 (1064.69)	1	5959.25 (866.12)	1
24	**284.76** (11.77)	312.47 (9.23)	1	299.97 (11.08)	1	302.26 (8.75)	1
25	**304.15** (12.10)	319.95 (10.70)	1	315.42 (10.94)	1	317.10 (12.88)	1
26	**274.52** (85.53)	305.59 (99.27)	1	318.75 (91.41)	1	327.04 (86.22)	1
27	**1069.60** (90.80)	1339.13 (78.20)	1	1245.09 (98.75)	1	1224.51 (94.35)	1
28	**2833.50** (645.72)	3950.36 (1026.11)	1	3761.13 (817.69)	1	3766.74 (471.56)	1

algorithm is used for sorting. The steps 13-15 corresponding to operations of metaheuristics take $\mathcal{O}(N.d)$. Steps 6-15 take $\mathcal{O}(N.d)$. Putting all together, finally, the time complexity of PCOBSCA $\mathcal{O}(T.N.d)$. Similarly, the time complexity of both PCOBWOA and PCOBAOA is $\mathcal{O}(T.N.d)$. After analyzing the time complexity, it has been observed that PCOBL does not affect the running time of SCA, WOA, and AOA.

V. EXPERIMENTAL SETUP

In this section, we present the experimental methodology and setup used to evaluate the proposed scheme. To assess the impact of the PCOBL on the metaheuristics' performance, we incorporated PCOBL in the SCA, WOA and AOA. These algorithms set with PCOBL (named PCOBSCA, PCOBWOA and PCOBAOA) were compared with their original versions (named SCA, WOA and AOA), and other two OBL schemes:

TABLE 8. Win, loss, and ties based on Wilcoxon Signed Rank Test statistics for 30D, 50D, and 100D problems in PCOBWOA versus WOA, OBWOA, and COBWOA.

D		WOA	OBWOA	COBWOA
30	Win	22	21	22
	Loss	5	3	2
	Tie	1	4	4
50	Win	21	20	22
	Loss	6	3	0
	Tie	1	5	6
100	Win	18	19	22
	Loss	7	7	0
	Tie	3	2	6

traditional OBL (named OBSCA, OBWOA and OBAOA) and COBL (named COBSCA, COBWOA and COBAOA). All methods were assessed in terms of best-error-runs $E = |f^*(x) - f(x)|$ where $f^*(x)$ is the global optimum and $f(x)$ is the best objective function values, convergence, and exploration and exploitation over twenty-eight benchmark single-objective optimization problems. We varied the problems' dimensions (30D, 50D, and 100D) to investigate the proposed approaches' performance in medium and high dimensionality scenarios. Besides, we also assessed all methods in the IEEE CEC2011 Real World Optimization Problems.

A. IEEE CEC2013 BENCHMARK FUNCTIONS

All methods were tested on 28 problems in IEEE CEC2013 benchmark suite [43] with 30, 50, and 100 dimensions. The summary of the problems is given in Table 1. Different functions have different levels of complexity. The readers are directed to [43] for further details of these functions. In the benchmark suite, functions 1-5 are unimodal, 6-20 are basic multimodal, and 21-28 are composition functions. The search space range of the problems is $[-100, 100]$.

B. PARAMETER SETTINGS AND PC CONFIGURATION

The parameter settings related to the SCA, WOA and AOA's variants are given in Tables 2- 4, respectively. When the maximum number of function evaluations is reached, the executions of algorithms are terminated. Herein, the maximum number of function evaluations (FEs) = $10,000 \times D$, where D is the dimension of the problems.

To perform the experimentation, we used a personal computer with the following configuration:

1) CPU: Intel(R) Core(TM) i7-9700K @3.60 GHz,
2) RAM: 64 GB,
3) Operating System: Windows 10 Professional 64-bit,
4) Software: MATLAB 2018b.

For all methods, the exact initial population was set in each run to make a fair comparison of all considered methods. The proposed algorithm has run for 30D, 50D, and 100D for 51 independent runs.

VI. RESULTS AND DISCUSSION

All methods were assessed regarding different aspects: best-error runs, convergence, exploration and exploitation, and computational time cost when applied in global optimization problems from the CEC 2013 competition. Besides, all methods had their performances evaluated in real optimization problems from the CEC 2011 competition. The experimental results are detailed in the following sections.

A. BEST-ERROR-RUNS ANALYSIS

Tables 5, 25 & 26 present the mean and standard deviation of the best-error-runs over 51 independent runs of the PCOBSCA, SCA, OBSCA, and COBSCA on 28 benchmark functions for D=30, D=50, and D=100, respectively. The boldfaced results in these tables indicate better. A non-parametric test, the Wilcoxon Signed-Rank Test (WSRT) [44], was applied with the significance level (α) = 0.05 to assess whether the best performing algorithm's results differ from the competitors' final results in a statistically significant way. A summary of the test results is reported in Table 6 for 30D, 50D, and 100D problems, respectively. In Tables 5, 25 & 26, $h = 1$ and $h = -1$ indicate that the null hypothesis is rejected, whereas $h = 0$ indicates the acceptance of the null hypothesis. $h = 1$ indicates PCOBSCA statistically performs better than others in the pair, whereas $h = -1$ indicates that other algorithm in the pair statistically performs better than PCOBSCA.

The best-error-runs for 30D problems are illustrated in Table 5. As it can be observed, the PCOBSCA outperforms the SCA statistically in 25 functions, the OBSCA in all the 28 functions, and the COBSCA in 25 functions, as revealed in Table 6. In general, for 30D problems, the PCOBSCA lost in two functions: one unimodal function (f_4), and one basic multimodal function (f_8). Although the PCOBSCA has lost in one out of five unimodal functions and in one basic multimodal function, its results are competitive compared to the other algorithms.

Another point to be highlighted is the performance of the PCOBSCA in basic multimodal and composition functions. These functions are harder to be optimized due to their multiple local minima. However, the PCOBSCA was able to find suitable solutions. The statistical superiority of the PCOBSCA over the other algorithms (winning 26 out of 28 30D problems, on the average case) shows its capacity to deal with functions with different complexities.

Tables 25 and 26 present the best-error-runs for the 50D and 100D problems, respectively. These tables reveal that the PCOBSCA's performance is not too affected by the increase in the number of dimensions.

In general, for 50D problems, the PCOBSCA lost in three functions: one unimodal function (f_4), and two basic multimodal functions (f_8 and f_{16}). Comparing the PCOBSCA's performance in the 50D with the 30D scenario, few changes happened, and the statistical superiority of the PCOBSCA over the other algorithms remained (winning 25.33 out of 28 50D problems, on the average case) - as revealed in Table 6. Regarding the 100D problems, in general, the PCOBSCA lost in two functions: one unimodal function (f_4), one basic

TABLE 9. Mean and standard deviation (in parenthesis) of best-error-runs over 51 independent runs of PCOBAOA, AOA, OBAOA, and COBAOA for 30D problems.

F	PCOBAOA	AOA	h	OBAOA	h	COBAOA	h
1	**41004.56** (5598.56)	45131.38 (6935.01)	1	44940.72 (6552.86)	1	46532.03 (6182.50)	1
2	**442400400.5** (265522190.4)	492724819.2 (236269086.6)	0	461808301.9 (223648223.1)	0	513816579.7 (315990079.1)	0
3	**1.00741E+15** (2.52984E+15)	2.24901E+17 (8.51385E+17)	1	3.08245E+16 (1.28108E+17)	0	3.72551E+15 (1.5744E+16)	0
4	**54015.53** (5866.10)	61231.27 (4304.11)	1	60297.91 (5570.86)	1	60346.38 (5394.95)	1
5	3111.30 (738.80)	**2859.50** (270.68)	-1	3812.23 (505.96)	1	3487.19 (584.71)	1
6	**6259.43** (2451.72)	7129.01 (2484.31)	1	7583.84 (2725.08)	1	7589.44 (2764.93)	1
7	**103961.44** (182961.92)	416345.55 (587251.33)	1	154261.86 (336043.39)	0	252717.84 (410028.11)	1
8	20.95 (0.05)	**20.94** (0.05)	0	20.966 (0.05)	0	20.96 (0.05)	0
9	**37.60** (2.02)	38.98 (1.63)	1	38.17 (1.67)	0	38.88 (1.34)	1
10	**4350.52** (681.09)	5255.71 (973.95)	1	5272.75 (977.30)	1	4871.11 (966.64)	1
11	**620.19** (83.80)	648.45 (88.46)	0	665.31 (92.58)	1	648.35 (80.91)	0
12	**599.59** (69.27)	663.876 (93.66)	1	647.24 (89.08)	1	623.59 (92.95)	0
13	**611.98** (80.235)	728.79 (77.8982)	1	657.79 (99.2638)	1	644.76 (73.1780)	1
14	**5610.77** (512.85)	6037.89 (487.69)	1	6056.77 (463.38)	1	6208.04 (533.25)	1
15	7058.32 (443.44)	**6796.16** (540.46)	-1	6935.72 (493.66)	0	7031.29 (494.61)	0
16	2.4662 (0.26)	**2.37** (0.31)	0	2.43 (0.39)	0	2.43 (0.30)	0
17	**726.40** (75.71)	847.67 (71.31)	1	876.92 (56.44)	1	831.61 (94.74)	1
18	807.38 (70.98)	822.55 (69.80)	0	827.98 (49.53)	0	**781.43** (63.09)	0
19	63855.56 (24366.7768)	40089.25 (7696.7751)	-1	67331.10 (23806.4062)	0	**60369.03** (19112.0682)	0
20	**14.81** (0.18)	14.87 (0.19)	0	14.87 (0.18)	0	14.88 (0.16)	0
21	**2296.87** (64.95)	2311.42 (53.27)	0	2342.90 (63.20)	1	2357.13 (57.10)	1
22	**6500.48** 9458.50)	6838.79 (568.93)	1	7128.33 (523.58)	1	7238.21 (711.22)	1
23	7666.06 (527.36)	7581.32 (649.07)	0	**7509.24** (722.71)	0	7832.76 (615.74)	0
24	**399.99** (33.64)	422.35 (35.71)	1	407.44 (35.58)	0	425.38 (33.54)	1
25	**392.55** (23.52)	394.80 (27.66)	0	388.09 (18.40)	0	395.25 (23.09)	0
26	**321.67** (68.16)	363.88 (57.31)	1	348.55 (62.51)	1	357.20 (58.90)	1
27	1477.50 (92.11)	1516.58 (130.13)	0	**1456.93** (97.60)	0	1500.52 (82.42)	0
28	**4632.29** (461.46)	5124.23 (477.89)	1	4902.06 (453.28)	1	4690.60 (421.09)	0

multimodal function (f_{16}). In this scenario, the PCOBSCA remains, in general, as the best algorithm, winning in 24.67 out of 28 100D problems, on the average case, as revealed in Table 6.

From the above analysis of the results, it is clear that the proposed PCOBSCA statistically outperforms SCA, OBSCA, and COBSCA for most of the functions (with different dimensions, 30D, 50D, and 100D) in the CEC2013 benchmark suite. Thus, we conclude that the proposed PCOBSCA is efficient and effective in several numerical optimization functions.

Tables 7, 27 & 28 present the mean and standard deviation of the best-error-runs over 51 independent runs of the PCOBWOA, WOA, OBWOA, and COBWOA algorithms

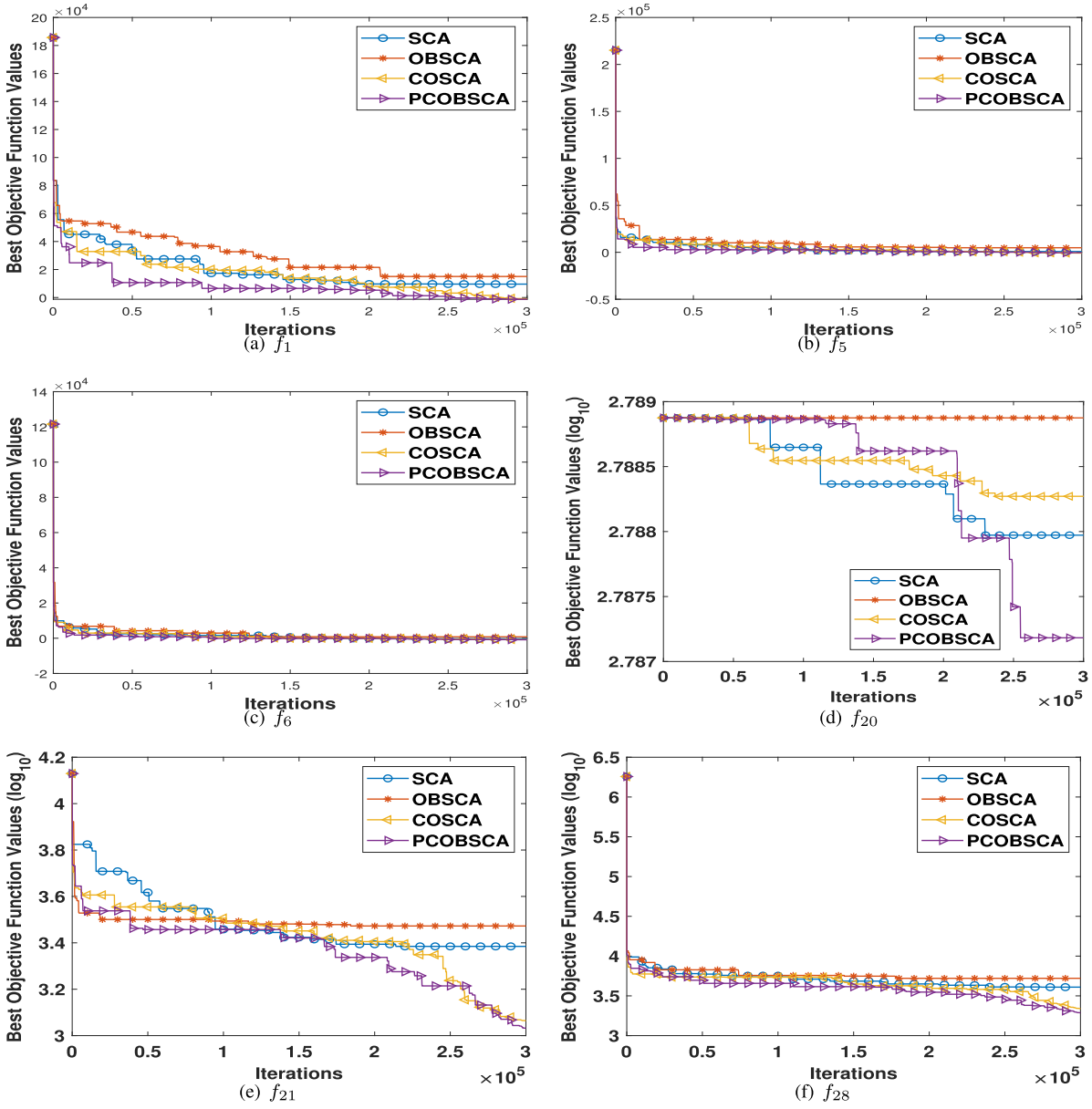

FIGURE 1. Convergence graphs of PCOBSCA for unimodal functions 1, 5, multimodal functions 6, 20, and composite functions 21, 28 with 30 dimensions.

