

EMPIRICAL ANALYSIS AND IMPROVEMENT OF THE PSO-SONO OPTIMIZATION ALGORITHM

MAHAMED G. H. OMRAN^{1,*}, HUI WANG² AND MOHAMMAD ALASKANDARANI³

Abstract. PSO-sono is a recent and promising variant of the Particle Swarm Optimization (PSO) algorithm. It outperforms other popular PSO variants on many benchmark test sets. In this paper, we investigate the performance of PSO-sono on more problems (including 21 real-world optimization problems). Moreover, we propose a new, more powerful yet simpler and more efficient variant of PSO-sono, called IPSO-sono. The proposed approach uses ring topology, non-linear ratio reduction and opposition-based learning to improve the performance of PSO-sono. The proposed approach is compared with other state-of-the-art metaheuristic algorithms on 12 IEEE CEC 2022 and 21 real-world problem defined in the IEEE CEC 2011. The results show that IPSO-sono outperforms PSO-sono on most problems and performs well compared to other state-of-the-art approaches.

Mathematics Subject Classification. 68W20, 90C59, 90C26.

Received March 4, 2024. Accepted March 16, 2025.

1. INTRODUCTION

Consider the following optimization problem:

$$\min_{\mathbf{x} \in \mathbb{R}^D, l_j \leq x_j \leq u_j; \forall j} f(\mathbf{x}) \quad (1)$$

where $f(\mathbf{x})$ is the objective function to be minimized, \mathbf{x} is a vector of decision variables in \mathbb{R}^D , and l_j and u_j are the lower and upper bounds, respectively, for the j -th component of \mathbf{x} . The goal is to find the value of \mathbf{x} that minimizes the objective function $f(\mathbf{x})$ while satisfying the boundary constraints.

Optimization problems arise in various fields of science, engineering, economics, and many other domains, where the goal is to find the best solution among a set of feasible solutions. Optimization problems can be categorized as linear or nonlinear, continuous or discrete, and deterministic or stochastic. In general, finding the global optimal solution for complex optimization problems is a challenging task, as the number of possible solutions increases exponentially with the problem size.

Metaheuristics [5, 11, 39] are a class of optimization algorithms that can handle complex optimization problems with multiple local optima, discontinuous objective functions, and constraints. Metaheuristics are iterative search

Keywords. Metaheuristics, optimization, Particle Swarm Optimization, PSO-sono, sustainability.

¹ College of Computing and Systems, Abdullah Al Salem University, Khaldiya, Kuwait.

² School of Information Engineering, Nanchang Institute of Technology, Nanchang 330099, P.R. China.

³ Computer Science Department, Gulf University for Science & Technology, West Mishref, Kuwait.

*Corresponding author: omran.m@ieee.org

algorithms that use heuristics or approximate methods to guide the search towards the optimal solution. Unlike exact methods, such as linear programming or dynamic programming, metaheuristics do not guarantee finding the global optimal solution, but rather aim to find a good solution within a reasonable time frame.

Metaheuristics can be divided into two categories: population-based and single-solution-based methods. Population-based methods, such as Genetic Algorithms (GA) [9], Particle Swarm Optimization (PSO) [12], Differential Evolution (DE) [35] and Ant Colony Optimization (ACO) [4, 34], maintain a set of candidate solutions called the population, and iteratively update the population using various operators such as selection, crossover, mutation, and local search. Single-solution-based methods, such as Simulated Annealing [36], Tabu Search [8] and Variable Neighborhood Search [10], start with a single initial solution and iteratively update it by moving to a neighboring solution that improves the objective function value.

Metaheuristics have been applied to various optimization problems in diverse fields, such as engineering design, transportation, finance, scheduling and bioinformatics [24]. They have also inspired the development of hybrid and adaptive methods that combine the strengths of different algorithms and adapt their parameters dynamically based on the problem characteristics.

A recent and promising metaheuristic is PSO-sono [20], which is a variant of the PSO algorithm. According to the results reported in [20], PSO-sono generally outperforms other popular PSO-variants on many benchmark test sets (namely, IEEE CEC 2013, 2014 and 2017) consisting of 88 real-parameter single-objective optimization functions. In this paper, we investigate the performance of PSO-sono on more problems, *i.e.* IEEE CEC 2022 and the 21 IEEE CEC 2011 real-world optimization problems. We believe that testing any optimizer on real-world problems is essential to validate its performance. Moreover, we propose some changes to the original algorithm to improve its performance. Another objective is to replace complex operations in PSO-sono with simpler operations without deteriorating the performance of the optimizer.

Hence, the contributions of this work are:

- Investigate the performance of PSO-sono on more problems, especially on real-world problem.
- Propose major and minor changes to the original algorithm to improve its performance.
- Make the PSO-sono easier to implement and simpler to understand by replacing relatively complex operations with simpler ones without degrading the performance of the algorithm.

The rest of the paper is organized as follows. Section 2 describes the PSO algorithm and its variants. The PSO-sono algorithm is explained in Section 3. In Section 4, the proposed approach is presented. The results of the proposed algorithm, in comparison with other approaches from the literature, on the IEEE CEC 2022 benchmark functions and IEEE CEC 2011 real-world problems, are reported and discussed in Section 5. Finally, Section 6 concludes the paper and highlight possible future work.

2. PARTICLE SWARM OPTIMIZATION

Particle swarm optimization (PSO) is a population-based stochastic optimization algorithm, inspired by the collective behavior of birds flocking or fish schooling. The algorithm simulates the social behavior of individuals in a group, who collectively search for the best solution to a problem. In PSO, each potential solution is represented by a particle, which moves through the search space based on its position and velocity. The movement of each particle is influenced by its own best-known position and the global best-known position of the population (a.k.a *swarm* in the PSO literature).

The PSO algorithm begins with an initial swarm of particles randomly distributed in the search space. The position and velocity of each particle are updated iteratively based on the following equations:

$$v_{i,j}(t+1) = \omega v_{i,j}(t) + c_1 r_1 (p_{i,j} - x_{i,j}(t)) + c_2 r_2 (g_j - x_{i,j}(t)) \quad (2)$$

$$x_{i,j}(t+1) = x_{i,j}(t) + v_{i,j}(t+1) \quad (3)$$

where $v_{i,j}(t)$ and $x_{i,j}(t)$ are the velocity and position of the i th particle in the j th dimension at time t , respectively. ω is the inertia weight, which controls the impact of the previous velocity on the new velocity. c_1 and c_2

are the acceleration coefficients, and r_1 and r_2 are random numbers uniformly distributed in the range $[0, 1]$. $p_{i,j}$ is the best-known personal position of the i th particle in the j th dimension, and g_j is the best-known position of the swarm in the j th dimension.

The values of $p_{i,j}$ and g_j are updated at each iteration, as follows:

$$p_{i,j}(t+1) = \begin{cases} x_{i,j}(t+1) & \text{if } f(x_{i,j}(t+1)) < f(p_{i,j}(t)) \\ p_{i,j}(t) & \text{otherwise} \end{cases} \quad (4)$$

$$g_j(t+1) = \begin{cases} p_{i,j}(t+1) & \text{if } f(p_{i,j}(t+1)) < f(g_j(t)) \\ g_j(t) & \text{otherwise} \end{cases} \quad (5)$$

where $f(\mathbf{x})$ is the objective function to be optimized.

The algorithm terminates when a stopping criterion is met, such as reaching a maximum number of iterations or achieving a satisfactory solution. The final solution is the position of the particle with the best-known fitness value.

PSO has several advantages over other optimization algorithms, such as its simplicity, fast convergence and ability to handle multimodal and non-linear functions. However, it also has some limitations, such as premature convergence and sensitivity to parameter settings [6].