TABLE 10. Win, loss, and ties based on Wilcoxon Signed Rank Test statistics for 30D, 50D, and 100D problems in PCOBAOA versus AOA, OBAOA, and COBAOA.

D		AOA	OBAOA	COBAOA
30	Win	15	14	14
	Loss	3	0	0
	Tie	10	14	14
50	Win	19	17	19
	Loss	3	1	0
	Tie	6	10	9
100	Win	17	14	17
	Loss	3	3	1
	Tie	8	11	10

on 28 benchmark functions for D=30, D=50 and D=100, respectively. The bold-faced results in these tables indicate better. A summary of the WSRT results with a significance level (α)=0.05 is reported in Table 8 for 30D, 50D, and 100D problems whereas function-wise WSRT results (h) are given in Tables 7, 27 & 28. It is observed that PCOBWOA wins over WOA, OBWOA, and COBWOA for 22, 21, and 22 functions, respectively, with 30 dimensions. PCOBWOA loses to WOA, OBWOA, and COBWOA for 5, 3, and 2 functions, respectively, whereas there are ties for 1, 4, and 4 functions with the same dimensions. With 50 dimensions, PCOBWOA wins over WOA, OBWOA, and COBWOA for 21, 20, and 22 functions, respectively. For the same dimensions, PCOBWOA loses to WOA, OBWOA, and COBWOA for 5, 3, and 2 functions, respectively, whereas there are ties for respectively 6, 3, and 0 functions. With 100 dimensions, PCOBWOA wins over WOA, OBWOA, and COBWOA for 18, 19, and 22 functions, respectively. For the same dimensions, PCOBWOA loses to

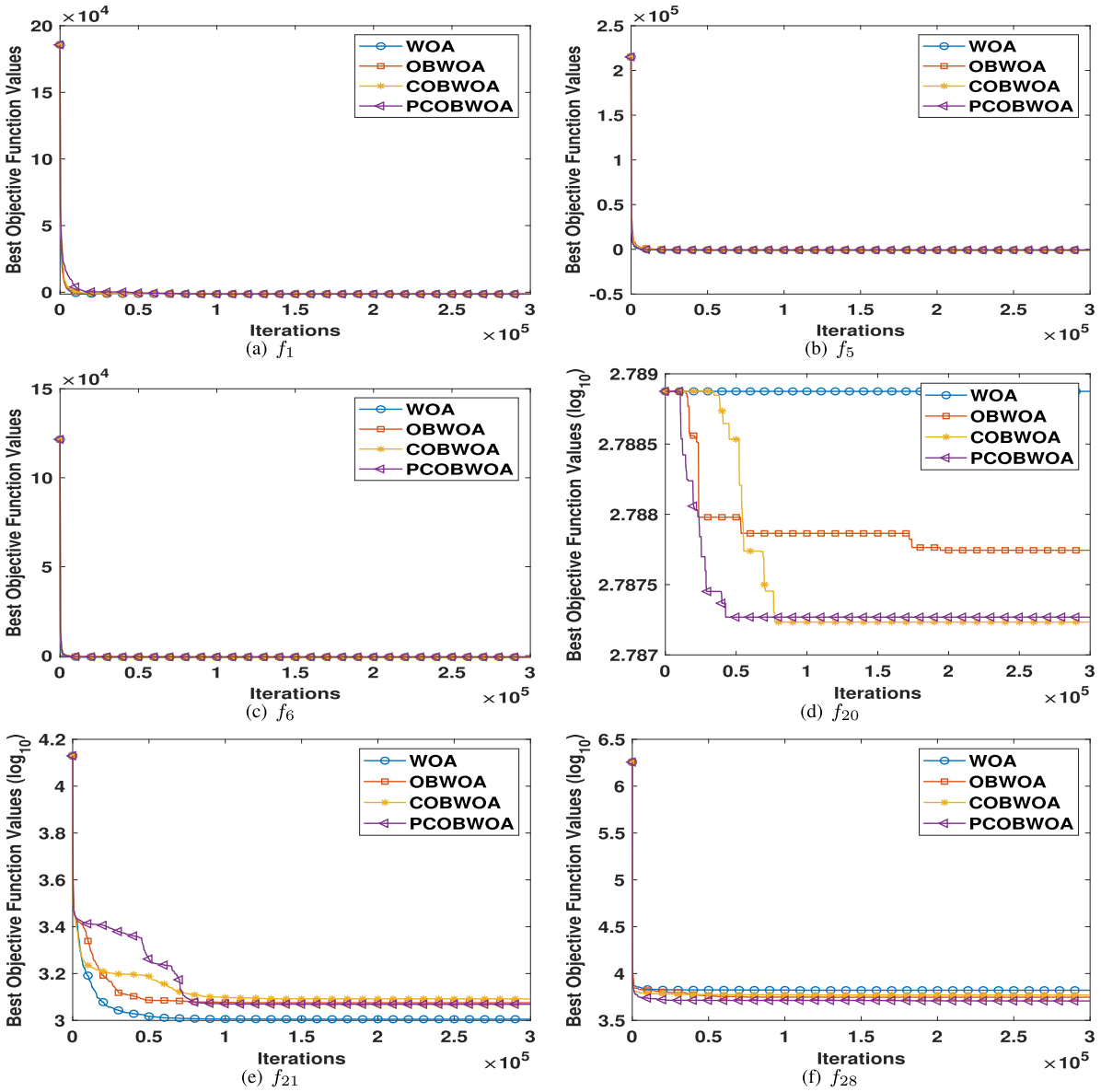

FIGURE 2. Convergence graphs of PCOBWOA for unimodal functions 1, 5, multimodal functions 6, 20, and composite functions 21, 28 with 30 dimensions.

WOA, OBWOA, and COBWOA for 7, 7, and 0 functions, respectively, whereas there are ties for respectively 3, 2, and 6 functions. As it can be seen in this analysis, the PCOBL improved the WOA's performance in most of the problems independently of the number of dimensions.

The mean and standard deviation of the best-error-runs over 51 independent runs of the PCOBAOA, AOA, OBAOA, and COBAOA algorithms on 28 benchmark functions for D=30, D=50, and D=100 are reported in Tables 9, 29 & 30 respectively. The bold-faced results in these tables indicate better. The summary of the WSRT results with a significance level (α)=0.05 is reported in Table 10 for 30D, 50D, and 100D problems whereas function-wise WSRT results (h) are given in Tables 9, 29 & 30. It has been observed that there are 15, 14, and 14 wins of PCOBAOA over AOA, OBAOA, and COBAOA, respectively, for 30D problems. There are three losses of PCOBAOA to AOA and no loss to OBAOA and COBAOA for the same dimensions, whereas there are 10, 14, and 14 ties of PCOBAOA with AOA and OBAOA, and COBAOA, respectively. PCOBAOA wins over AOA, OBAOA, and COBAOA for 19, 17, and 19 functions, respectively, with D=50. PCOBAOA loses to AOA, OBAOA, and COBAOA for 3, 1, and 0 functions, respectively, whereas there are 6, 10, are 9 ties, respectively. For 100D problems, there are 17, 14, and 17 wins of PCOBAOA over AOA, OBAOA, and COBAOA, respectively. For the same

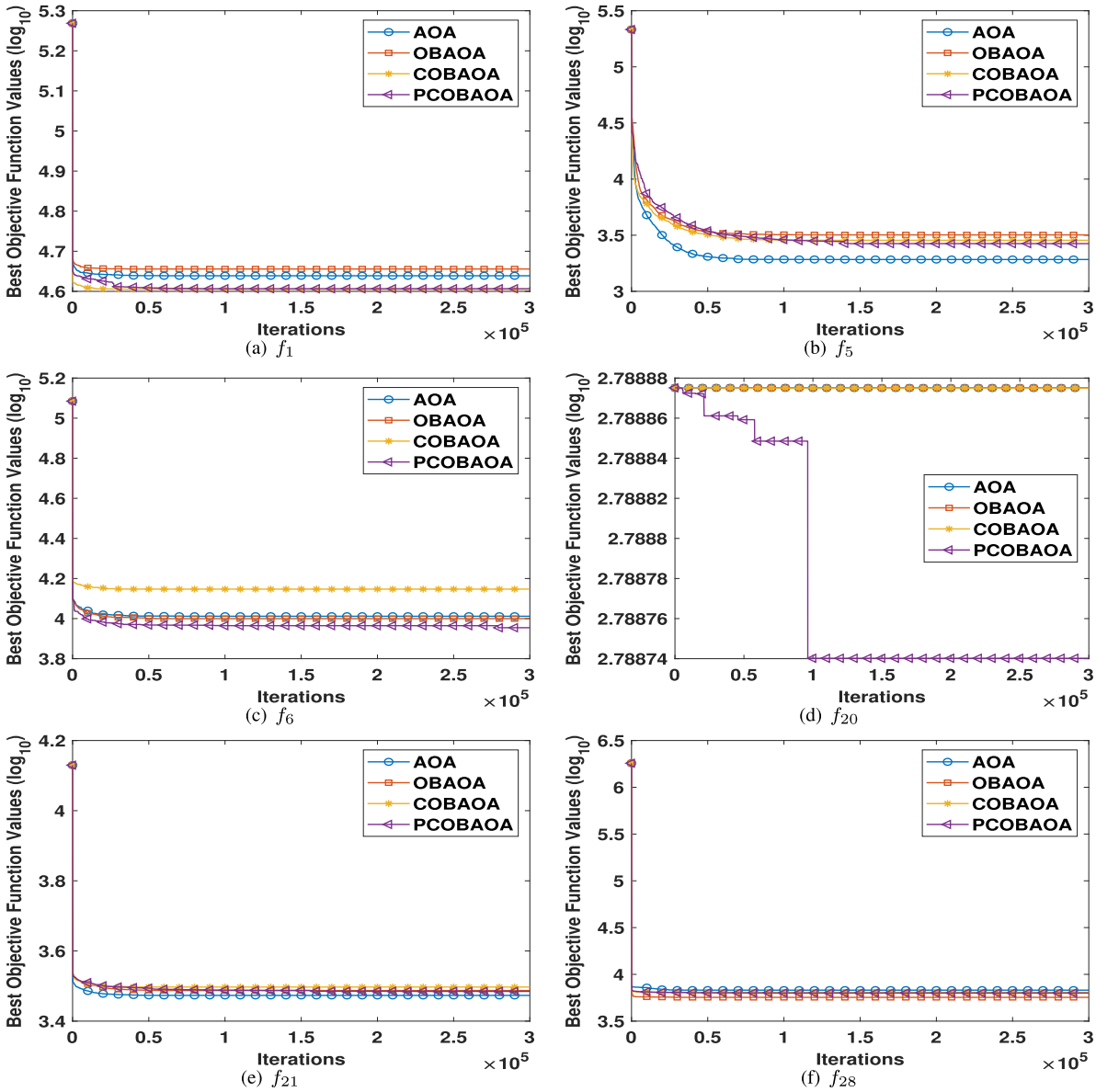

FIGURE 3. Convergence graphs of PCOBAOA for unimodal functions 1, 5, multimodal functions 6, 20, and composite functions 21, 28 with 30 dimensions.

dimensions, there are 3, 3, and 1 losses of PCOBAOA to AOA, OBAOA, and COBAOA, respectively, whereas there are respectively 8, 11, and 10 ties.

By proper tuning of generation jumping probability or by incorporating deterministic/adaptive rules to control it, the performance of PCOBSCA, PCOBWOA, and PCOBAOA can be enhanced. We have noticed the performance improvement in the case of different opposition-based metaheuristic algorithms (e.g., Opposition-based DE, adaptive generalized opposition-based DE) with the strategy above in the study [26].

Robustness (i.e., the ability to produce similar results over multiple runs on a single problem) [45] is another performance measure and it is measured in terms of standard deviation. A lower standard deviation value indicates higher robustness. It is observed that PCOBSCA, PCOBWOA, and PCOBAOA have higher or competitive robustness compared to their counterpart algorithms.

The above analysis shows that PCOBSCA, PCOBWOA, and PCOBAOA statistically outperform their counterparts for most of the functions in the benchmark suite with D=30, 50, and 100. These facts are evidence that establishes that the proposed PCOBL scheme is better than basic OBL and COBL schemes.

B. CONVERGENCE ANALYSIS

To complement the best-error runs investigation, we also performed the convergence analysis. Herein, we investigate

TABLE 11. Mean XPL(%), mean XPT(%) and mean TO(%) of PCOBSCA, SCA, OBSCA, and COBSCA for functions 1-18 with D=30.