Several PSO variants have been proposed recently. One popular variant is the Comprehensive Learning PSO (CLPSO) proposed by Liang *et al.* [16]. In CLPSO, an exemplar particle is chosen from the swarm and each dimension of each particle either learns from the corresponding dimension of the exemplar or the particle's own personal best solution. Furthermore, in CLPSO, ω follows a linear reduction from 0.9 to 0.4. An extension of CLPSO was proposed by Nasir *et al.* [21], which uses a ring topology that changed during the iteration by some intervals. Furthermore, a regrouping operation of the particles is conducted every few iterations. Rather than using the global and personal best experience of the swarm, Social Learning PSO (SLPSO) [1] uses a novel social learning approach where each particle learns from any particle with a better objective function value. Another PSO variant, Ensemble PSO (EPSO), proposed by Lynn and Suganthan [18] uses five basic update equations from different PSO variants to tackle optimization problems. The EPSO uses a self-adaptive selection process to choose the most appropriate update equation used by the particles in the larger sub-population in each iteration. The MPSO algorithm [17] uses a chaos-based non-linear inertia weight to "balance" exploration and exploitation. Furthermore, MPSO employs an adaptive strategy-based update equation to enhance the performance of PSO when applied to complex optimization problems. A recent PSO variant is PSO-sono [20], which is described in Section 3.

A recent survey of PSO and its variants can be found in [32]. Moreover, Peng *et al.* [25] investigated the impact of swarm topology on the performance of PSO and its variants.

3. THE PSO-SONO ALGORITHM

In this section, the details of the PSO-sono algorithm are presented.

3.1. The update paradigm

The swarm is divided into two sub-swarms, *i.e.* the better-particle group and the worst-particle group, by sorting the particles by their fitness values. The size of each group is dynamically changed during each iteration, t . The ratio, r , is used to denote the percentage of the better-particle group, which is calculated according to

$$r = \frac{ns_b}{ns_b + ns_w}, \quad (6)$$

where ns_b and ns_w denote the number of success particles in the better-particle group and worse-particle group, respectively. The ratio, r , is truncated to be within $[0.1, 0.9]$.

The better-particle group uses the following update equation,

$$\mathbf{v}_i(t+1) = \omega(t) \cdot \mathbf{v}_i(t) + c_1(t) \cdot r_1 \cdot (\mathbf{p}_i(t) - \mathbf{x}_i(t)) + c_2(t) \cdot r_2 \cdot (\mathbf{g}(t) - \mathbf{x}_i(t)), \quad (7)$$

and

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1). \quad (8)$$

where $r_1, r_2 \sim U(0, 1)$ and $\omega_t, c_{1,t}$ and $c_{2,t}$ are updated as follows,

$$\omega(t) = \omega_{\max} - \frac{t}{T} \cdot (\omega_{\max} - \omega_{\min}), \quad (9)$$

$$c_{1,t} = 2.5 - 2 \cdot \frac{t}{T}, \quad (10)$$

and

$$c_{2,t} = 0.5 + 2 \cdot \frac{t}{T} \quad (11)$$

where T is the maximum number of iterations. The other worse-particle group uses another update paradigm:

$$v_{i,j}(t+1) = r_0 \cdot v_{i,j}(t) + r_1 \cdot I_{i,j}(t) + r_1 \cdot \epsilon \cdot C_{i,j}(t), \quad (12)$$

and

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \quad (13)$$

where

$$\mathbf{I}_i(t) = \mathbf{x}_k(t) - \mathbf{x}_i(t), \quad (14)$$

and

$$\mathbf{C}_i(t) = \bar{\mathbf{x}}_{\text{center}}(t) - \mathbf{x}_i(t) \quad (15)$$

where $r_0 \sim U(0, 1)$ is generated anew for each parameter in the D -dimensional velocity, $r_1 \sim U(0, 1)$, \mathbf{x}_k is a randomly chosen particle with better objective function value than particle i , ϵ denotes the social influence, $\epsilon = D/N \cdot 0.01$ and $\bar{\mathbf{x}}_{\text{center}}(t) = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i(t)$, which is the center of the population.

3.2. The Fully-Informed Search (FIS) scheme

To address the premature convergence problem of PSO, a *fully-informed search scheme* is used. In this scheme, the knowledge of the entire swarm is utilized to help the global best particle, *i.e.* \mathbf{g} , to escape from a local minimum. The scheme is applied once in each iteration as described below.

$$\mathbf{g}(t) = \begin{cases} \mathbf{g}(t) & f(\mathbf{g}(t)) < f(\mathbf{x}_{\text{FIS}}(t)) \\ \mathbf{x}_{\text{FIS}}(t) & \text{Otherwise} \end{cases} \quad (16)$$

where $\mathbf{x}_{\text{FIS}}(t)$ is computed as follow:

$$\mathbf{x}_{\text{FIS}}(t) = \begin{cases} \mathbf{g}(t) \cdot \cos(1 - \frac{t}{T}) + \boldsymbol{\sigma} & \text{if } U(0, 1) < 0.5 \\ \mathbf{g}(t) \cdot \sin(1 - \frac{t}{T}) + \boldsymbol{\sigma} & \text{Otherwise} \end{cases} \quad (17)$$

where $\boldsymbol{\sigma}$ is the fully-informed vector calculated as follow:

$$\begin{cases} \bar{\mathbf{x}}_{\text{center}}(t) & = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i(t) \\ \mathbf{var}_i & = (\mathbf{x}_i(t) - \bar{\mathbf{x}}_{\text{center}}(t)) + \boldsymbol{\phi} \otimes (\mathbf{p}_i(t) - \mathbf{x}_i(t)) \\ \boldsymbol{\sigma} & = \frac{1}{N} \sum_{i=1}^N \mathbf{var}_i \end{cases} \quad (18)$$

where \otimes is the element-wise (*i.e.* Hadamard product), $\mathbf{p}_i = (\sum_{k=1}^n (\phi_k \otimes \mathbf{x}_{nb,k}(t)) / n) \oslash \phi$, $\mathbf{x}_{nb,k}(t)$ is a neighbor particle of $\mathbf{x}_i(t)$ in the ring topology, \oslash is the element-wise division and ϕ is the acceleration weight, *i.e.* $\phi = \sum_{k=1}^n \phi_k$ where $\phi_k \sim U([0, 4.1/n])$ and n is the size of the neighborhood, typically set to 2.

The pseudocode of the PSO-sono algorithm is listed in Algorithm 1.

The PSO-sono has been compared with the PSO variants discussed in Section 2 on a large test set containing all the functions from CEC 2013, CEC 2014 and CEC 2017. The results show that PSO-sono generally outperforms the other PSO-variants on most tested problems.

Algorithm 1: Pseudocode for the PSO-sono algorithm.

input : Solution space $[\mathbf{l}, \mathbf{u}]$, Velocity $[V_{\min}, V_{\max}]$ and maximum number of function evaluations nfe_{\max} .

output: Number of function evaluations, nfe , best solution \mathbf{g} and best fitness value $f(\mathbf{g})$.

```

1  $t \leftarrow 1$ ;
2 for  $i \leftarrow 1$  to  $N$  do
3   Initialize particle  $i$  velocity  $\mathbf{v}_i(t) \sim U([V_{\min}, V_{\max}])^D$ ;
4   Initialize the  $i$ -th particle,  $\mathbf{x}_i(t) \sim U([\mathbf{l}, \mathbf{u}])^D$ ;
5   Calculate the fitness value  $f(\mathbf{x}_i(t))$ ;
6  $nfe \leftarrow N$ ;
7 Label  $\mathbf{g}(t)$  and  $f(\mathbf{g})$ ;
8 while  $nfe < nfe_{\max}$  do
9   Sort the swarm in descending order;
10  Calculate the parameters according to equations (9), (10) and (11);
11  Separate it into two groups according to  $r$ ;
12  Update particles in the better-particle group according to equations (7) and (8);
13  Update particles in the worst-particle group according to equations (12) and (13);
14  for  $i \leftarrow 1$  to  $N$  do
15    Calculate the fitness value  $f(\mathbf{x}_i(t))$ ;
16   $nfe \leftarrow nfe + N$ ;
17  Update the ratio  $r$  according to equation (6);
18  Apply fully-informed search according to equation (16);
19   $nfe \leftarrow nfe + 1$ ;
20  Update  $\mathbf{g}(t)$  and best fitness value  $f(\mathbf{g}(t))$ .
```

4. THE PROPOSED APPROACH

In this section, the details of the modifications of PSO-sono are discussed and motivated. First we will discuss minor changes and fixes that will be incorporated into the proposed major changes. To investigate the effect of these changes, the set of 12 IEEE CEC 2022 benchmark functions is used. The details of these functions along with the experimental setup are explained in Section 5.