F		PCOBSCA	SCA	OBSCA	COBSCA
1	XPL	**51.41**	51.36	48.49	47.38
	XPT	48.59	48.64	51.51	**52.62**
	TO	46.84	46.95	**48.03**	43.61
2	XPL	55.83	54.57	**57.61**	53.46
	XPT	44.17	45.43	42.39	**46.54**
	TO	46.19	**46.57**	44.61	43.02
3	XPL	57.20	48.95	**57.72**	51.75
	XPT	42.80	**51.05**	42.28	48.25
	TO	**46.74**	46.50	45.55	43.35
4	XPL	44.43	40.88	**62.07**	45.84
	XPT	55.57	**59.12**	37.93	54.16
	TO	44.22	45.13	40.32	**46.14**
5	XPL	50.95	52.62	**59.70**	45.13
	XPT	49.05	47.38	40.30	**54.87**
	TO	**45.88**	45.43	45.22	44.03
6	XPL	**58.32**	49.09	53.20	46.17
	XPT	41.68	50.91	46.80	**53.83**
	TO	46.26	46.61	**46.77**	44.51
7	XPL	57.56	47.88	**60.74**	49.44
	XPT	42.44	**52.12**	39.26	50.56
	TO	**46.16**	46.04	44.55	43.78
8	XPL	64.26	57.46	**67.51**	54.90
	XPT	35.74	42.54	32.49	**45.10**
	TO	45.15	45.93	41.14	**46.57**
9	XPL	**69.77**	49.47	65.15	51.94
	XPT	30.23	**50.53**	34.85	48.06
	TO	44.18	**47.03**	39.93	45.20
10	XPL	47.78	50.12	**50.74**	47.56
	XPT	52.22	49.88	49.26	**52.44**
	TO	44.86	**46.31**	46.10	43.68
11	XPL	45.80	**52.39**	48.54	46.16
	XPT	**54.20**	47.61	51.46	53.84
	TO	46.57	46.45	46.49	45.12
12	XPL	46.12	**49.08**	48.03	48.95
	XPT	**53.88**	50.92	51.97	51.05
	TO	45.76	46.54	**47.14**	44.36
13	XPL	46.61	48.05	**48.17**	47.96
	XPT	**53.39**	51.95	51.83	52.04
	TO	**46.99**	46.94	46.28	44.19
14	XPL	56.01	53.50	**65.67**	51.92
	XPT	43.99	46.50	34.33	**48.08**
	TO	45.78	**46.18**	40.69	45.80
15	XPL	57.40	51.08	**64.42**	57.17
	XPT	42.60	**48.92**	35.58	42.83
	TO	45.00	**46.57**	39.91	46.39
16	XPL	60.89	56.55	**68.74**	62.10
	XPT	39.11	**43.45**	31.26	37.90
	TO	46.76	**46.91**	41.27	45.89
17	XPL	38.08	38.38	34.09	**43.18**
	XPT	61.92	61.62	**65.91**	56.82
	TO	42.71	**45.06**	43.88	42.97
18	XPL	33.52	**38.09**	35.85	37.53
	XPT	**66.48**	61.91	64.15	62.47
	TO	41.43	44.07	**44.85**	42.73

TABLE 12. Mean XPL(%), mean XPT(%) and mean TO(%) of PCOBSCA, SCA, OBSCA, and COBSCA for functions 19-28 with D=30.

F		PCOBSCA	SCA	OBSCA	COBSCA
19	XPL	**48.41**	46.42	42.07	48.09
	XPT	51.59	53.58	**57.93**	51.91
	TO	46.39	**47.15**	44.60	44.63
20	XPL	57.41	**65.10**	9.46	60.63
	XPT	42.59	34.90	**90.54**	39.37
	TO	**47.34**	45.84	8.11	45.60
21	XPL	45.02	**47.59**	23.53	39.97
	XPT	54.98	52.41	**76.47**	60.03
	TO	**47.28**	45.36	38.60	44.59
22	XPL	**63.27**	53.52	62.66	53.32
	XPT	36.73	46.48	37.34	**46.68**
	TO	44.78	**48.03**	40.99	46.96
23	XPL	54.59	60.60	**64.43**	57.82
	XPT	**45.41**	39.40	35.57	42.18
	TO	**47.64**	45.64	39.95	44.87
24	XPL	59.93	59.68	**62.57**	52.94
	XPT	40.07	40.32	37.43	**47.06**
	TO	46.85	**47.10**	43.98	43.32
25	XPL	67.22	63.00	**67.55**	59.19
	XPT	32.78	37.00	32.45	**40.81**
	TO	45.47	**46.10**	42.90	44.85
26	XPL	54.39	49.57	**61.52**	44.71
	XPT	45.61	50.43	38.48	**55.29**
	TO	**46.68**	45.13	41.94	42.56
27	XPL	55.33	54.19	**65.87**	55.82
	XPT	44.67	**45.81**	34.13	44.18
	TO	45.40	**46.28**	43.95	43.71
28	XPL	54.43	50.07	**54.67**	49.96
	XPT	45.57	49.93	45.33	**50.04**
	TO	**46.76**	46.41	44.79	43.81

TABLE 13. Wilcoxon Signed Rank Test using TO values for PCOBSCA vs. SCA, OBSCA, and COBSCA.

Sl. No.	Pair	$p-values$	h
1	PCOBSCA vs. SCA	0.068498	0
2	PCOBSCA vs. OBSCA	0.000416	1
3	PCOBSCA vs. COBSCA	0.002111	1

whether the proposed method promotes *good convergence*, where *good convergence* is the algorithm's capacity of converging (fast or not) to optimal or near-optimal solutions. It is important to highlight that the convergence graphs for 50 and 100 dimensions are not presented here because they demonstrated to be similar to the 30D. Here, convergence graphs for two functions from each function type for all the algorithms are provided and these are unimodal functions f_1, f_5, multimodal functions f_6, f_{20}, and composite functions f_{21}, f_{28} with 30 dimensions. A particular random seed is used to generate the initial population and the same initial population is used for all the algorithms to make a fair comparison of the convergence. All the algorithms are executed with a maximum number of 3,00,000 function evaluations (FEs) and all of them terminate after the completion of this maximum number of FEs, and none of them terminates before the maximum number of FEs.

Fig. 1 presents the convergence characteristics of PCOBSCA, SCA, OBSCA, and COBSCA algorithms unimodal functions f_1, f_5, multimodal functions f_6, f_{20}, and composite functions f_{21}, f_{28} with 30 dimensions. Figs. 1(a) and 1(b) present the graphs related to the unimodal functions f_1 and f_5. In Fig. 1(a) for f_1, it is observed that PCOBSCA has better convergence characteristics than SCA, OBSCA, and COBSCA. From the convergence graph in Fig. 1(b) for f_5, all the algorithms have almost the same convergence characteristic. It is observed from Fig. 1(c) for f_6 that all the algorithms have almost the same convergence characteristic. Regarding the convergence graphs in Fig. 1(d) for f_{20}, PCOBSCA has better

TABLE 14. Mean XPL(%), mean XPT(%) and mean TO(%) of PCOBWOA, WOA, OBWOA, and COBWOA for functions 1-23 with D=30.

F		PCOBWOA	WOA	OBWOA	COBWOA
1	XPL	18.30	12.50	**19.60**	13.61
	XPT	81.70	**87.50**	80.40	86.39
	TO	34.74	28.86	**34.93**	29.16
2	XPL	14.22	16.34	**18.81**	9.69
	XPT	85.78	83.67	81.19	**90.31**
	TO	31.92	31.98	**34.27**	25.11
3	XPL	13.98	14.24	**18.34**	10.57
	XPT	86.02	85.76	81.66	**89.43**
	TO	31.50	29.68	**33.65**	26.39
4	XPL	**9.69**	6.68	8.60	7.50
	XPT	90.31	**93.32**	91.40	92.50
	TO	**26.65**	21.19	21.92	23.05
5	XPL	15.43	18.47	**19.27**	13.21
	XPT	84.57	81.53	80.74	**86.79**
	TO	32.51	32.79	**34.27**	28.94
6	XPL	16.83	16.16	**17.49**	11.52
	XPT	83.17	83.84	82.51	**88.48**
	TO	**33.81**	31.92	32.94	27.86
7	XPL	16.93	15.43	**17.05**	14.02
	XPT	83.07	84.57	82.95	**85.98**
	TO	**33.93**	31.22	33.11	30.24
8	XPL	19.29	**19.65**	15.82	9.00
	XPT	80.71	80.35	84.18	**91.00**
	TO	29.39	**34.69**	20.19	14.52
9	XPL	15.59	14.02	**17.95**	11.85
	XPT	84.41	85.98	82.05	**88.15**
	TO	**32.84**	30.90	32.10	27.58
10	XPL	18.17	13.85	**20.33**	12.43
	XPT	81.83	86.15	79.67	**87.57**
	TO	34.54	30.17	**35.27**	28.37
11	XPL	12.91	11.96	**16.00**	8.36
	XPT	87.09	88.04	84.00	**91.64**
	TO	**30.26**	27.43	31.89	23.43
12	XPL	**18.34**	13.58	13.33	10.17
	XPT	81.66	86.42	86.67	**89.83**
	TO	**35.11**	29.17	29.33	25.71
13	XPL	**18.53**	11.71	12.63	9.61
	XPT	81.47	88.29	87.37	**90.39**
	TO	**35.70**	27.08	28.66	25.65
14	XPL	13.85	14.68	**24.95**	14.87
	XPT	**86.15**	85.32	75.05	85.13
	TO	30.86	31.15	**35.56**	30.10
15	XPL	**21.25**	14.66	19.66	15.35
	XPT	78.75	**85.34**	80.34	84.65
	TO	**37.17**	30.89	32.49	30.87
16	XPL	**22.07**	15.75	13.92	6.70
	XPT	77.93	84.25	86.08	**93.30**
	TO	**34.12**	31.89	22.59	14.70
17	XPL	8.37	8.70	**9.74**	6.09
	XPT	91.63	91.30	90.26	**93.91**
	TO	25.41	23.45	**26.17**	20.33
18	XPL	**9.73**	5.92	9.30	6.08
	XPT	90.27	**94.08**	90.70	93.92
	TO	**27.51**	19.55	25.33	20.81
19	XPL	17.90	17.11	**19.02**	11.47
	XPT	82.10	82.89	80.98	**88.53**
	TO	**34.86**	32.44	34.32	27.49
20	XPL	11.76	16.47	**29.82**	12.12
	XPT	**88.24**	83.53	70.18	87.88
	TO	28.86	32.08	**37.57**	28.52
21	XPL	17.52	12.09	**17.69**	11.62
	XPT	82.48	87.91	82.31	**88.38**
	TO	**33.96**	28.29	33.42	27.17
22	XPL	15.88	12.60	**19.12**	14.14
	XPT	84.12	**87.40**	80.88	85.86
	TO	33.05	28.80	**34.37**	29.66
23	XPL	16.93	17.12	**24.32**	12.39
	XPT	83.07	82.88	75.68	**87.61**
	TO	**33.73**	32.60	34.20	27.97

TABLE 15. Mean XPL(%), mean XPT(%) and mean TO(%) of PCOBWOA, WOA, OBWOA, and COBWOA for functions 24-28 with D=30.

F		PCOBWOA	WOA	OBWOA	COBWOA
24	XPL	**21.12**	19.68	19.01	17.17
	XPT	78.88	80.32	80.99	**82.83**
	TO	**37.53**	34.71	34.31	33.25
25	XPL	**21.36**	13.89	20.40	17.06
	XPT	78.64	**86.11**	79.60	82.94
	TO	**37.71**	30.82	36.00	32.92
26	XPL	15.96	**17.89**	17.82	8.81
	XPT	84.04	82.11	82.18	**91.19**
	TO	**33.63**	33.49	33.17	24.20
27	XPL	18.17	18.66	**20.55**	14.44
	XPT	81.83	81.34	79.45	**85.56**
	TO	35.59	34.00	**35.79**	30.61
28	XPL	13.09	13.35	**15.23**	10.20
	XPT	86.91	86.65	84.77	**89.80**
	TO	30.38	28.87	**30.92**	25.67

TABLE 16. Wilcoxon Signed Rank Test using TO values for PCOBWOA vs. WOA, OBWOA, and COBWOA.

Sl. No.	Pair	$p-values$	h
1	PCOBWOA vs. WOA	0.000321	1
2	PCOBWOA vs. OBWOA	0.305498	0
3	PCOBWOA vs. COBWOA	0.000004	1

convergence behavior than all competitors. It is observed in Fig. 1(e) that the PCOBSCA presented an incremental convergence, exploring better the search space and consequently reaching the best results compared to all the competitors. As it can be seen, in the convergence graphs in Fig. 1(f) for f_{28}, the PCOBSCA converges similarly to the other algorithms. However, in the last iterations, the proposed algorithm was able to find even better solutions, outperforming all the competitors. Comparing the convergence graphs and best-error-runs results, among PCOBSCA, SCA, OBSCA, and COBSCA, it is observed that PCOBSCA has fast convergence leading it towards achieving a better quality of solutions. This analysis shows that the PCOBSCA is able to find promising and competitive solutions through an incremental convergence, balancing exploration and exploitation. Section VI-C presents a deeper investigation of the PCOBSCA's exploration/exploitation capacity in the 30D problems.

Fig. 2 presents the convergence graphs for PCOBWOA. For unimodal functions f_1 and f_5 (Figs. 2(a) & 2(f)), it is observed that there is no significant difference in convergence characteristics of PCOBWOA, WOA, OBWOA, and COBWOA. For function f_6, PCOBWOA has shown similar convergence characteristics to WOA, OBWOA, and COBWOA. For composition function f_{20}, WOA has better convergence than PCOBWOA whereas COBWOA has better convergence than PCOBWOA for other composition function f_{28}. When comparing the convergence graphs with best-error-runs results, WOA achieves better results than others for functions f_1, f_5, and f_6 but the difference with PCOBWOA

TABLE 17. Mean XPL(%), mean XPT(%) and mean TO(%) of PCOBAOA, AOA, OBAOA, COBAOA for functions 1-23 with D=30.