4.1. Minor changes

- (1) In the publicly available Matlab code of PSO-sono¹, there is a bug. After generating \mathbf{x}_{FIS} in equation (17) boundary constraint violation is not checked. This bug is fixed in this study.
- (2) Instead of being initialized randomly within the range of $[V_{\min}, V_{\max}]$ as in PSO-sono, the initial velocity of each particle is set to zero. In real world, the velocity of physical objects in their initial positions is zero. Particles initialized with non-zero velocities violate this analogy [6]. Moreover, this is not needed given the fact that particles' positions are randomly initialized, ensuring random positions and moving directions [6].

¹ <https://sites.google.com/view/zhenyumeng/>.

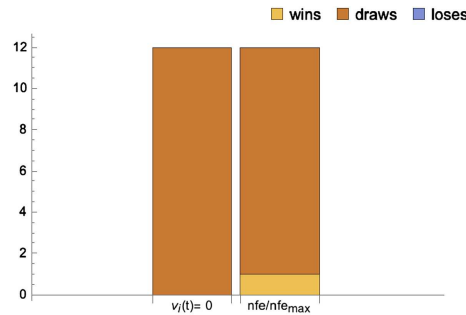


FIGURE 1. The effect of the minor changes on the performance of PSO-sono, “wins/draws/loses” means the proposed change wins, draws and loses, respectively, against the original PSO-sono.

- (3) Replacing t/T with nfe/nfe_{max} . Most metaheuristic algorithms iterates until a maximum number of function evaluations, *i.e.* nfe_{max} , is reached. Finding the maximum number of iterations, T , is not always straightforward since some metaheuristics perform variable number of function evaluations in each iteration. Thus, replacing T with nfe_{max} is more useful.

The effect of the above minor changes on the performance of PSO-sono is depicted in Figure 1. The figure shows that initializing the velocities of particles to zero allows us to remove two parameters, *i.e.* V_{min} and V_{max} without degrading the performance of PSO-sono as shown in the left column of Figure 1. The right column shows that using nfe/nfe_{max} is at least as good as using t/T .

4.2. Major changes

After introducing the minor changes and showing their usefulness, we integrate them into the PSO-sono algorithm. In this section, we will introduce some major changes to the algorithm and show their usefulness.

4.2.1. The ring topology

Equation (7) employs the fully connected topology, as depicted in Figure 2, which enables all particles in the swarm to share information globally. In this global neighborhood, information is swiftly disseminated to all particles, resulting in the entire swarm quickly converging to the same search region, which increases the risk of premature convergence. To address this issue, the *ring* topology, illustrated in Figure 3, utilizes local neighborhoods, similar to the one used in the *lbest* PSO [13], in which each particle only communicates with its immediate neighbors. Consequently, the population requires more iterations to converge, but it may discover better solutions. For instance, in Figure 3, solution \mathbf{x}_1 has two neighbors, \mathbf{x}_2 and \mathbf{x}_6 , while solution \mathbf{x}_4 has \mathbf{x}_3 and \mathbf{x}_5 as its neighbors. As a result, each particle interacts only with its two neighbors (rather than the entire swarm), which slows down the convergence speed of the proposed approach but reduces the likelihood of premature convergence. In summary, while the fully connected topology tends to bias the search toward the global best and adopt an exploitative approach, the ring topology favors exploration over exploitation. This advantage has made the ring topology quite popular in metaheuristic research. Recently, Lynn *et al.* [19] identified the use of population topologies, including the ring topology, as a key area for research.

Hence, equation (7) is replaced by:

$$\mathbf{v}_i(t+1) = \omega(t) \cdot \mathbf{v}_i(t) + c_1(t) \cdot r_1 \cdot (\mathbf{p}_i(t) - \mathbf{x}_i(t)) + c_2(t) \cdot r_2 \cdot (\mathbf{g}_i(t) - \mathbf{x}_i(t)), \quad (19)$$

where \mathbf{g}_i is the best particle in the neighborhood \mathcal{N}_i , which is defined as

$$\mathbf{g}_i(t+1) \in \{\nu_i | f(\mathbf{p}_i(t+1)) = \min\{f(\mathbf{x})\}, \forall \mathbf{x} \in \mathcal{N}_i\}, \quad (20)$$

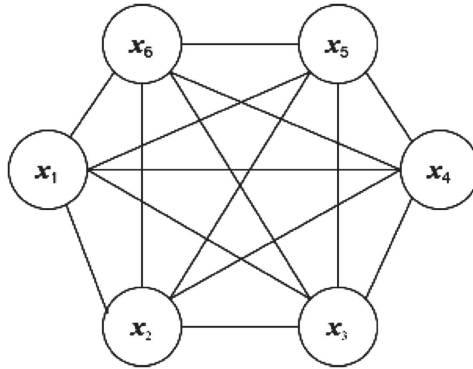


FIGURE 2. A fully-connected topology with 6 particles.

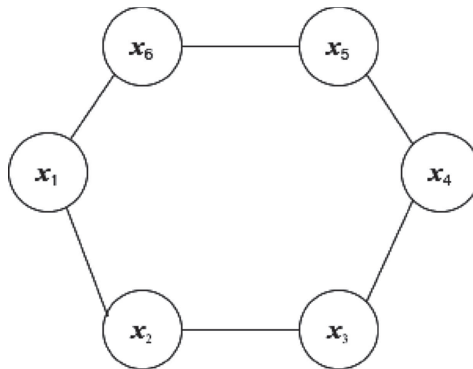


FIGURE 3. A ring topology with 6 particles.

with the neighborhood defined as $\mathcal{N}_i = \{\mathbf{p}_{i-1}(t), \mathbf{p}_i(t), \mathbf{p}_{i+1}(t)\}$. Notice that the ring topology used in equation (18) excludes the particle itself (*i.e.* \mathbf{x}_i) and uses its left and right neighbors (*i.e.* \mathbf{x}_{i-1} and \mathbf{x}_{i+1}). Figure 5 (first column from the left) compares PSO-sono using the fully-connected topology with PSO-sono using the ring topology. The results show that using the ring topology outperforms the original PSO-sono on 7 functions, *i.e.* it performs better on about 60% of the functions. The fully connected topology performs better on only one function.

4.2.2. The ratio r

Rather than using equation (6) to dynamically adjust the ratio r , two simpler alternatives are proposed. The first approach is to linearly increase the ratio r from 0.1 to 0.9 as follows:

$$r = 0.1 + 0.8 \cdot \frac{nfe}{nfe_{\max}}. \quad (21)$$

The second approach uses a non-linear equation to increase r from 0.1 to 0.9:

$$r = 1 - 0.9 \cdot (0.1/0.9)^{\frac{nfe}{nfe_{\max}}}. \quad (22)$$

Thus, the focus will first be on exploration, *i.e.* 90% of the particles use equation (12), then on exploitation, *i.e.* 90% of the swarm use equation (7).

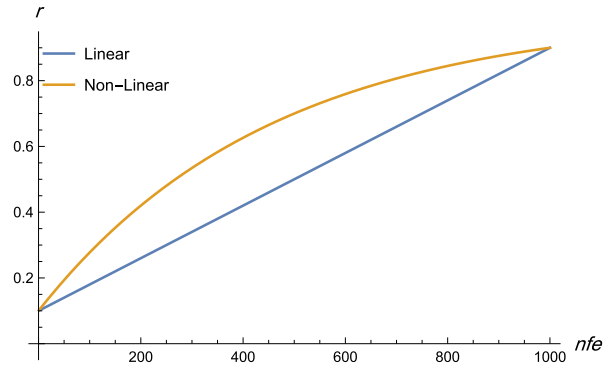


FIGURE 4. Linear *vs.* non-linear increase of r .

Figure 4 illustrates the difference between the two approaches.

Figure 5 compares the performance of PSO-sono using the original approach (Eq. (6)), the linear approach (Eq. (21)) and the non-linear approach (Eq. (22)). The results (second, third and fourth columns from the left) show that the two proposed approaches are generally better than the original one. The linear and non-linear approaches are comparably. However, the non-linear approach will be adopted in this study since it did not degrade the performance of PSO-sono on any problem.

4.2.3. The simplified FIS scheme

The fully-informed search scheme of the PSO-sono algorithm uses the knowledge of the whole population to avoid premature convergence. However, the scheme is relatively difficult to understand and implement. A simpler scheme is the Centriod Opposition-Based Learning (COBL) [28], which achieved remarkable success in Differential Evolution algorithm in every competition [33]. In COBL, an opposite-point (relative to a point \mathbf{x}) is defined as follows,

$$\mathbf{x}_O = 2 \cdot \bar{\mathbf{x}}_{\text{center}} - \mathbf{x}. \quad (23)$$

From the above equation, it is clear that COBL uses the knowledge of the whole swarm as in the more complex FIS scheme but in a simpler way. Using COBL, equations (17) and (18) are replaced with the following equation:

$$\mathbf{x}_{\text{FIS}}(t) = 2 \cdot \bar{\mathbf{x}}_{\text{center}}(t) - \mathbf{g}(t). \quad (24)$$

Hence, we are comparing the best solution found so far with its opposite as defined in equation (23). Figure 5 (the rightmost column) summarizes the results of using the simplified FIS rather than the original FIS in PSO-sono. The results show that the simpler scheme is at least as good as the original FIS scheme.

To summarize, Figure 5 shows that using ring topology is the most important change to the original approach followed by changing the ratio, r , update equation and finally using COBL.

4.2.4. The IPSO-sono algorithm

The final version after incorporating all the aforementioned changes is called the Improved PSO-sono (IPSO-sono) algorithm. A pseudocode of the proposed approach is shown in Algorithm 2.

5. EXPERIMENTAL RESULTS

To investigate the performance of the proposed approach, the 12 benchmark functions of the CEC 2022 competition on single objective bound-constrained numerical optimization [15] have first been used. A set of 21 real-world optimization problems have then been used for comparison purposes.

The 12 CEC 2022 test set consists of 12 functions with different characteristics:

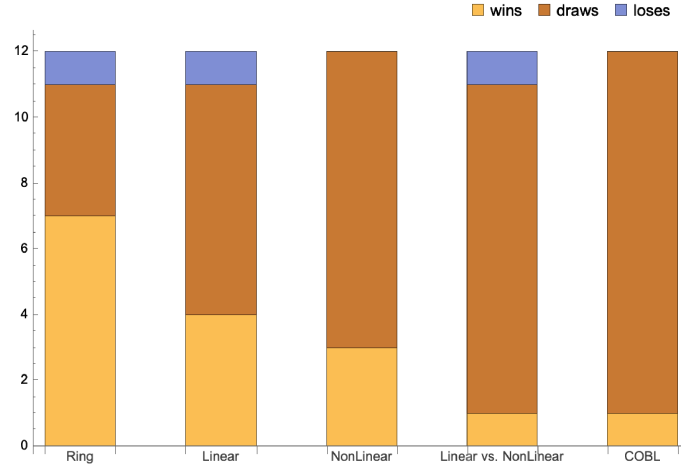


FIGURE 5. The effect of the major changes on the performance of PSO-sono, “wins/draws/loses” means the proposed change wins, draws and loses, respectively, against the original PSO-sono.

Algorithm 2: Pseudocode for the IPSO-sono algorithm.

input : Solution space $[\mathbf{l}, \mathbf{u}]$ and maximum number of function evaluations nfe_{\max} .

output: Number of function evaluations, nfe , best solution \mathbf{g} and best fitness value $f(\mathbf{g})$.

```

1  $t \leftarrow 1$ ;
2 for  $i \leftarrow 1$  to  $N$  do
3   Initialize particle  $i$  velocity  $\mathbf{v}_i(t) \sim \mathbf{0}$ ;
4   Initialize the  $i$ -th particle,  $\mathbf{x}_i(t) \sim U([\mathbf{l}, \mathbf{u}]^D)$ ;
5   Calculate the fitness value  $f(\mathbf{x}_i(t))$ ;
6  $nfe \leftarrow N$ ;
7 Label  $\mathbf{g}(t)$  and  $f(\mathbf{g})$ ;
8 while  $nfe < nfe_{\max}$  do
9   Sort the swarm in descending order;
10  Calculate the parameters according to equations (9), (10) and (11) but using  $nfe/nfe_{\max}$  (rather than  $t/T$ );
11  Separate the swarm into two groups according to  $r$  as defined in equation (22);
12  Update particles in the better-particle group according to equations (19) and (8);
13  Update particles in the worst-particle group according to equations (12) and (13);
14  for  $i \leftarrow 1$  to  $N$  do
15    Calculate the fitness value  $f(\mathbf{x}_i(t))$ ;
16   $nfe \leftarrow nfe + N$ ;
17  Apply the simplified FIS using equation (24);
18   $nfe \leftarrow nfe + 1$ ;
19  Update  $\mathbf{g}(t)$  and best fitness value  $f(\mathbf{g}(t))$ .

```

- One unimodal function.
- Four shifted and rotated functions.
- three hybrid functions that are linear combinations of some functions.
- Four composition functions.

TABLE 1. Summary of the CEC 2022 benchmark functions.

Type	Function	Description	Optimum value
Unimodal Function	f_1	Shifted and rotated Zakharov	300
Basic Function	f_2	Shifted and rotated Rosenbrock	400
Basic Function	f_3	Shifted and rotated expanded Schaffer's f_6	600
Basic Function	f_4	Shifted and rotated non-continuous Rastrigin	800
Basic Function	f_5	Shifted and rotated Levy	900
Hybrid Function	f_6	Hybrid Function 1 ($N = 3$)	1800
Hybrid Function	f_7	Hybrid Function 2 ($N = 6$)	2000
Hybrid Function	f_8	Hybrid Function 3 ($N = 5$)	2200
Composition Function	f_9	Composition Function 1 ($N = 5$)	2300
Composition Function	f_{10}	Composition Function 2 ($N = 4$)	2400
Composition Function	f_{11}	Composition Function 3 ($N = 5$)	2600
Composition Function	f_{12}	Composition Function 4 ($N = 6$)	2700

TABLE 2. The default parameter settings of the PSO-sono variants.

Algorithm	The default parameter settings
IPSO-sono	$N = 100$, $iw \in [0.4, 0.9]$ and ring topology
PSO-sono	$N = 100$, $iw \in [0.4, 0.9]$, $\epsilon = D/N \cdot 0.01$, $r = 0.5$, $V_{\min} = -30$, $V_{\max} = 30$ and ring topology

The above functions can be used with different values of D , in this study we set it to 20. The search space for the functions in the set is $[-100, 100]^D$. The details of these functions can be found in [15]. A brief description is given in Table 1.

In this study, the competing optimization approaches run for 1 000 000 function evaluations. Each experiment is repeated 30 times, then the pairwise Wilcoxon signed rank test [38] (with $\alpha = 5\%$) is used to validate the results. Moreover, Laplace's Rule of Succession is used to compare the different optimization algorithms as described in [22].