F		PCOBAOA	AOA	OBAOA	COBAOA
1	XPL	**0.61**	0.23	0.26	0.23
	XPT	99.39	**99.77**	99.74	**99.77**
	TO	**7.47**	4.67	4.92	4.73
2	XPL	**0.37**	0.28	0.28	0.28
	XPT	99.63	**99.72**	**99.72**	**99.72**
	TO	**5.41**	5.14	4.97	5.14
3	XPL	**0.51**	0.23	0.47	0.31
	XPT	99.49	**99.77**	99.53	99.69
	TO	**6.65**	4.52	6.55	5.43
4	XPL	**4.38**	0.37	0.72	0.32
	XPT	95.62	99.63	99.28	**99.68**
	TO	**17.96**	4.64	5.06	5.06
5	XPL	0.58	0.21	**0.66**	0.20
	XPT	99.42	99.79	99.34	**99.80**
	TO	6.95	4.31	**7.71**	4.38
6	XPL	**0.45**	0.26	0.30	0.24
	XPT	99.55	99.74	99.70	**99.76**
	TO	**6.36**	4.85	5.23	4.80
7	XPL	**0.60**	0.49	0.26	0.36
	XPT	99.40	99.51	**99.74**	99.64
	TO	**7.42**	6.73	4.82	5.99
8	XPL	**67.33**	16.08	55.69	32.26
	XPT	32.67	**83.92**	44.31	67.74
	TO	45.28	36.41	**49.35**	46.60
9	XPL	0.49	0.31	0.16	**0.51**
	XPT	99.51	99.69	**99.84**	99.49
	TO	5.24	5.14	3.28	**6.56**
10	XPL	**0.42**	0.31	0.34	0.26
	XPT	99.58	99.69	99.66	**99.74**
	TO	**5.98**	5.38	5.68	5.06
11	XPL	**0.64**	0.41	0.36	0.27
	XPT	99.36	99.59	99.64	**99.73**
	TO	**7.65**	6.24	5.74	5.15
12	XPL	**0.28**	0.21	0.26	0.22
	XPT	99.72	**99.79**	99.74	99.78
	TO	**4.97**	4.44	4.95	4.61
13	XPL	0.37	0.22	0.24	**0.46**
	XPT	99.63	**99.78**	99.76	99.54
	TO	5.55	4.50	4.66	**6.69**
14	XPL	**3.11**	2.20	1.77	2.68
	XPT	96.89	97.80	**98.23**	97.32
	TO	**16.78**	14.51	13.02	16.12
15	XPL	**31.08**	6.95	5.80	1.31
	XPT	68.92	93.05	94.20	**98.69**
	TO	**45.31**	24.37	22.26	10.24
16	XPL	**42.95**	22.26	19.33	30.27
	XPT	57.05	77.74	**80.67**	69.73
	TO	**48.44**	40.31	38.92	45.68
17	XPL	**0.54**	0.32	0.49	0.32
	XPT	99.46	**99.68**	99.51	99.68
	TO	**7.00**	5.54	6.85	5.65
18	XPL	0.21	0.16	**0.23**	0.19
	XPT	99.79	**99.84**	99.77	99.81
	TO	4.36	3.82	**4.60**	4.32
19	XPL	**0.31**	0.23	0.15	0.18
	XPT	99.69	99.77	**99.85**	99.82
	TO	**5.23**	4.65	3.65	4.20
20	XPL	8.43	32.73	26.37	**100.00**
	XPT	**91.57**	67.27	73.63	0.00
	TO	20.82	**20.86**	9.37	0.00
21	XPL	**0.31**	0.20	0.24	0.25
	XPT	99.69	**99.80**	99.76	99.75
	TO	**5.33**	4.30	4.65	4.91
22	XPL	1.66	0.61	0.83	**2.36**
	XPT	98.34	**99.39**	99.17	97.64
	TO	12.17	7.48	8.38	**15.12**
23	XPL	2.85	0.30	0.32	**17.74**
	XPT	97.15	**99.70**	99.68	82.26
	TO	13.34	4.71	3.30	**36.67**

TABLE 18. Mean XPL(%), mean XPT(%) and mean TO(%) of PCOBAOA, AOA, OBAOA, COBAOA for functions 24-28 with D=30.

F		PCOBAOA	AOA	OBAOA	COBAOA
24	XPL	0.30	0.29	0.10	**3.19**
	XPT	99.70	99.71	**99.90**	96.81
	TO	5.03	5.12	0.73	**17.54**
25	XPL	1.85	1.20	0.85	**2.34**
	XPT	98.15	98.80	**99.15**	97.66
	TO	12.70	10.20	8.72	**15.06**
26	XPL	0.36	0.19	0.31	**4.46**
	XPT	99.64	**99.81**	99.69	95.54
	TO	5.38	4.03	2.84	**20.60**
27	XPL	0.81	0.34	0.12	**1.01**
	XPT	99.19	99.66	**99.88**	98.99
	TO	7.38	4.67	1.34	**9.87**
28	XPL	**0.30**	0.28	0.25	0.26
	XPT	99.70	99.72	**99.75**	99.74
	TO	4.71	4.76	4.66	**5.01**

TABLE 19. Wilcoxon Signed Rank Test using TO values for PCOBAOA vs. AOA, OBAOA, and COBAOA.

Sl. No.	Pair	$p-values$	h
1	PCOBAOA vs. AOA	0.000007	1
2	PCOBAOA vs. OBAOA	0.000130	1
3	PCOBAOA vs. COBAOA	0.386866	0

is not statistically significant for f_5. COBWOA and WOA have shown better convergence and achieved a better quality of results respectively for f_{20} and f_{21}. A similar observation can made with PCOBWOA for f_{28}.

Fig. 3 presents the convergence graphs for PCOBAOA. Regarding the unimodal function f_1, COBAOA and PCOBAOA reached better convergence and final fitness values whereas AOA and PCOBAOA have the same characteristics for another unimodal function f_5. It is observed in Fig. 3(c) that PCOBAOA has better convergence and final fitness values than others. From Fig. 3(d) for basic multimodal function f_{20}, it is observed that PCOBAOA has better convergence characteristics than AOA, OBAOA, and COBAOA. For composite functions f_{21} and f_{28}, PCOBAOA has shown similar convergence characteristics to AOA, OBAOA, and COBAOA. Comparing the best-error-runs results and convergence graphs, it has been observed that PCOBAOA has achieved better convergence and results for f_1, and f_{20}. PCOBAOA has better convergence and result than OBAOA and COBAOA for f_5 whereas AOA achieves better convergence and result than PCOBAOA for the same function. For f_6, PCOBAOA has shown better convergence and its result is statistically better than other algorithms. It is observed that PCOBAOA, AOA, OBAOA, and COBAOA have shown similar convergence and there is no statistically significant difference in their results for f_{21}. In convergence graphs for function f_{28}, PCOBAOA achieves better fitness values than AOA, and OBAOA but not better than COBAOA. If we compare it with the results, it can be observed that

TABLE 20. Computation complexities (in seconds) of algorithms for function f_{14} with dimension = 30, 50, 100.

		30D			50D			100D		
Algorithm	T_0	T_1	T_2	$(T_2-T_1)/T_0$	T_1	T_2	$(T_2-T_1)/T_0$	T_1	T_2	$(T_2-T_1)/T_0$
PCOBSCA	0.087399	0.856036	1.5872	8.3667	1.120184	2.2023	12.3826	1.798526	3.4606	19.0192
SCA	-	-	1.8994	11.9379	-	2.5980	16.9110	-	4.3980	29.7456
OBSCA	–	–	1.4666	6.9859	-	1.9883	9.9340	-	3.2157	16.2171
COBSCA	–	–	1.6936	9.5842	-	2.3589	14.1744	-	3.9381	24.4834
PCOBWOA	-	-	1.5274	7.6826	-	2.0183	10.2769	-	3.2412	16.5082
WOA	-	-	1.5711	8.1824	-	2.1528	11.8159	-	3.4064	18.3986
OBWOA	-	-	1.3778	5.9706	-	1.9052	8.9834	-	3.0248	14.0322
COBWOA	-	-	1.4327	6.5984	-	1.9576	9.5830	-	3.1196	15.1172
PCOBAOA	-	-	1.7281	9.9785	-	2.2144	12.5213	-	3.4050	18.3832
AOA	-	-	1.6187	8.7271	-	2.1387	11.6549	-	3.2889	17.0538
OBAOA	-	-	1.4450	6.7395	-	1.9211	9.1653	-	3.0186	13.9608
COBAOA	-	-	1.9432	12.4403	-	2.5558	16.4276	-	3.8242	23.1796

TABLE 21. Problem Descriptions.

Problem No.	Problem	Variables
P07	Spread spectrum radar polly phase code design	20
P10	Circular antenna array design problem	12
P12	Messenger: Spacecraft trajectory optimization problem	26
P13	Cassini 2: Spacecraft trajectory optimization problem	22

TABLE 22. Best, worst, median, mean and standard deviation of objective values from PCOBSCA, SCA, OBSCA, and COBSCA for real-world optimization problems.

Problem		PCOBSCA	SCA	OBSCA	COBSCA
P07	Best	1.03	1.87	**0.82**	1.16
	Worst	2.00	2.74	**1.48**	2.11
	Median	1.50	2.14	**1.21**	1.64
	Mean	1.49	2.16	**1.19**	1.66
	Std. Dev.	0.26	0.21	**0.15**	0.19
	p-value	NA	0.000014	0.000296	0.009417
	h	NA	1	-1	1
P10	Best	**-21.00**	-12.0552	-9.2144	-20.02
	Worst	**-20.08**	-10.07	-8.25	-10.27
	Median	**-20.78**	-10.85	-8.71	-11.50
	Mean	**-20.75**	-10.86	-8.70	-12.49
	Std. Dev.	**0.21**	0.45	0.2941	2.55
	p-value	NA	0.000012	0.000012	0.000012
	h	NA	1	1	1
P12	Best	**24.36**	31.73	24.70	25.82
	Worst	**38.95**	53.85	68.90	41.50
	Median	**29.14**	42.71	49.85	34.51
	Mean	**30.39**	42.64	49.85	34.34
	Std. Dev.	**4.11**	6.03	11.15	4.27
	p-value	NA	0.00001229	0.00001774	0.004926937
	h	NA	1	1	1
P13	Best	**25.50**	26.47	46.00	28.33
	Worst	**41.94**	47.71	62.17	47.02
	Median	**38.01**	41.34	54.05	36.74
	Mean	**36.85**	40.16	53.03	38.08
	Std. Dev.	**4.12**	5.92	4.52	5.97
	p-value	NA	0.037043378	0.00001229	0.00001229
	h	NA	1	1	1

PCOBAOA statistically outperforms AOA, and OBAOA whereas there is no statistically significant difference with COBAOA.

TABLE 23. Best, worst, median, mean and standard deviation of objective values from PCOBWOA, WOA, OBWOA, and COBWOA for real-world optimization problems.

Problem		PCOBWOA	WOA	OBWOA	COBWOA
P07	Best	**0.68**	1.48	1.39	0.84
	Worst	**1.61**	2.07	2.45	2.36
	Median	**1.15**	1.83	1.72	1.40
	Mean	**1.12**	1.83	1.79	1.46
	Std. Dev.	0.23	**0.15**	0.30	0.29
	p-value	NA	0.000012	0.000012	0.000602
	h	NA	1	1	1
P10	Best	**-21.13**	-12.06	-12.94	-18.49
	Worst	**-10.38**	-9.64	-7.55	-9.64
	Median	**-12.50**	-10.92	-10.81	-11.87
	Mean	**-14.09**	-10.94	-10.98	-12.47
	Std. Dev.	3.57	**0.87**	1.13	2.34
	p-value	NA	0.000072	0.000032	0.054374
	h	NA	1	1	0
P12	Best	31.35	24.67	**21.46**	33.94
	Worst	47.88	54.08	**45.97**	57.17
	Median	**33.67**	35.92	34.71	46.92
	Mean	**34.85**	38.03	35.58	46.78
	Std. Dev.	**3.73**	7.09	6.4369	5.01
	p-value	NA	0.000025	0.000029	0.000493
	h	NA	1	1	1
P13	Best	**18.21**	28.51	36.39	35.93
	Worst	**35.10**	53.95	47.30	48.97
	Median	**25.26**	39.72	40.89	42.59
	Mean	**25.50**	39.73	42.09	41.98
	Std. Dev.	4.60	6.14	2.89	**3.57**
	p-value	NA	0.000016	0.000029	0.000101
	h	NA	1	1	1

From the analysis of convergence graphs of PCOBSCA, PCOBWOA, and PCOBAOA, the PCOBL strategy becomes successful in achieving better convergence characteristics and final objective values by these algorithms for most of the problems when PCOBL is incorporated in SCA, WAO, and AOA leading to obtaining a better quality of solutions. To achieve a better quality of solutions, a metaheuristic algorithm needs to have better convergence characteristics. PCOBL is capable of enhancing the convergence characteristics and performance of SCA, WOA, and AOA when it is incorporated in these basic algorithms.

TABLE 24. Best, worst, median, mean and standard deviation of objective values from PCOBAOA, AOA, OBAOA, and COBAOA for real-world optimization problems.