All algorithms have been implemented in the Matlab programming language (Matlab R2017b). The programs have been run on an HP Desktop with 3.6 GHz Intel Core *i7* and 32 GB RAM.

5.1. IPSO-sono vs. PSO-sono

In this section, IPSO-sono is compared against the original PSO-sono algorithm. The parameter settings of these two algorithms are listed in Table 2. For PSO-sono we used the values suggested by the authors in [20].

First the signatures [2] of the two approaches are generated and depicted in Figure 6. The two signatures have been generated using 5 runs, a swarm size of 50 particles and 1000 function evaluations. The signature of IPSO-sono is more uniform covering the whole search space, while the signature of PSO-sono shows some biases towards the corners of the search space. Focusing on certain parts of the search space is considered to be harmful and should be avoided.

Table 3 summarizes the median and minimum objective function values reached by IPSO-sono and PSO-sono when applied to the IEEE CEC 2022 test set. The results of the Wilcoxon test and Laplace's Rule of Succession are reported in Table 4. The results of the two tables show that IPSO-sono outperformed PSO-sono on 9 functions, while performing comparably on 2 functions. The only function where PSO-sono performs better was f_8 .

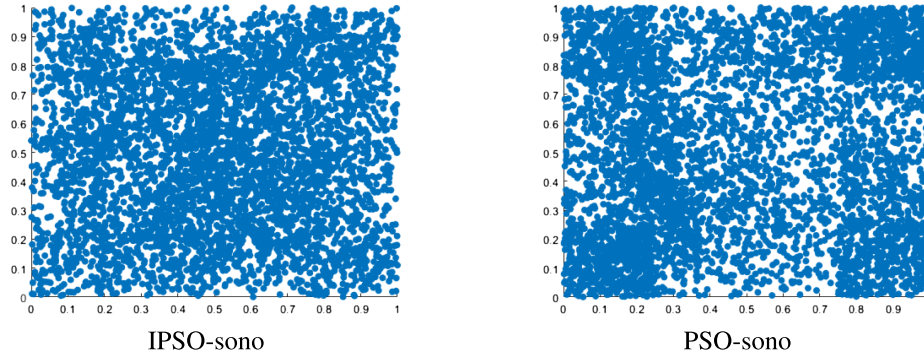


FIGURE 6. The signatures of IPSO-sono and PSO-sono.

TABLE 3. Comparison between the median and best objective function values obtained by IPSO-sono and PSO-sono on the IEEE CEC 2022 functions.

Function	IPSO-sono median	PSO-sono median	IPSO-sono best	PSO-sono best
f_1	3.000000e+02	3.000000e+02	3.000000e+02	3.000000e+02
f_2	4.512388e+02	4.521901e+02	4.000000e+02	4.000000e+02
f_3	6.000000e+02	6.000120e+02	6.000000e+02	6.000000e+02
f_4	8.129345e+02	8.194017e+02	8.049748e+02	8.119395e+02
f_5	9.000000e+02	9.000000e+02	9.000000e+02	9.000000e+02
f_6	1.871739e+03	2.710879e+03	1.813307e+03	1.830137e+03
f_7	2.025320e+03	2.035596e+03	2.003305e+03	2.023337e+03
f_8	2.225003e+03	2.222185e+03	2.222229e+03	2.220813e+03
f_9	2.484040e+03	2.485231e+03	2.482070e+03	2.482301e+03
f_{10}	2.500322e+03	2.500450e+03	2.500239e+03	2.500265e+03
f_{11}	2.900000e+03	2.900000e+03	2.900000e+03	2.600000e+03
f_{12}	2.953127e+03	2.967288e+03	2.938418e+03	2.951180e+03

To study the convergence behavior of the two algorithms, a set of six represented functions has been depicted in Figure 7. The figure shows the average minimum objective function obtained by each algorithm for each value of nfe . It shows that IPSO-sono generally reaches better solutions than PSO-sono.

Another important factor to consider is the diversity of the swarm defined (as in [27]) by:

$$DI = \sqrt{\frac{1}{P} \sum_{i=1}^P \sum_{j=1}^D (x_{i,j} - \bar{x}_j)^2}, \quad (25)$$

where \bar{x}_j is the j -th component of the mean vector of the solutions in the swarm. Figure 8 depicts the average diversity of the swarm for the two algorithms on the same six functions shown in Figure 7.

The figure shows that IPSO-sono maintains the diversity of its swarm for a longer period than PSO-sono, thus, reducing the possibility of premature convergence.

5.2. The algorithmic overhead

In this section, the efficiency of IPSO-sono and PSO-sono are compared. To measure the efficiency, the algorithmic overhead is determined. This overhead is defined as the difference between the time taken by an

TABLE 4. Statistical results of IEEE CEC 2022 problems using Wilcoxon’s Test ($\alpha = 0.05$), where + indicates that IPSO-sono wins, = indicates that both algorithms are the same and – means that PSO-sono wins. The last column shows the prediction of Laplace’s Rule that IPSO-sono will win in the next (future) run.

Function	p -value	Laplace’s (%)
f_1	1.000000e+00(=)	56
f_2	3.160338e−02(+)	63
f_3	1.734398e−06(+)	97
f_4	7.712174e−04(+)	81
f_5	1.953125e−03(+)	47
f_6	1.044440e−02(+)	72
f_7	3.882182e−06(+)	91
f_8	7.690859e−06(−)	16
f_9	4.681835e−03(+)	66
f_{10}	1.956922e−02(+)	78
f_{11}	6.484375e−01(=)	28
f_{12}	2.163022e−05(+)	88
+/=/−	9/2/1	

optimization algorithm using nfe_{\max} function evaluations and the time needed to perform nfe_{\max} function evaluations.

The time required to perform nfe_{\max} function evaluations is called T_{evals} . In this section, T_{evals} is set to the average (over 30 runs) time needed to perform 1 000 000 function evaluation for f_1 .

The mean time needed by an optimization algorithm to run for 1 000 000 function evaluations is recorded as T_{elapsed} .

The difference between T_{elapsed} and T_{evals} , representing the average algorithmic overhead, for D equal to 10 and 20 is shown in Figure 9. Compared to PSO-sono, IPSO-sono exhibits significantly lower overhead (2.66 ms vs. 4.16 ms for $D = 10$, and 3.69 ms vs. 5.78 ms for $D = 20$), resulting in a speedup factor of more than 1.56.

5.3. IPSO-sono vs. other state-of-the-art approaches

In this section, IPSO-sono is compared with five state-of-the-art methods, namely:

- Genetic Algorithm with Multi-Parent Crossover (in short, GA-MPC) [7], is the winner of the IEEE CEC 2011 competition.
- Improved Rao (I-Rao) [31], is a recent variant of the Rao [29] algorithm. The I-Rao has been compared with 13 approaches on 45 CEC problems and 19 real-world problems. According to the Authors I-Rao generally outperformed the 13 approaches.
- L-SHADE [37], is a very popular and effective DE variant, which ranked first in the IEEE CEC 2014 competition.
- Jaya2 [23], which is a very recent, simple and effective variant of the Jaya algorithm [30].
- The Spherical Search (SS) algorithm [14], is a recent optimization algorithm that creates a spherical boundary and then construct candidate solutions on the surface of that boundary.

Table 5 reports the parameter values used for the above approaches. We followed the recommendations of the original works when setting these values.