Problem		PCOBAOA	AOA	OBAOA	COBAOA
P07	Best	1.62	1.52	**1.35**	1.54
	Worst	**1.88**	2.05	2.21	2.05
	Median	**1.80**	1.95	1.93	1.86
	Mean	**1.78**	1.89	1.90	1.86
	Std. Dev.	**0.07**	0.14	0.19	0.13
	p-value	NA	0.005816	0.005816	0.003216
	h	NA	1	1	1
P10	Best	-12.57	-12.24	**-12.94**	-11.82
	Worst	-9.33	-9.86	-7.55	**-9.94**
	Median	**-11.43**	-10.99	-10.81	-10.69
	Mean	**-11.22**	-10.96	-10.98	-10.90
	Std. Dev.	0.80	0.64	1.13	**0.63**
	p-value	NA	0.121828	0.142532	0.121828
	h	NA	0	0	0
P12	Best	**31.35**	31.54	35.98	33.94
	Worst	**47.88**	59.33	61.29	57.17
	Median	**33.67**	45.48	45.51	46.92
	Mean	**34.85**	44.76	46.12	46.78
	Std. Dev.	**3.73**	6.13	5.42	5.01
	p-value	NA	0.000023	0.000032	0.000014
	h	NA	1	1	1
P13	Best	**32.66**	36.38	36.39	35.93
	Worst	**41.93**	51.02	47.30	48.97
	Median	**37.22**	42.35	40.89	42.59
	Mean	**36.71**	42.59	42.09	41.98
	Std. Dev.	**2.55**	3.30	2.89	3.57
	p-value	NA	0.000016	0.000020	0.000081
	h	NA	1	1	1

C. EXPLORATION AND EXPLOITATION ANALYSIS

Metaheuristics need to perform a good balance, or *trade-off* (TO), between exploration and exploitation during the search process to achieve a good performance, i.e., good quality of solutions [46], and this principle is widely accepted among researchers [47]. The search agents with the best solutions in meta-heuristic algorithms tend to direct the search process towards them. Therefore, the exploitation effect increases when the distance among the search agents decreases. On the other hand, the exploration effect increases when the distance among the search agents increases. The mechanisms to calculate the exploration (XPL) and exploitation (XPT) are adopted from the article [47]. These mechanisms are also used to analyze the exploration and exploitation of opposition-based SSAs in the study [21]. The population diversity is measured to calculate the increments and decrements in the distance among the search agents in time t as follows:

$$div_j(t) = \frac{1}{N}\sum_{i=1}^{N}\left|median(x_j(t)) - x_{ij}(t)\right|, \tag{11}$$

where $median(x_j(t))$ is the median of jth elements in the whole population. $x_{ij}(t)$ is the jth element in the position of ith search agent. N is the population size. $div_j(t)$ indicates the diversity along the jth dimension. The population diversity $div(t)$ is defined by:

$$div(t) = \frac{1}{D}\sum_{j=1}^{D} div_j(t), \tag{12}$$

where D is the dimension.

After the end of the maximum iteration, the maximum diversity among all iterations is computed. Then, the percentage of exploration and exploitation in the iteration t is defined as:

$$XPL\%(t) = \left(\frac{div(t)}{div_{max}}\right) \times 100, \tag{13}$$

$$XPT\%(t) = \left(\frac{|div(t) - div_{max}|}{div_{max}}\right) \times 100. \tag{14}$$

The higher values of XPL and XPT indicate higher exploration and exploitation. To calculate the balance between exploration and exploitation, we incorporate the following formula:

$$TO\%(t) = \sqrt{XPL\%(t) \times XPT\%(t)}. \tag{15}$$

The higher value of TO indicates a higher balance between exploration and exploitation. There is a correlation between XPL, XPT, or TO and the quality of solutions. A higher XPL value indicates a higher exploration during the search process, which helps the algorithms in global search, i.e., exploring the unexplored regions of the search space which may contain the optimal solution. A higher XPT value indicates higher exploitation during the search process, which helps the algorithms in local search, i.e, fine-tuning the existing solutions. The metaheuristics need both global and local search but in a well-balanced way. Therefore, a good *trade-off* of exploration and exploitation is needed to achieve a better quality of solutions. TO is an estimation of this *trade-off*. Finally, we calculated the average of $XPL\%(t)$, $XPT\%(t)$, and $TO\%(t)$ over all the iterations obtained from PCOBSCA, SCA, OBSCA, and COBSCA. These results can be seen in Tables 11 and 12.

Regarding the unimodal functions, it can be observed in Table 11 that OBSCA reached the highest exploration results except for function f_1, COBSCA achieved the highest exploitation results for functions f_1, f_2, and f_5, and SCA achieved the highest exploitation result for functions f_3, and f_4. SCA, OBSCA, and COBSCA provided the best TO values for unimodal functions f_2, f_1, and f_4 respectively whereas PCOBSCA provided the best TO values for unimodal functions f_3, and f_5. Although the POBSCA has not reached the best TO values for f_2, f_1, and f_4, its results are competitive.

For most of the multimodal functions (see Tables 11 and 12), the OBSCA presented the best results for exploration. Regarding exploitation, the SCA, and COBSCA presented the highest values. Regarding TO, the SCA also presented the highest values. It shows that these algorithms

TABLE 25. Mean and standard deviation (in parenthesis) of best-error-runs over 51 independent runs of PCOBSCA, SCA, OBSCA, and COBSCA for 50D problems.

F	PCOBSCA	SCA	h	OBSCA	h	COBSCA	h
1	**79.57** (71.91)	26706.39 (3850.74)	1	60673.55 (7877.83)	1	929.29 (426.96)	1
2	**29217335.25** (15274301.81)	436607581.8 (100164632)	1	1734200042 (437950511.8)	1	53727686.31 (32939116.18)	1
3	**1112673023** (1019521977)	80146658402 (16552537652)	1	4.49769E+14 (1.34811E+15)	1	1.14343E+11 (8366418636)	1
4	80539.84 (8540.26)	**55358.05** (7514.73)	-1	269219.87 (180266.80)	1	72332.81 (7342.13)	-1
5	**303.21** (187.98)	3075.19 (729.80)	1	14186.33 (2963.26)	1	1363.64 (585.38)	1
6	**56.06** (8.77)	1783.57 (323.32)	1	6441.12 (1583.42)	1	105.41 (33.15)	1
7	**24.8808** (6.54)	185.26 (31.82)	1	6404.11 (7358.98)	1	61.33 (13.48)	1
8	21.15 (0.05)	**21.13** (0.04)	-1	21.46 (0.06)	1	21.14 (0.037)	0
9	**24.69** (3.80)	72.54 (1.51)	1	62.68 (2.31)	1	36.6 (5.40)	1
10	**107.13** (41.95)	3354.03 (518.01)	1	8812.32 (1244.72)	1	304.02 (114.70)	1
11	**36.07** (8.69)	670.19 (45.68)	1	1132.60 (71.90)	1	130.98 (29.48)	1
12	**122.98** (38.81)	716.37 (39.40)	1	1206.52 (78.72)	1	224.86 (43.24)	1
13	**169.04** (50.21)	727.82 (47.35)	1	1246.06 (98.21)	1	307.33 (43.00)	1
14	**2233.58** (535.81)	13202.04 (484.54)	1	12277.97 (613.94)	1	9488.61 (2039.80)	1
15	**7129.88** (2957.86)	14232.65 (349.00)	1	13268.37 (577.81)	1	11668.68 (2119.96)	1
16	3.42 (0.34)	3.37 (0.23)	0	4.24 (0.49)	1	**3.27** (0.28)	-1
17	**202.54** (35.61)	947.07 (77.16)	1	1379.26 (65.36)	1	319.85 (38.46)	1
18	**390.24** (13.99)	953.42 (76.26)	1	1422.13 (53.37)	1	425.25 (26.65)	1
19	**28.73** (23.94)	27965.94 (26364.71)	1	1472074.29 (427137.14)	1	3471.10 (4433.14)	1
20	**21.81** (0.55)	23.73 (0.45)	1	24.95 (0.11)	1	24.20 (0.36)	1
21	**906.03** (188.40)	3818.05 (137.74)	1	4424.12 (78.01)	1	1265.01 (409.08)	1
22	**2050.48** (657.76)	14306.89 (420.27)	1	14434.42 (573.03)	1	8615.40 (2103.43)	1
23	**5480.11** (1689.03)	14956.75 (408.92)	1	14945.83 (771.43)	1	13828.69 (1663.24)	1
24	**238.25** (6.85)	421.04 (7.15)	1	523.11 (28.03)	1	276.57 (11.30)	1
25	**318.87** (10.01)	444.58 (6.32)	1	583.79 (24.92)	1	359.58 (11.29)	1
26	**269.04** (69.16)	316.03 (114.73)	0	294.98 (23.78)	1	257.38 (81.57)	0
27	**691.26** (156.09)	2287.42 (55.70)	1	2726.07 (125.47)	1	1153.63 (129.57)	1
28	**527.31** (414.62)	4692.83 (722.92)	1	9275.97 (806.03)	1	747.79 (253.68)	1

may be less susceptible to get trapped in local optima. The POBSCA has achieved the best TO values for functions f_7, and f_{13}, and f_{20}, but, for other multimodal functions, the results are competitive compared to SCA. Regarding the composition functions (see Table 12), the OBSCA and COBSCA reached the best values for exploration and exploitation, respectively. Finally, the POBSCA and SCA reached the best TO values.

To complement the previous analysis, we have performed the pair-wise Wilcoxon Signed Rank Test with a significance level (α) = 0.05 to check whether there is any statistically significant difference in the TO values of the algorithms considering all the benchmark functions together. The test results are given in Table 13. In this table, $h = 1$ indicates that TO values of PCOBSCA are statistically better than the other in the pair. On the other hand, $h = 0$ indicates no

TABLE 26. Mean and standard deviation (in parenthesis) of best-error-runs over 51 independent runs of PCOBSCA, SCA, OBSCA, and COBSCA for 100D problems.

F	PCOBSCA	SCA	h	OBSCA	h	COBSCA	h
1	**498.39** (263.44)	96018.55 (6355.79)	1	164543.43 (15145.40)	1	4821.87 (1795.15)	1
2	**66339182.29** (28017812.81)	2390862308 (483892175)	1	7581447366 (1367526590)	1	143922485.1 (65121016.4)	1
3	**24229100870** (7687196717)	7.12E+16 (2.09E+17)	1	7.58624E+22 (1.96503E+23)	1	1.3882E+11 (70865167235)	1
4	198346.71 (13755.76)	**153706.50** (10552.83)	-1	2794531.86 (6688719.06)	1	168795.12 (12751.61)	-1
5	**856.32** (248.32)	13981.77 (1726.57)	1	69212.51 (10023.04)	1	6146.41 (2328.15)	1
6	**406.55** (54.24)	17001.87 (2090.11)	1	41203.44 (3528.86)	1	765.29 (130.71)	1
7	**60.71** (9.09)	85020.89 (55692.74)	1	108132707.10 (98101399.10)	1	138.98 (25.18)	1
8	21.30 (0.04)	**21.30** (0.03)	0	21.53 (0.04)	1	21.30 (0.03)	0
9	**57.51** (6.79)	160.68 (2.07)	1	147.55 (3.91)	1	86.06 (7.46)	1
10	**427.83** (173.76)	13432.02 (1441.93)	1	27364.10 (3032.63)	1	1315.36 (401.59)	1
11	**116.37** (17.84)	1863.32 (103.51)	1	3062.08 (178.76)	1	377.47 (61.70)	1
12	**254.43** (37.61)	2022.30 (111.44)	1	3104.84 (147.54)	1	494.12 (66.16)	1
13	**414.25** (59.40)	2019.90 (106.96)	1	3143.72 (174.27)	1	706.03 (81.10)	1
14	**6759.40** (1241.36)	30228.84 (855.16)	1	25826.86 (528.86)	1	19875.02 (4006.26)	1
15	**11803.39** (3038.66)	30790.95 (574.66)	1	27761.19 (1057.46)	1	21528.45 (4561.96)	1
16	4.12 (0.24)	3.96 (0.23)	-1	**3.85** (0.39)	-1	4.03 (0.24)	0
17	**385.74** (47.97)	2661.66 (138.58)	1	3315.10 (46.37)	1	746.38 (88.02)	1
18	**919.19** (29.95)	2702.69 (136.09)	1	3402.35 (52.67)	1	1051.06 (55.24)	1
19	**628.50** (1136.01)	515865.05 (332236.71)	1	5184888.74 (429359.68)	1	81325.84 (93202.47)	1
20	50 (3.50314E-13)	50 (1.24E-11)	0	50 (0)	0	50 (2.26055E-12)	0
21	**501.12** (32.42)	7694.42 (188.42)	1	8573.75 (108.33)	1	1605.14 (911.15)	1
22	**7082.91** (1087.72)	32117.43 (694.89)	1	30433.18 (685.84)	1	23820.39 (4364.80)	1
23	**13078.32** (3075.44)	32459.85 (650.38)	1	32342.10 (770.28)	1	26822.33 (4080.99)	1
24	**285.64** (11.65)	732.51 (22.60)	1	1368.06 (161.30)	1	380.48 (18.82)	1
25	**452.29** (14.53)	772.87 (14.16)	1	1224.47 (48.67)	1	562.08 (17.81)	1
26	**418.09** (14.99)	719.45 (5.87)	1	638.66 (96.25)	1	490.75 (14.59)	1
27	**1276.70** (175.34)	4846.39 (89.41)	1	6036.76 (262.03)	1	2181.08 (191.37)	1
28	**2891.51** (577.05)	15158.69 (642.70)	1	26684.78 (3104.63)	1	5530.91 (1336.31)	1

significant difference in the TO values of the algorithms in the pair. From this table, it is observed that PCOBSCA has statistically better TO values than OBSCA, and COBSCA, whereas there is no significant difference in the TO values of PCOBSCA, SCA, and MSCA. If we simultaneously analyze the TO values and solutions optimized by PCOBSCA, it may be concluded that PCOBSCA has an effective good balance between exploration and exploitation.

The average of $XPL\%(t)$, $XPT\%(t)$, and $TO\%(t)$ over all the iterations obtained from PCOBWOA, WOA, OBWOA, and COBWOA is tabulated in Tables 14 and 15. The Wilcoxon Signed Rank Test results are given in Table 16. It is observed that TO values of PCOBWOA are higher statistically significant than WOA, and COBWOA. It indicates that PCOBWOA has a better trade-off between exploration and exploitation than that of WOA, and COBWOA. It is also

TABLE 27. Mean and standard deviation (in parenthesis) of best-error-runs over 51 independent runs of PCOBWOA, WOA, OBWOA, and COBWOA for 50D problems.

F	PCOBOWA	WOA	h	OBWOA	h	COBWOA	h
1	1184.40 (1492.79)	**2.78** (2.22)	-1	24.77 (10.65)	-1	1716.87 (1663.87)	0
2	**34349037.56** (18398854.97)	38271144.66 (11793045.14)	0	41343631.76 (13538720.67)	1	47174693.77 (16802544.03)	1
3	**15700276611** (6912359552)	20743401501 (12935512048)	1	18715947404 (8640068527)	0	27100848987 (8374590817)	1
4	**33048.09** (11528.75)	39655.46 (6860.21)	1	58541.15 (12233.54)	1	38625.59 (10972.75)	1
5	207.10 (140.65)	**105.33** (24.59)	-1	152.57 (45.21)	0	402.32 (244.08)	1
6	250.13 (102.50)	**135.58** (60.69)	-1	153.31 (64.86)	-1	291.52 (87.91)	1
7	**97.20** (13.25)	679.38 (1785.77)	1	145.28 (21.71)	1	117.91 (21.26)	1
8	21.38 (0.06)	**21.12** (0.04)	-1	21.39 (0.06)	0	21.38 (0.04)	0
9	**51.86** (5.03)	68.77 (4.06)	1	61.59 (4.02)	1	58.87 (4.25)	1
10	379.65 (168.21)	**135.18** (54.37)	-1	162.77 (42.61)	-1	516.27 (229.09)	1
11	**553.47** (74.40)	781.86 (104.50)	1	755.31 (85.55)	1	731.98 (91.06)	1
12	**635.48** (113.27)	943.34 (164.32)	1	865.79 (122.36)	1	816.15 (115.33)	1
13	**572.73** (78.30)	960.90 (136.95)	1	727.10 (88.52)	1	696.24 (112.49)	1
14	**4937.30** (924.26)	8781.19 (1509.17)	1	8834.30 (1435.96)	1	7250.75 (1492.78)	1
15	**8073.26** (1020.79)	10906.05 (1226.72)	1	10907.71 (1460.88)	1	8650.04 (952.35)	1
16	1.93 (0.67)	2.45 (0.52)	1	2.90 (1.07)	1	**1.82** (0.70)	0
17	**523.18** (90.79)	1074.94 (108.26)	1	878.62 (154.29)	1	766.28 (124.24)	1
18	**642.55** (88.83)	1121.07 (122.77)	1	905.12 (134.97)	1	674.94 (119.61)	0
19	137.24 (366.71)	141.04 (39.36)	1	**51.79** (8.37)	0	248.83 (532.87)	1
20	23.70 (1.06)	24.49 (0.19)	1	23.55 (0.85)	0	**23.28** (1.30)	0
21	1481.82 (407.47)	**900.22** (281.01)	-1	2321.44 (591.28)	1	2143.65 (610.87)	1
22	**8853.73** (1444.99)	11377.72 (1298.03)	1	10613.62 (1081.82)	1	9440.11 (1264.18)	1
23	**10634.35** (1271.83)	12750.04 (1439.39)	1	12298.66 (1171.94)	1	11159.40 (1574.23)	0
24	**360.35** (17.86)	405.14 (13.88)	1	395.51 (14.42)	1	388.94 (15.32)	1
25	**405.58** (21.03)	424.12 (17.90)	1	422.78 (19.11)	1	423.78 (20.53)	1
26	**396.14** (91.38)	460.01 (76.38)	1	437.23 (78.66)	1	426.44 (82.60)	1
27	**1775.35** (123.83)	2192.11 (126.36)	1	2028.87 (108.89)	1	2022.69 (127.51)	1
28	**4170.07** (1640.25)	7577.30 (1533.60)	1	6267.60 (1893.99)	1	5962.01 (1585.91)	1

observed that the difference between TO values of PCOBWOA and OBWOA is not statistically significant.

The average of $XPL\%(t)$, $XPT\%(t)$, and $TO\%(t)$ over all the iterations obtained from PCOBAOA, AOA, OBAOA, and COBAOA are tabulated in Tables 17 and 18. The Wilcoxon Signed Rank Test results are given in Table 19. The TO values of PCOBAOA has a higher statistical significance than AOA, and OBAOA whereas there is no significant difference in TO values of PCOBAOA and COBAOA.

After analyzing the best-error runs and TO values of PCOBSCA, PCOBWOA, and PCOBAOA, it is observed that these algorithms maintain an effective balance between exploration and exploitation during the search process, which results in better performance, i.e., achieving a better quality

TABLE 28. Mean and standard deviation (in parenthesis) of best-error-runs over 51 independent runs of PCOBWOA, WOA, OBWOA, and COBWOA for 100D problems.

F	PCOBOWA	WOA	h	OBWOA	h	COBWOA	h
1	5414.71 (3492.24)	**9.83** (6.39)	-1	105.75 (30.98)	-1	5733.85 (4263.14)	0
2	**87907805.83** (25394596.09)	90722579.56 (16362659.47)	0	94187705.55 (18388317.61)	0	118516478.7 (25778423.26)	1
3	1.6757E+11 (1.18691E+11)	**67616847046** (22147175407)	-1	68054453342 (18256121070)	-1	4.08883E+11 (5.7686E+11)	1
4	**163469.76** (17146.78)	337162.30 (103098.73)	1	198529.90 (29496.28)	1	177945.55 (15306.39)	1
5	658.66 (425.77)	**183.90** (24.37)	-1	287.51 (50.68)	-1	1079.94 (929.09)	1
6	981.12 (334.16)	**449.45** (88.85)	-1	481.54 (77.91)	-1	1062.55 (309.55)	0
7	**688.86** (553.87)	350747.16 (457509.85)	1	1985.22 (2890.70)	1	3165.07 (2780.45)	1
8	21.42 (0.05)	**21.29** (0.04)	-1	21.46 (0.05)	1	21.43 (0.05)	0
9	**120.19** (8.68)	152.95 (5.07)	1	139.06 (6.38)	1	132.20 (8.25)	1
10	1119.10 (300.17)	**258.38** (62.14)	-1	376.19 (89.34)	-1	1457.44 (411.30)	1
11	**1546.65** (159.17)	2118.44 (151.16)	1	1935.16 (187.06)	1	1960.51 (161.68)	1
12	**1741.00** (210.99)	2300.80 (178.77)	1	2313.98 (231.19)	1	2175.39 (181.69)	1
13	**1382.97** (154.33)	2552.61 (229.81)	1	1823.77 (223.93)	1	1658.34 (214.89)	1
14	**13104.54** (1857.16)	20922.01 (3632.59)	1	18694.59 (2718.14)	1	14339.28 (2280.13)	1
15	**15739.00** (1466.96)	22644.96 (2657.18)	1	19708.66 (2511.11)	1	17158.91 (1644.22)	1
16	2.33 (0.67)	3.01 (0.48)	1	3.39 (1.14)	1	**2.26** (0.55)	0
17	**1494.63** (210.45)	2829.29 (146.34)	1	2329.39 (267.12)	1	2008.48 (261.08)	1
18	**1653.72** (175.56)	2828.55 (175.45)	1	2317.32 (323.24)	1	1816.95 (223.50)	1
19	602.87 (936.84)	493.95 (127.69)	0	**141.26** (17.87)	-1	1405.14 (3113.92)	0
20	50.00 (0.00)	50.00 (0.00)	0	**49.99** (0.07)	0	50.00 (0.00)	0
21	1434.20 (523.90)	**443.34** (54.14)	-1	1007.44 (745.87)	-1	1755.28 (907.35)	1
22	**17886.39** (2976.29)	24456.11 (2766.81)	1	22794.80 (1663.00)	1	18620.67 (1699.93)	1
23	**22676.13** (1667.32)	28575.15 (2292.97)	1	26281.06 (2382.57)	1	24057.20 (1665.97)	1
24	**581.29** (28.23)	656.06 (19.90)	1	639.17 (27.35)	1	635.33 (28.59)	1
25	**675.31** (34.52)	712.22 (32.51)	1	705.22 (32.75)	1	703.92 (36.84)	1
26	**605.51** (20.23)	696.72 (16.13)	1	661.38 (23.59)	1	649.62 (22.61)	1
27	**3698.76** (230.47)	4613.58 (204.31)	1	4329.70 (270.47)	1	4194.36 (183.97)	1
28	**14976.74** (1386.06)	21126.04 (2345.47)	1	18230.78 (1651.28)	1	17698.44 (1290.75)	1

of solutions compared to the counterparts of these algorithms. Now, it may be concluded that PCOBL can provide an effective balance between exploration and exploitation when it is combined with metaheuristic algorithms. PCOBL provides a set of diverse partial opposite solutions, whereas other OBL schemes provide only single opposite solutions. Therefore, PCOBL not only enhances the diversification in the search space but also enhances the chances of obtaining better solutions when PCOBL is incorporated in metaheuristics.

D. COMPUTATIONAL TIME COST ANALYSIS

The computation cost time of each algorithm is calculated, in every dimension, using the CEC2013 criteria, and they are presented in Table 20. The criteria are as follows:

TABLE 29. Mean and standard deviation (in parenthesis) of best-error-runs over 51 independent runs of PCOBAOA, AOA, OBAOA, and COBAOA for 50D problems.

F	PCOBAOA	AOA	h	OBAOA	h	COBAOA	h
1	**63769.40** (8923.68)	70658.68 (7810.82)	1	70805.89 (8397.29)	1	70719.49 (6658.52)	1
2	1231136849 (427821759.8)	**1169642236** (464076724.8)	0	1256404707 (598012546.2)	0	1591828357 (593607441.8)	1
3	**1.10444E+13** (3.29816E+13)	9.51997E+13 (1.75942E+14)	1	6.55302E+13 (1.99739E+14)	1	8.98474E+13 (3.282E+14)	1
4	**78182.11** (5992.25)	88015.83 (8942.69)	1	88815.64 (7455.95)	1	86836.63 (9646.57)	1
5	**4195.52** (346.91)	3504.53 (280.58)	-1	4165.09 (391.82)	0	4415.85 (317.11)	1
6	**5849.85** (1577.52)	7622.24 (1407.99)	1	6934.71 (1648.76)	1	7491.47 (1634.54)	1
7	**1283.31** (1504.51)	19980.80 (68048.27)	1	9138.04 (19407.97)	1	8394.13 (15905.97)	1
8	21.14 (0.04)	**21.13** (0.04)	0	21.14 (0.05)	0	21.13 (0.04)	0
9	**68.92** (2.57)	69.61 (3.04)	0	69.02 (2.79)	0	69.53 (2.79)	0
10	**8705.41** (1952.72)	10471.92 (1533.76)	1	10551.97 (1551.18)	1	10624.27 (2071.98)	1
11	**955.30** (88.24)	1074.07 (111.80)	1	1047.56 (95.66)	1	1029.80 (92.65)	1
12	**1033.74** (75.87)	1124.14 (92.27)	1	1080.78 (120.57)	0	1048.50 (96.18)	0
13	**1085.87** (91.20)	1131.01 (99.40)	1	1125.42 (97.61)	1	1124.92 (92.17)	0
14	**11575.24** (842.24)	13492.32 (598.98)	1	13166.61 (640.94)	1	13334.67 (652.17)	1
15	14221.01 (378.08)	**13928.12** (547.47)	-1	14096.44 (580.02)	0	14083.87 (493.75)	0
16	3.35 (0.27)	**3.33** (0.22)	0	3.46 (0.28)	1	3.38 (0.29)	0
17	**1129.87** (49.83)	1331.31 (61.54)	1	1332.03 (75.98)	1	1333.84 (56.80)	1
18	**1201.75** (108.96)	1319.31 (58.23)	1	1290.77 (70.78)	1	1292.32 (83.47)	1
19	142248.95 (63544.27)	**4547.66** (18777.35)	-1	133633.67 (38526.73)	0	125345.66 (33175.97)	0
20	**24.35** (0.40)	24.47 (0.27)	0	24.53 (0.22)	1	24.40 (0.37)	0
21	**4288.00** (112.69)	4339.94 (57.88)	1	4389.24 (83.10)	1	4390.27 (69.92)	1
22	**14717.76** (706.70)	15849.91 (490.80)	1	15126.79 (777.32)	1	15773.32 (669.62)	1
23	**15270.46** (431.79)	15449.98 (507.04)	0	15417.58 (496.54)	0	15477.08 (598.95)	0
24	757.65 (188.95)	822.31 (143.55)	1	**646.85** (117.97)	-1	809.21 (169.27)	1
25	**599.68** (28.69)	615.01 (32.18)	1	601.08 (23.23)	0	610.83 (21.67)	1
26	**416.69** (99.53)	478.73 (44.59)	1	461.34 (53.32)	1	457.01 (73.36)	1
27	**2567.80** (167.79)	2633.11 (176.93)	1	2660.13 (195.16)	1	2644.15 (165.09)	1
28	**8535.14** (688.60)	8889.42 (690.83)	1	8698.90 (868.47)	0	8949.75 (736.37)	1

1) We run the following test program whose computational time is T_0:

```
for i=1:1000000
x= 0.55 + (double) i;
x=x + x; x=x./2; x=x*x; x=sqrt(x);
x=log(x); x=exp(x); y=x/x;
end
```

2) We evaluate the computation time T_1 just for function f_{14} with certain dimension and 200000 evaluations.
3) The complete computation time of the algorithm for the function f_{14} with the same dimension and 200000 evaluations is T_2.
4) The step 3 is executed for 5 times independently and we get 5 T_2. $\bar{T}_2 = \text{mean}(T_2)$.

TABLE 30. Mean and standard deviation (in parenthesis) of best-error-runs over 51 independent runs of PCOBAOA, AOA, OBAOA, and COBAOA for 100D problems.