Table 6 summarizes the median objective functions values obtained by the six competing algorithms. Table 7 reports the results of the Wilcoxon’s test. The results show that IPSO-sono outperformed GA-MPC on 6

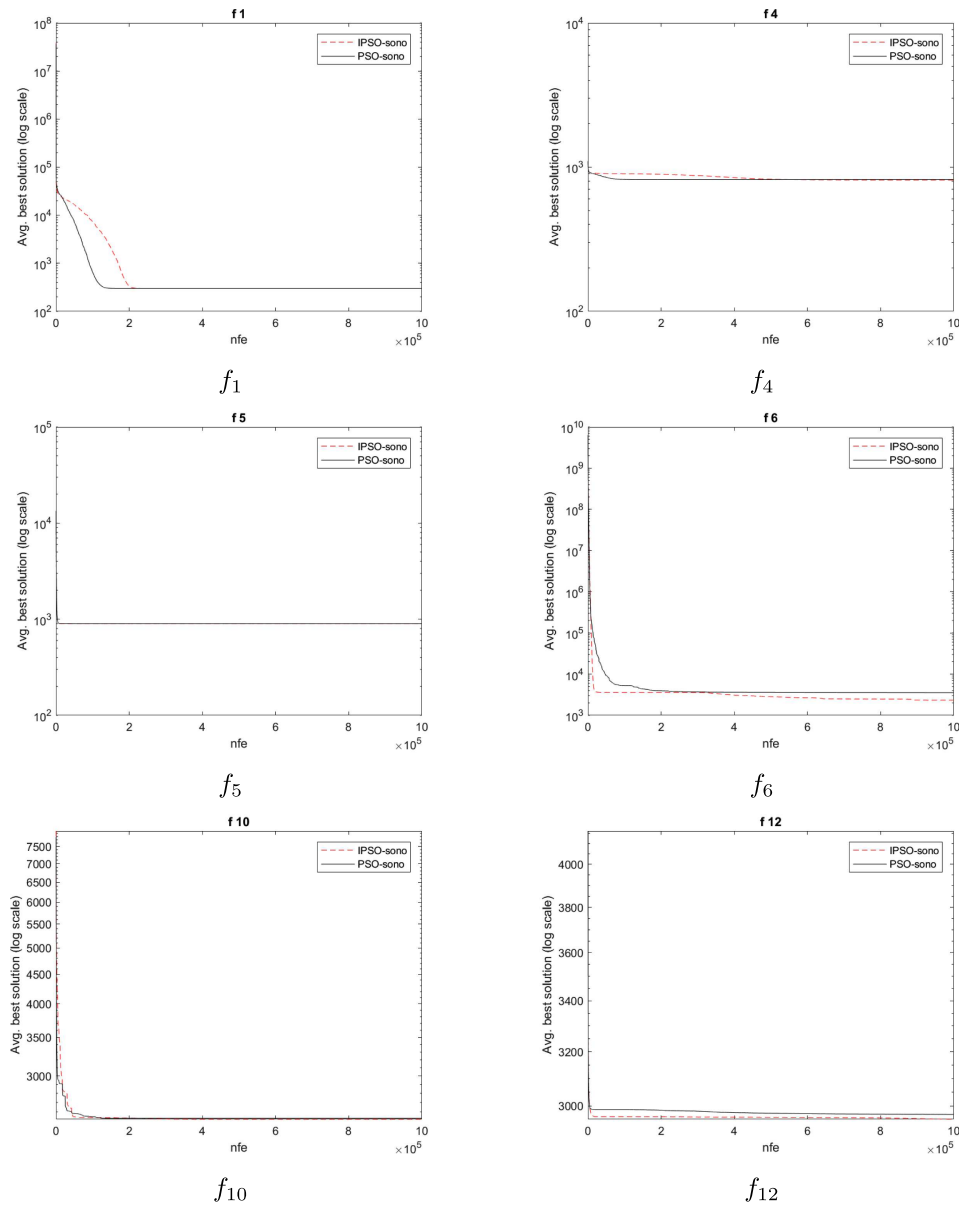


FIGURE 7. Best function value curves (averaged across 30 runs) of IPSO-sono and PSO-sono for selected CEC 2022 benchmark functions ($D = 20$).

functions, while being outperformed on 3 problems. Compared with I-Rao, IPSO-sono performed better on 7 functions while performing worse on 3 functions. IPSO-sono performed better than Jaya2 on 5 functions, while Jaya2 performed better on 4 functions. Compared to SS, the proposed approach performed better on only two functions. Table 7 shows that L-SHADE is a clear winner on this set of benchmark functions.

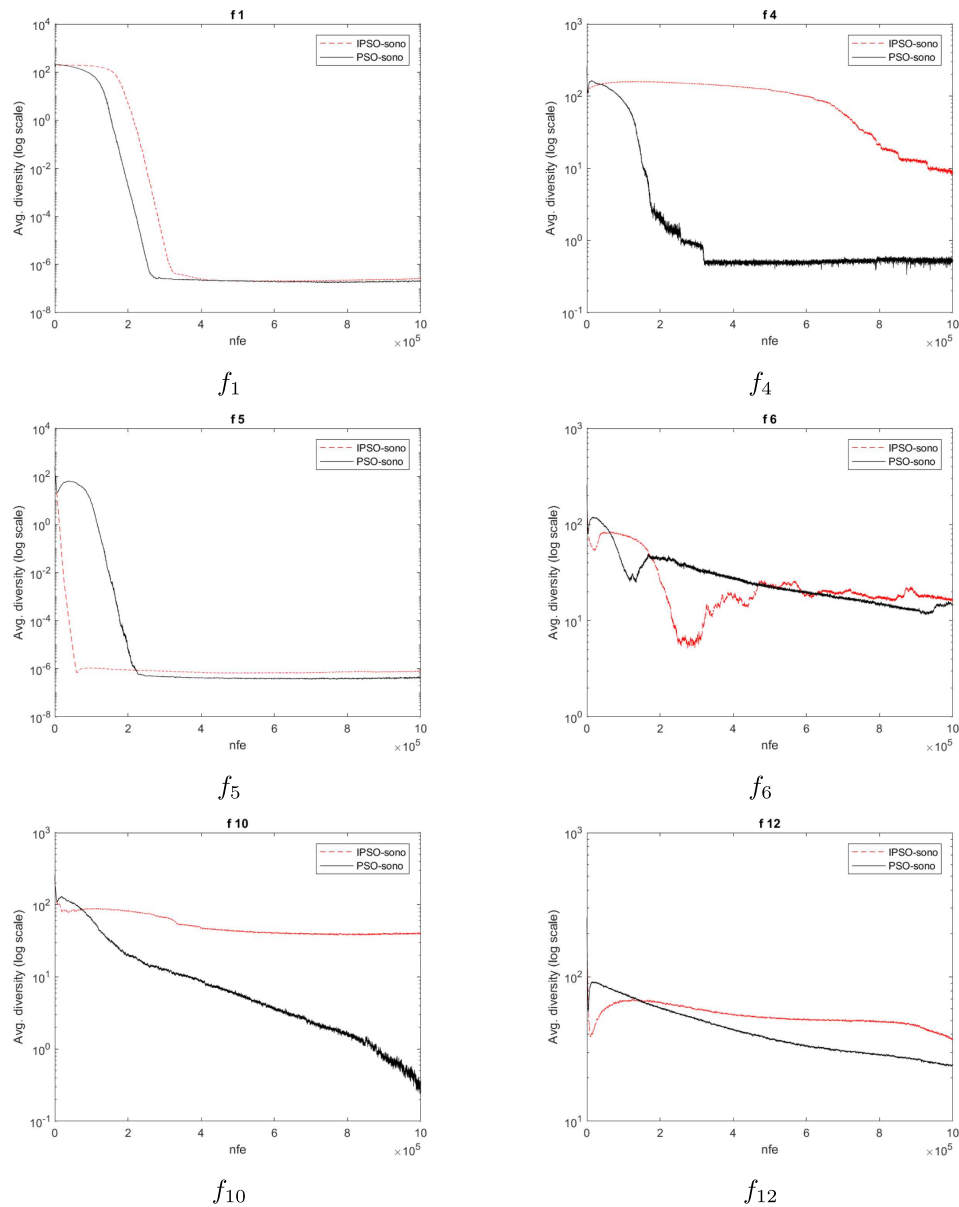


FIGURE 8. Diversity curves (averaged across 30 runs) of IPSO-sono and PSO-sono for selected CEC 2022 benchmark functions ($D = 20$).

5.4. Real-world optimization problems

Any optimization algorithm should be tested on a diverse set of real-world optimization problems to validate any conclusion regarding its performance. One of the best set of diverse problems is the the 22² IEEE CEC 2011 real-world problems [3] summarized in Table 8. This set includes problems from different areas, *e.g.*, chemical

² We did not test our algorithms on T_3 due to its computational cost.

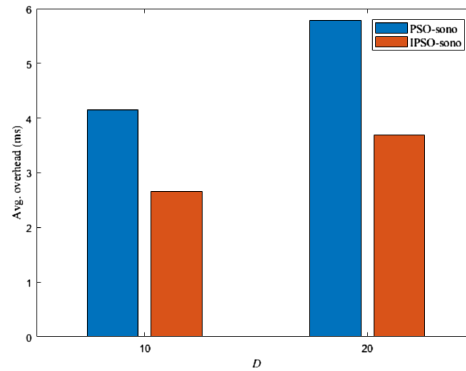


FIGURE 9. Average algorithmic overhead (in seconds) for PSO-sono and IPSO-sono, over increasing dimensionality values.

TABLE 5. Parameter settings for the state-of-the-art approaches.

Algorithm	Parameters
GA-MPC	Population size = 90 and $p = 0.1$.
IRao	$NP = 50$.
Jaya2	$P_{\max} = 100$ and $P_{\min} = 3$.
L-SHADE	$NP = 18 \times D$, $p = 0.11$, and number of historical circle memories is 5.
SS	$N_{init} = 100$, $p = 0.1$, $rank = 0.5 \times D$, and $c = 0.5$.

TABLE 6. Comparison between the median objective function values obtained by IPSO-sono and other metaheuristics on the IEEE CEC 2022 benchmark functions.

Function	IPSO-sono	GA-MPC	I-Rao	Jaya2	L-SHADE	SS
f_1	3.00e+02	3.00e+02	3.00e+02	3.00e+02	3.00e+02	3.00e+02
f_2	4.51e+02	4.49e+02	4.49e+02	4.49e+02	4.49e+02	4.49e+02
f_3	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.00e+02
f_4	8.13e+02	8.23e+02	8.52e+02	8.16e+02	8.04e+02	8.75e+02
f_5	9.00e+02	9.04e+02	9.00e+02	9.00e+02	9.00e+02	9.00e+02
f_6	1.88e+03	3.27e+03	2.49e+04	3.34e+03	1.80e+03	1.80e+03
f_7	2.02e+03	2.02e+03	2.03e+03	2.02e+03	2.00e+03	2.03e+03
f_8	2.23e+03	2.22e+03	2.23e+03	2.23e+03	2.22e+03	2.23e+03
f_9	2.48e+03	2.48e+03	2.48e+03	2.48e+03	2.48e+03	2.48e+03
f_{10}	2.50e+03	2.50e+03	2.50e+03	2.50e+03	2.50e+03	2.50e+03
f_{11}	2.90e+03	2.90e+03	2.90e+03	2.90e+03	2.90e+03	2.90e+03
f_{12}	2.95e+03	2.95e+03	2.94e+03	2.94e+03	2.93e+03	2.94e+03

systems, TLC devices, spacecraft trajectory. They also have different values for D ranging from 1 to 216. The constraint-violation handling used is the one provided by the CEC 2011 official Matlab implementation.

As suggested by Das and Suganthan [3], each run is repeated for 25 runs using $nfe_{\max} = 150\,000$.

First, PSO-sono and IPSO-sono are compared on the IEEE CEC 2011 problems. Table 9 reports the median and best objective function values by the two approaches. The results of the Wilcoxon's Test are summarized in

TABLE 7. Statistical results of the IEEE CEC 2022 using Wilcoxon’s Test ($\alpha = 0.05$), where + indicates that IPSO-sono wins, = indicates that both algorithms are the same and – means that other algorithm wins.

Function	GA-MPC	I-Rao	Jaya2	L-SHADE	SS
f_1	5.96e-05(+)	1.00e+00(=)	1.00e+00(=)	1.00e+00(=)	1.00e+00(=)
f_2	1.73e-06(-)	1.49e-05(-)	1.73e-06(-)	1.73e-06(-)	1.73e-06(-)
f_3	1.73e-06(+)	1.25e-04(+)	6.23e-04(+)	1.00e+00(=)	5.00e-01(=)
f_4	1.02e-05(+)	1.73e-06(+)	5.26e-03(+)	1.73e-06(-)	1.73e-06(+)
f_5	2.56e-06(+)	1.16e-05(+)	2.08e-06(+)	1.00e+00(=)	1.00e+00(=)
f_6	1.89e-04(+)	1.92e-06(+)	4.07e-05(+)	1.73e-06(-)	1.73e-06(-)
f_7	7.19e-01(=)	5.79e-05(+)	3.39e-01(=)	1.73e-06(-)	1.16e-01(=)
f_8	1.80e-05(-)	2.35e-06(+)	6.29e-01(=)	1.73e-06(-)	2.70e-02(+)
f_9	1.73e-06(-)	1.73e-06(-)	1.73e-06(-)	1.73e-06(-)	1.73e-06(-)
f_{10}	7.97e-01(=)	3.93e-01(=)	6.42e-03(-)	4.86e-05(-)	4.90e-04(-)
f_{11}	2.85e-04(+)	3.13e-02(+)	7.81e-03(+)	1.00e+00(=)	1.56e-02(-)
f_{12}	3.60e-01(=)	5.71e-04(-)	1.13e-05(-)	1.73e-06(-)	1.73e-06(-)
+/=/-	6/3/3	7/2/3	5/3/4	0/4/8	2/4/6

TABLE 8. Summary of the CEC 2011 real-world problems.

Problem	Description	D	Constraints
T_1	A frequency-modulated sound waves problem	6	Bound-constrained
T_2	A Lennard-Jones potential problem	30	Bound-constrained
T_3	A bifunctional catalyst blend control problem	1	Bound-constrained
T_4	A stirred tank reactor control problem	1	Unconstrained
T_5	A Tersoff potential minimization problem	30	Bound-constrained
T_6	A Tersoff potential minimization problem	30	Bound-constrained
T_7	A radar polyphase code design problem	20	Bound-constrained
T_8	A transmission network expansion problem	7	Equality/inequality constraints
T_9	A transmission pricing problem	126	Linear equality constraints
T_{10}	An antenna array design problem	12	Bound-constrained
$T_{11.1}$	A dynamic economic dispatch problem	120	Inequality constraints
$T_{11.2}$	A dynamic economic dispatch problem	216	Inequality constraints
$T_{11.3}$	A static economic dispatch problem	6	Inequality constraints
$T_{11.4}$	A static economic dispatch problem	13	Inequality constraints
$T_{11.5}$	A static economic dispatch problem	15	Inequality constraints
$T_{11.6}$	A static economic dispatch problem	40	Inequality constraints
$T_{11.7}$	A static economic dispatch problem	140	Inequality constraints
$T_{11.8}$	A hydrothermal scheduling problem	96	Inequality constraints
$T_{11.9}$	A hydrothermal scheduling problem	96	Inequality constraints
$T_{11.10}$	A hydrothermal scheduling problem	96	Inequality constraints
T_{12}	A spacecraft trajectory optimization problem	26	Bound-constrained
T_{13}	A spacecraft trajectory optimization problem	22	Bound-constrained

TABLE 9. Comparison between the median and best objective function values obtained by IPSO-sono and PSO-sono on IEEE CEC 2011.