F	PCOBAOA	AOA	h	OBAOA	h	COBAOA	h
1	**160722.84** (10451.97)	175756.44 (5837.94)	1	176864.58 (6366.15)	1	173519.77 (6090.60)	1
2	**6692220149** (1653499390)	7906252158 (2076942548)	1	8114952162 (2014872627)	1	8079365015 (1793083532)	1
3	6.17067E+19 (1.50979E+20)	**2.92266E+20** (6.32519E+20)	1	7.01986E+20 (2.8989E+21)	1	9.36478E+20 (3.73106E+21)	1
4	**199324.15**D (13723.41)	202614.38 (14950.32)	0	203672.36 (15585.03)	0	199128.08 (13892.22)	0
5	19587.81 (1341.34)	**18888.71** (406.82)	-1	19707.92 (533.97)	0	19615.89 (499.78)	0
6	**35516.97** (3047.27)	39613.96 (3723.49)	1	40856.22 (3615.76)	1	39812.07 (3604.05)	1
7	4218801.12 (5181174.60)	17980001.18 (35812418.02)	1	**4152345.21** (3949508.27)	0	10319657.71 (17271498.59)	1
8	21.32 (0.02)	**21.31** (0.03)	0	21.32 (0.02)	0	21.31 (0.03)	0
9	**154.42** (4.12)	155.79 (4.37)	0	157.61 (3.72)	0	156.76 (4.33)	0
10	**30096.19** (3781.47)	33659.35 (4411.09)	1	31203.13 (3812.72)	0	33337.39 (4411.14)	1
11	**2724.68** (237.78)	2912.80 (216.55)	1	2978.12 (261.89)	1	3037.87 (240.77)	1
12	**2733.41** (153.51)	2832.40 (160.42)	1	2845.00 (182.70)	1	2860.58 (168.71)	1
13	**2685.36** (172.24)	2898.87 (169.27)	1	2856.63 (172.10)	1	2884.64 (231.49)	1
14	**27184.66** (1085.73)	29020.73 (999.93)	1	28971.05 (1018.63)	1	28940.15 (1011.90)	1
15	29935.69 (620.73)	**29125.41** (895.42)	1	29677.89 (788.42)	1	29511.88 (991.37)	1
16	4.04 (0.23)	**3.95** (0.27)	-1	4.14 (0.21)	0	4.02 (0.33)	-1
17	**2984.57** (101.95)	3297.74 (75.41)	1	3292.54 (63.69)	1	3298.88 (69.21)	1
18	**3221.24** (87.81)	3235.62 (83.21)	0	3256.24 (81.16)	0	3246.63 (90.73)	0
19	1407081.49 (283656.66)	**1112259.49** (158364.54)	-1	1328395.40 (182400.66)	0	1291720.63 (182603.72)	0
20	50 (2.57094E-08)	50 (6.96984E-14)	0	50 (3.26579E-07)	0	50 (4.29764E-13)	0
21	**8413.24** (193.25)	8502.87 (110.97)	1	8577.62 (116.60)	1	8573.06 (107.48)	1
22	**31595.97** (971.89)	32704.75 (862.77)	1	32543.56 (811.69)	1	32535.02 (935.72)	1
23	33089.93 (580.07)	**33009.85** (758.01)	0	33410.46 (494.14)	1	33411.07 (682.67)	1
24	**2592.57** (397.70)	2748.32 (429.39)	1	2354.73 (422.06)	-1	2658.25 (483.02)	0
25	1271.56 (60.04)	1288.82 (63.76)	0	**1235.86** (49.46)	-1	1274.30 (69.91)	0
26	840.17 (314.03)	1157.53 (671.61)	1	**779.39** (76.70)	-1	960.23 (568.17)	0
27	**6373.72** (398.78)	6397.53 (554.00)	0	6471.36 (531.27)	0	6512.16 (427.51)	1
28	**20852.14** (1250.73)	22416.81 (1801.60)	1	21475.09 (1741.53)	1	21741.19 (1806.88)	1

The complexity of the algorithm is reflected by T_0, T_1, $\bar{T}_2$, $(\bar{T}_2 - T_1)/T_0$.

As it can be seen, although we have included the PCOBL in the original metaheuristics, the computational time cost has not increased significantly. The computational cost of PCOBSCA is lower than SCA and COBSCA but higher than OBSCA. Regarding WOA, the computational cost of PCOBWOA is lower than the original WOA but higher than OBWA and COBWOA. On the other hand, the computational cost of PCOBAOA is lower than COBAOA, whereas it is higher than AOA and OBAOA. It is worth noting that PCOBL reduces the computational cost when combined with SCA and WOA, whereas it increases the same when combined with AOA. This pattern is found in each dimension.

The computational time of SCA is greater than that of PCOBSCA because basic operations (i.e., sine and cosine operators) of classical SCA in PCOBSCA are less executed than in SCA due to generation jumping in PCOBSCA. A similar observation can be made in the case of WOA. However, while comparing AOA with PCOBSCA, it can be observed that there is the execution of only one math operator from four math operators such as addition (+), subtraction (−), multiplication (×), and division (/) in each generation of AOA. In contrast, execution time is less than that of generated solutions using PCOBL in the generation jumping phase of PCOBAOA.

E. REAL WORLD OPTIMIZATION PROBLEMS

The proposal is also validated on engineering design problems that play an essential role in assessing the proposal's potential. The proposal is applied to solve four engineering design problems (presented in Table 21) collected from IEEE CEC2011 on real-world optimization problems CEC2011. All the algorithms are executed with the same initial population for the same problem to guarantee a fair comparison. Table 22 presents the best, worst, median, mean, and standard deviation of objective values obtained from PCOBSCA, SCA, OBSCA, and COBSCA, and the Wilcoxon Signed Rank Test statistic using the mean objective values. It is observed that PCOBSCA statistically outperforms SCA and COBSCA for all four engineering problems. PCOBSCA statistically performs better than OBSCA for P10, P12, and P13 problems, whereas OBSCA statistically outperforms PCOBSCA for the P07 problem.

Similar to Table 22, Table 23 presents the results of PCOBWOA, WOA, OBWOA, and COBWOA, and statistics. It is observed that PCOBWOA statistically outperforms WOA and OBWOA for all problems. Though PCOBWOA statistically outperforms COBWOA for problems P07, P12, and P13, there is no statistically significant difference in PCOBWOA and COBWOA for problem P10.

Table 24 also presents the results of PCOBAOA, AOA, OBAOA, and COBAOA, and statistics. It is observed that PCOBAOA statistically outperforms AOA, OBAOA, and COBAOA for problems P07, P12, and P13.

From Tables 22 & 24, it is observed that PCOBSCA and PCOBAOA are more robust than their counterparts for three out of four problems as they have a lower standard deviation. PCOBSCA and PCOBAOA have a competitive standard deviation for the remaining one problem compared to their counterpart algorithms. It has been observed from Table 23 that PCOBWOA has a lower standard deviation than its counterparts only for the P12 problem, whereas it has competitive standard deviation values for the remaining problems.

From the analysis of the results with the statistical test, it is observed that the combination of PCOBL with SCA, WOA, and AOA can perform better than their counterparts while solving the engineering design problems. This is evidence that the proposed PCOBL scheme has better efficiency and efficacy than OBL and COBL schemes. The source codes of the proposed PCOBL algorithm are available at https://github.com/tapassiaiml/PCOBL

VII. CONCLUSION

Metaheuristics are commonly used to solve global optimization problems. Although there are metaheuristics with different inspirations and operations, balancing exploration and exploitation during the search is still a challenge. This paper proposes a novel Opposition-based learning scheme, called PCOBL, inspired in the partial centroid. PCOBL strategy aims to improve metaheuristics' performance by maintaining an effective balance between exploration and exploitation during the search process. The PCOBL scheme was incorporated in three different metaheuristics (AOA, SCA, and WOA), and a comparative study was conducted on 28 CEC2013 benchmark problems with 30, 50, and 100 dimensions. Besides, metaheuristics using the PCOBL scheme were applied to solve four engineering design problems from IEEE CEC2011 on real-world optimization problems. The proposal was assessed in terms of best-error runs, convergence, exploration and exploitation, and computational cost time.

According to the best-error-runs analysis, the proposed PCOBSCA, PCOBWOA, and PCOBAOA statistically outperformed their competing algorithms for most of the CEC2013 benchmark functions, with different dimensions, i.e., 30D, 50D, and 100D. We also performed a convergence analysis, and the PCOBSCA, PCOBWOA, and PCOBAOA showed to be able to find promising and competitive solutions through an incremental convergence, demonstrating a good balance between exploration and exploitation. Finally, we analyzed the capacity of PCOBSCA, PCOBWOA, and PCOBAOA to balance exploration and exploitation during the search. The results showed that the PCOBSCA is statistically superior to OBSCA, and COBSCA, whereas it balances exploration and exploitation similarly to SCA. PCOBWOA is statistically superior to WOA and COBWOA in maintaining the balance of exploration and exploitation, whereas there is no statistically significant difference between PCOBWOA and OBWOA. PCOBAOA has the statistical superiority to both AOA and OBAOA, whereas there is no statistically significant difference between PCOBAOA and COBWOA in balancing exploration and exploitation. PCOBSCA, PCOBWOA, and PCOBAOA have an effective balance between exploration and exploitation results in achieving better solutions than their competing algorithms. The computational time cost of the proposed PCOBSCA, PCOBWOA, and PCOBAOA is competitive to their competitive algorithms. Apart from the solving of IEEE CEC2013 benchmark problems, PCOBSCA, PCOBWOA, and PCOBAOA have also shown their more effectiveness in solving four

IEEE CEC2011 real-world optimization problems than the competitive algorithms.

This paper shows that the PCOBL has a higher chance of providing better solutions than the OBL and COBL scheme. We intend to conduct a theoretical study of the PCOBL scheme in future work. We also plan to verify the applicability of metaheuristics set with PCOBL for solving real-world problems such as Artificial Neural Network training for regression analysis and medical data classification. Besides, we intend to adjust the proposed algorithm to solve multi-objective optimization problems. Moreover, PCOBL is also useful to other problems associated with the specific areas of biomedical sciences and social networking problems for long-term commercial and healthcare benefits.

REFERENCES

[1] S. Mirjalili, "SCA: A sine cosine algorithm for solving optimization problems," *Knowl.-Based Syst.*, vol. 96, pp. 120–133, Mar. 2016, doi: 10.1016/j.knosys.2015.12.022.

[2] M. A. Elaziz, D. Oliva, and S. Xiong, "An improved opposition-based sine cosine algorithm for global optimization," *Expert Syst. Appl.*, vol. 90, pp. 484–500, Dec. 2017.

[3] N. Rojas-Morales, M.-C. Riff Rojas, and E. Montero Ureta, "A survey and classification of opposition-based metaheuristics," *Comput. Ind. Eng.*, vol. 110, pp. 424–435, Aug. 2017.

[4] A. Malisia, "Improving the exploration ability of ant-based algorithms," in *Oppositional Concepts in Computational Intelligence* (Studies in Computational Intelligence), vol. 155, H. Tizhoosh and M. Ventresca, Eds. Berlin, Germany: Springer, 2008.

[5] H. R. Tizhoosh, "Opposition-based learning: A new scheme for machine intelligence," in *Proc. Int. Conf. Comput. Intell. Modeling, Control Autom. Int. Conf. Intell. Agents, Web Technol. Internet Commerce (CIMCA-IAWTIC)*, vol. 1, 2005, pp. 695–701.

[6] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution algorithms," in *Proc. IEEE Int. Conf. Evol. Comput.*, Jul. 2006, pp. 2010–2017.

[7] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Quasi-oppositional differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, Sep. 2007, pp. 2229–2236.

[8] A. Esmailzadeh and S. Rahnamayan, "Opposition-based differential evolution with protective generation jumping," in *Proc. IEEE Symp. Differ. Evol. (SDE)*, Apr. 2011, pp. 1–8.

[9] A. Ahandani and H. Alavi-Rad, "Opposition-based learning in the shuffled differential evolution algorithm," *Soft Comput.*, vol. 16, no. 8, pp. 1303–1337, 2012.

[10] L. Han and X. He, "A novel opposition-based particle swarm optimization for noisy problems," in *Proc. 3rd Int. Conf. Natural Comput. (ICNC)*, 2007, pp. 624–629.

[11] H. Wang, H. Li, Y. Liu, C. Li, and S. Zeng, "Opposition-based particle swarm algorithm with Cauchy mutation," in *Proc. IEEE Congr. Evol. Comput.*, Sep. 2007, pp. 4750–4756.

[12] H. Wang, Z. Wu, S. Rahnamayan, Y. Liu, and M. Ventresca, "Enhancing particle swarm optimization using generalized opposition-based learning," *Inf. Sci.*, vol. 181, no. 20, pp. 4699–4714, 2011.

[13] W.-F. Gao, S.-Y. Liu, and L.-I. Huang, "Particle swarm optimization with chaotic opposition-based population initialization and stochastic search technique," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 17, no. 11, pp. 4316–4327, 2012.

[14] B. Mandal and T. Si, "Opposition based particle swarm optimization with exploration and exploitation through gbest," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Aug. 2015, pp. 245–250.

[15] A. K. Qin and F. Forbes, "Dynamic regional harmony search with opposition and local learning," in *Proc. 13th Annu. Conf. Companion Genet. Evol. Comput.*, Jul. 2011, pp. 53–54.

[16] Q. Niu, H. Zhang, X. Wang, K. Li, and G. W. Irwin, "A hybrid harmony search with arithmetic crossover operation for economic dispatch," *Int. J. Elect. Power Energy Syst.*, vol. 62, pp. 237–257, Nov. 2014.

[17] G. Shankar and V. Mukherjee, "Quasi oppositional harmony search algorithm based controller tuning for load frequency control of multi-source multi-area power system," *Int. J. Electr. Power Energy Syst.*, vol. 75, pp. 289–302, Feb. 2016.

[18] M. Ventresca and H. R. Tizhoosh, "Simulated annealing with opposite neighbors," in *Proc. IEEE Symp. Found. Comput. Intell.*, Apr. 2007, pp. 186–192.

[19] M. Tubishat, N. Idris, L. Shuib, M. A. M. Abushariah, and S. Mirjalili, "Improved salp swarm algorithm based on opposition based learning and novel local search algorithm for feature selection," *Expert Syst. Appl.*, vol. 145, May 2020, Art. no. 113122.

[20] X. Zhao, F. Yang, Y. Han, and Y. Cui, "An opposition-based chaotic salp swarm algorithm for global optimization," *IEEE Access*, vol. 8, pp. 36485–36501, 2020.

[21] T. Si, P. B. C. Miranda, and D. Bhattacharya, "Novel enhanced salp swarm algorithms using opposition-based learning schemes for global optimization problems," *Expert Syst. Appl.*, vol. 207, Nov. 2022, Art. no. 117961, doi: 10.1016/j.eswa.2022.117961.

[22] T. Si and D. Bhattacharya, "Sine cosine algorithm with centroid opposition-based computation," in *Applications of Artificial Intelligence in Engineering* (Algorithms for Intelligent Systems), X.-Z. Gao, R. Kumar, S. Srivastava, and B. P. Soni, Eds. Singapore: Springer, 2020.

[23] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, May 2016, doi: 10.1016/j.advengsoft.2016.01.008.

[24] L. Abualigah, A. Diabat, S. Mirjalili, M. A. Elaziz, and A. H. Gandomi, "The arithmetic optimization algorithm," *Comput. Methods Appl. Mech. Eng.*, vol. 376, Apr. 2021, Art. no. 113609, doi: 10.1016/j.cma.2020.113609.

[25] S. Das and P. Suganthan, "Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems," Jadavpur Univ., Nanyang Technol. Univ., Singapore, Tech. Rep., 2011.

[26] S. Mahdavi, S. Rahnamayan, and K. Deb, "Opposition based learning: A literature review," *Swarm Evol. Comput.*, vol. 39, pp. 1–23, Apr. 2018.

[27] S. Rahnamayan, J. Jesuthasan, F. Bourennani, H. Salehinejad, and G. F. Naterer, "Computing opposition by involving entire population," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2014, pp. 1800–1807.

[28] Z. Hu, Y. Bao, and T. Xiong, "Partial opposition-based adaptive differential evolution algorithms: Evaluation on the CEC 2014 benchmark set for real-parameter optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2014, pp. 2259–2265.

[29] T. Si and R. Dutta, "Partial opposition-based particle swarm optimizer in artificial neural network training for medical data classification," *Int. J. Inf. Technol. Decis. Making*, vol. 18, no. 5, pp. 1717–1750, Sep. 2019.

[30] H. Huang, X. Feng, A. A. Heidari, Y. Xu, M. Wang, G. Liang, H. Chen, and X. Cai, "Rationalized sine cosine optimization with efficient searching patterns," *IEEE Access*, vol. 8, pp. 61471–61490, 2020, doi: 10.1109/ACCESS.2020.2983451.

[31] M. Meshkat and M. Parhizgar, "A novel weighted update position mechanism to improve the performance of sine cosine algorithm," in *Proc. 5th Iranian Joint Congr. Fuzzy Intell. Syst. (CFIS)*, Mar. 2017, pp. 166–171.

[32] C. Qu, Z. Zeng, J. Dai, Z. Yi, and W. He, "A modified sine-cosine algorithm based on neighborhood search and greedy Levy mutation," *Comput. Intell. Neurosci.*, vol. 2018, Jul. 2018, Art. no. 4231647, doi: 10.1155/2018/4231647.

[33] S. Gupta, K. Deep, S. Mirjalili, and J. H. Kim, "A modified sine cosine algorithm with novel transition parameter and mutation operator for global optimization," *Expert Syst. Appl.*, vol. 154, Sep. 2020, Art. no. 113395, doi: 10.1016/j.eswa.2020.113395.

[34] S. Gupta, K. Deep, and A. P. Engelbrecht, "A memory guided sine cosine algorithm for global optimization," *Eng. Appl. Artif. Intell.*, vol. 93, Aug. 2020, Art. no. 103718.

[35] S. Gupta and K. Deep, "A novel hybrid sine cosine algorithm for global optimization and its application to train multilayer perceptrons," *Appl. Intell.*, vol. 50, no. 4, pp. 993–1026, Apr. 2020.

[36] M. Z. Rehman, A. Khan, R. Ghazali, M. Aamir, and N. M. Nawi, "A new multi sine-cosine algorithm for unconstrained optimization problems," *PLoS ONE*, vol. 16, no. 8, Aug. 2021, Art. no. e0255269, doi: 10.1371/journal.pone.0255269.

[37] W. Long, T. Wu, J. Jiao, M. Tang, and M. Xu, "Refraction-learning-based whale optimization algorithm for high-dimensional problems and parameter estimation of PV model," *Eng. Appl. Artif. Intell.*, vol. 89, Mar. 2020, Art. no. 103457, doi: 10.1016/j.engappai.2019.103457.
[38] M. Li, G. Xu, Q. Lai, and J. Chen, "A chaotic strategy-based quadratic opposition-based learning adaptive variable-speed whale optimization algorithm," *Math. Comput. Simul.*, vol. 193, pp. 71–99, Mar. 2022, doi: 10.1016/j.matcom.2021.10.003.
[39] H. S. Alamri, Y. A. Alsariera, and K. Z. Zamli, "Opposition-based whale optimization algorithm," *Adv. Sci. Lett.*, vol. 24, no. 10, pp. 51–67, 2016, doi: 10.1166/asl.2018.12959.
[40] S. Chauhan and G. Vashishtha, "Mutation-based arithmetic optimization algorithm for global optimization," in *Proc. Int. Conf. Intell. Technol. (CONIT)*, Jun. 2021, pp. 1–6.
[41] J. O. Agushaka and A. E. Ezugwu, "Advanced arithmetic optimization algorithm for solving mechanical engineering design problems," *PLoS ONE*, vol. 16, no. 8, Aug. 2021, Art. no. e0255703, doi: 10.1371/journal.pone.0255703.
[42] D. Izci, S. Ekinci, E. Eker, and A. Dundar, "Improving arithmetic optimization algorithm through modified opposition-based learning mechanism," in *Proc. 5th Int. Symp. Multidisciplinary Stud. Innov. Technol. (ISMSIT)*, vol. 1, Oct. 2021, pp. 1–5.
[43] J. Liang, B. Qu, P. Suganthan, and A. G. Hernndez-Díaz, "Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization," Zhengzhou Univ., Nanyang Technol. Univ., Tech. Rep. TR201212, 2013.
[44] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, Mar. 2011.
[45] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 526–553, Jun. 2009, doi: 10.1109/TEVC.2008.2009457.
[46] J. Xu and J. Zhang, "Exploration-exploitation tradeoffs in metaheuristics: Survey and analysis," in *Proc. 33rd Chin. Control Conf. (CCC)*, Jul. 2014, pp. 8633–8638.
[47] B. Morales-Castañeda, D. Zaldívar, E. Cuevas, F. Fausto, and A. Rodríguez, "A better balance in metaheuristic algorithms: Does it exist?" *Swarm Evol. Comput.*, vol. 54, May 2020, Art. no. 100671, doi: 10.1016/j.swevo.2020.100671.

TAPAS SI (Member, IEEE) received the B.Tech. degree in computer science and engineering from Maulana Abul Kalam Azad University (formerly West Bengal University of Technology), and the M.Tech. degree in information technology and the Ph.D. degree in engineering from the National Institute of Technology Durgapur, West Bengal, India. He is currently an Associate Professor with the Department of Computer Science and Engineering, University of Engineering and Management, Jaipur, Rajasthan, India. He has published 72 papers in reputed international journals/conferences. His research interests include machine learning, swarm intelligence, medical image processing, and medical data mining. He is a member of the Institute of Engineers, India; the Soft Computing Research Society; and the Machine Intelligence Research Laboratories, USA. He serves as a reviewer for journals of Springer, Elsevier, and IEEE. He is listed in *Who's Who in the World*, in 2016 and 2020, and Marquis's *Who's Who in the USA*.

DEBOLINA BHATTACHARYA received the B.Tech. degree in computer science and engineering from Maulana Abul Kalam Azad University (formerly West Bengal University of Technology), WB, India. She is currently with the Department of Computer Science and Engineering, Sarala Birla University, Ranchi, JH, India. She has published three papers at reputed international conferences. Her research interests include swarm intelligence and machine learning.

SOMEN NAYAK (Member, IEEE) received the B.C.A. degree from the University of Burdwan, the M.C.A. degree from Punjab Technical University (formerly I. K. Gujral Punjab Technical University), Jalandhar, and the M.Tech. degree in computer science and engineering from Maulana Abul Kalam Azad University (formerly West Bengal University of Technology). He is currently an Assistant Professor and the Head of the Department of Computer Application, University of Engineering and Management, Jaipur, Rajasthan, India. His research interests include machine learning, swarm intelligence, medical image processing, and the Internet of Things (IoT). He is an active member of the IEEE Computer Intelligence Society. He is acting as a Convener of two international conferences, such as the Global Conference on Artificial Intelligence and Applications (GCAIA) and the International Conference on Human-Centric Smart Computing (ICHCSC). He serves as a reviewer for many IEEE and Springer conferences. He is the Managing Editor of *American Journal of Advance Computing* (AJAC).

PÉRICLES B. C. MIRANDA received the bachelor's degree in computer engineering from the University of Pernambuco, in 2010, and the master's and Ph.D. degrees in computer science from the Federal University of Pernambuco, in 2012 and 2016, respectively. Currently, he is an Adjunct Professor with Universidade Federal Rural de Pernambuco (UFRPE). He has extensive experience in computer science, with an emphasis on artificial intelligence, working mainly on topics related to optimization, machine learning, and meta-heuristic and hyper-heuristic algorithms.

UTPAL NANDI received the M.Sc. degree in computer science from Vidyasagar University, Midnapore, Paschim Medinipur, West Bengal, India, in 2006, and the M.Tech. degree in computer science and engineering and the Ph.D. degree from the University of Kalyani, Nadia, West Bengal, in 2009 and 2018, respectively. He is currently an Assistant Professor with the Department of Computer Science, Vidyasagar University. He has 12 years of teaching and research experience. He has published more than 38 articles in international journals, book chapters, and conferences. His research interests include data and image compression techniques, image processing, multimedia technology, computer vision, and artificial intelligence.

SAURAV MALLIK (Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Engineering, Jadavpur University, Kolkata, India, in 2017. His Ph.D. works were conducted with the Machine Intelligence Unit, Indian Statistical Institute, Kolkata, India. He is currently a Postdoctoral Fellow of environmental epigenetics with the Harvard T. H. Chan School of Public Health, Boston, USA. Previously, he was a Postdoctoral Fellow with the Department of School of Biomedical Informatics, University of Texas Health Science Center at Houston, Houston, TX, USA; and the Division of Bio-Statistics, Department of Public Health Sciences, University of Miami Miller School of Medicine, Miami, FL, USA. He has coauthored more than 75 research articles with a Google H-index of 17. His research interests include computational biology, bioinformatics, data mining, bio-statistics, and pattern recognition. He was a Research Associate of the Council of Scientific and Industrial Research, MHRD, Government of India, in 2017. He was also a recipient of the Emerging Researcher in Bioinformatics Award from the Bioclues BIRD Award Steering Committee, India, in 2020. He is an editor of many journals.

UJJWAL MAULIK (Fellow, IEEE) has been a Professor with the Department of Computer Science and Engineering, Jadavpur University, since 2004. He was the former Head of the Department of Computer Science and Engineering, Jadavpur University. He also held the position of the Principal in Charge and the Head of the Department of Computer Science and Engineering, Kalyani Government Engineering College. He has worked with many universities and research laboratories around the world as a Visiting Professor/a Scientist, including the Los Alamos National Laboratory, USA, in 1997; the University of New South Wales, Australia, in 1999; the University of Texas at Arlington, USA, in 2001; the University of Maryland at Baltimore County, USA, in 2004; the Fraunhofer Institute for Autonome Intelligent Systems, St. Augustin, Germany, in 2005; Tsinghua University, China, in 2007; Sapienza University, Rome, Italy, in 2008; the University of Heidelberg, Germany, in 2009; the German Cancer Research Center (DKFZ), Germany, in 2010, 2011, and 2012; Grenoble INP, France, in 2010, 2013, and 2016; the University of Warsaw, in 2013 and 2019; the University of Padova, Italy, in 2014 and 2016; Corvinus University, Budapest, Hungary, in 2015 and 2016; the University of Ljubljana, Slovenia, in 2015 and 2017; and the International Center for Theoretical Physics (ICTP), Trieste, Italy, in 2014, 2017, and 2018. His research interests include machine learning, pattern analysis, data science, bioinformatics, multi-objective optimization, social networking, the IoT, and autonomous car. In these areas, he has published ten books and more than 400 papers, mentored several start-ups, filed several patents, and has guided 22 doctoral students. He is a Fellow of the Indian National Academy of Engineers (INAE), India, the National Academy of Science India (NASI), and the International Association for Pattern Recognition (IAPR), USA. He is a Distinguished Member of ACM. He is a Distinguished Speaker of IEEE and ACM. He was a recipient of the Alexander von Humboldt Fellowship in 2010, 2011, and 2012, and was a Senior Associate of ICTP, Italy, from 2012 to 2018.

HONG QIN received the Ph.D. degree from the Department of Biochemistry and Molecular Biology, University of Chicago, Chicago, IL, USA, in 2001. He is a Professor with the Department of Computer Science and Engineering, The University of Tennessee at Chattanooga, USA. He has published more than 60 international peer-reviewed journals and conferences. His research interests include cellular aging, COVID-19, mathematical modeling, machine learning, data science, bioinformatics, deep learning, and genomics.