Function	IPSO-sono median	PSO-sono median	IPSO-sono best	PSO-sono best
T_1	3.525653e+00	1.137395e+01	3.561404e-05	0.000000e+00
T_2	-1.543388e+01	-2.640805e+01	-1.878921e+01	-2.841409e+01
T_4	1.394558e+01	1.377076e+01	1.377076e+01	1.377076e+01
T_5	-2.949597e+01	-3.191871e+01	-3.471266e+01	-3.556650e+01
T_6	-2.126204e+01	-2.300593e+01	-2.647627e+01	-2.916612e+01
T_7	1.380087e+00	1.057883e+00	1.035778e+00	7.650146e-01
T_8	2.200000e+02	2.200000e+02	2.200000e+02	2.200000e+02
T_9	7.345011e+04	1.207066e+05	5.310040e+04	8.197735e+04
T_{10}	-2.100682e+01	-2.029619e+01	-2.119516e+01	-2.124577e+01
$T_{11.1}$	5.298315e+04	2.743252e+06	5.193080e+04	1.247925e+06
$T_{11.2}$	2.220234e+07	2.579824e+07	2.136842e+07	2.423546e+07
$T_{11.3}$	1.547669e+04	1.547753e+04	1.544700e+04	1.544443e+04
$T_{11.4}$	1.893293e+04	1.864120e+04	1.864640e+04	1.851259e+04
$T_{11.5}$	3.299777e+04	3.299611e+04	3.291417e+04	3.292042e+04
$T_{11.6}$	1.361637e+05	1.380874e+05	1.317670e+05	1.317072e+05
$T_{11.7}$	1.947539e+06	1.951093e+06	1.929639e+06	1.913456e+06
$T_{11.8}$	9.432657e+05	9.507742e+05	9.369032e+05	9.422754e+05
$T_{11.9}$	1.295551e+06	1.687659e+06	1.178849e+06	1.225939e+06
$T_{11.10}$	9.429457e+05	9.531398e+05	9.397135e+05	9.435308e+05
T_{12}	1.829857e+01	1.770570e+01	1.418579e+01	1.428107e+01
T_{13}	1.966660e+01	2.132887e+01	1.487286e+01	1.561390e+01

Table 10, which show that IPSO-sono generally outperforms PSO-sono especially when the problem size, *i.e.* D , is large. This may suggest that IPSO-sono is more suitable than PSO-sono for problems with higher dimensions.

Secondly, IPSO-sono is compared with the five state-of-the-art approaches described in Section 5.3. Table 11 reports the median objective function values obtained by the competing approaches. Table 12 summarizes the statistical test results. The results show that IPSO-sono outperforms both I-Rao and SS (actually SS performs very poorly on this set of problems). Jaya2 slightly outperforms IPSO-sono (7 wins *vs.* 5). GA-MPC (the winner of the IEEE CEC 2011 competition) performs better than the proposed approach, which manages to outperform GA-MPC on 3 problems. L-SHADE confirms its superiority on this set of problems too.

Finally, one interesting case here is the performance of SS. It performs relatively well on the IEEE CEC 2022 but has a very bad performance on the real-world problems. This may suggest that IEEE CEC 2022 test set is not enough to test the performance of an optimizer. Moreover, as we mentioned from the outset, we believe that any optimization algorithm should be tested on many real-world problems to validate its performance. A set of diverse problems like the IEEE CEC 2011 is extremely useful to compare different optimization problems.

6. CONCLUSIONS AND FUTURE WORK

In this paper we proposed a new variant of PSO-sono that incorporates the following techniques:

- Ring topology: The original PSO-sono uses the fully-connected topology for its better-particle group. This often results in premature convergence. Thus, a ring topology is used to slow down the speed of information exchange between the particles, thus, reducing the likelihood of premature convergence.
- Non-linear ratio, r , reduction: a simpler formula is used to update the parameter r . The formula focuses on exploration at the beginning of a run while gradually switching to exploitation at the end of the run.

TABLE 10. Statistical results of the CEC 2011 problems using Wilcoxon's Test ($\alpha = 0.05$), where + indicates that IPSO-sono wins, = indicates that both algorithms are the same and - means that PSO-sono wins.

Function	p -value
T_1	1.284505e-01(=)
T_2	1.229032e-05(-)
T_4	5.478329e-01(=)
T_5	2.831363e-02(-)
T_6	2.758321e-01(=)
T_7	5.132932e-05(-)
T_8	1.000000e+00(=)
T_9	2.001302e-05(+)
T_{10}	8.085130e-05(+)
$T_{11.1}$	1.229032e-05(+)
$T_{11.2}$	1.229032e-05(+)
$T_{11.3}$	8.611622e-01(=)
$T_{11.4}$	9.042082e-05(-)
$T_{11.5}$	5.097549e-01(=)
$T_{11.6}$	8.705089e-03(+)
$T_{11.7}$	3.260495e-01(=)
$T_{11.8}$	6.450980e-05(+)
$T_{11.9}$	7.224473e-05(+)
$T_{11.10}$	1.259557e-04(+)
T_{12}	7.366172e-01(=)
T_{13}	5.271827e-01(=)
+/=/-	8/9/4

- COBL - a simpler FIS scheme is used, which is based on COBL. The simplified scheme still uses the knowledge of the whole swarm as in the original FIS scheme but it is much easier to understand and implement.

Several additional minor modifications have also been introduced. The newly proposed approach, named IPSO-sono, was evaluated on 12 benchmark functions from the IEEE CEC 2022 competition and 21 real-world problems from the IEEE CEC 2011 suite. The results indicate that IPSO-sono significantly outperforms the original PSO-sono algorithm on the majority of the test problems. Notably, IPSO-sono demonstrates improved efficiency, requiring less computational time than its predecessor. This enhanced performance, despite using a simplified version of the mechanisms implemented in PSO-sono, highlights the effectiveness of the new approach. The findings suggest that IPSO-sono not only delivers superior optimization results but also offers greater computational efficiency, making it a better overall choice for tackling both benchmark and real-world optimization problems.

However, when compared with other metaheuristics the results are mixed. For example, it generally outperforms I-Rao and SS, while performing comparably to Jaya2. The performance of IPSO-sono compared to GA-MPC depends on the problem type. In the case of IEEE CEC 2022, IPSO-sono is generally better. However, on the IEEE CEC 2011, GA-MPC (the winner of the competition) is much better. L-SHADE, a DE variant, is clearly the winner on both set of problems. This confirms the findings of Piotrowski *et al.* [26] where they found that DE variants generally outperform PSO variants on a wide range of problems.

Future work will explore using IPSO-sono to solve more practical problems (*e.g.* we are currently investigating its use for color image quantization). A discrete version of IPSO-sono may also be investigated.

TABLE 11. Comparison between the median objective function values obtained by IPSO-sono and other metaheuristics on the IEEE CEC 2011 benchmark problems.

Function	IPSO-sono	GA-MPC	I-Rao	Jaya2	L-SHADE	SS
T_1	1.27e+00	0.00e+00	9.06e+00	0.00e+00	1.68e-21	1.39e+01
T_2	-1.55e+01	-2.64e+01	-1.09e+01	-2.65e+01	-2.60e+01	-9.01e+00
T_4	1.40e+01	1.40e+01	1.38e+01	1.38e+01	1.43e+01	2.11e+01
T_5	-2.77e+01	-3.43e+01	-2.21e+01	-3.16e+01	-3.65e+01	-2.06e+01
T_6	-2.28e+01	-2.30e+01	-1.73e+01	-2.30e+01	-2.92e+01	-1.53e+01
T_7	1.38e+00	9.22e-01	1.78e+00	1.67e+00	1.20e+00	2.22e+00
T_8	2.20e+02	2.20e+02	2.20e+02	2.20e+02	2.20e+02	2.92e+02
T_9	7.43e+04	4.96e+04	3.03e+03	1.52e+03	2.47e+03	3.25e+06
T_{10}	-2.11e+01	-2.14e+01	-1.47e+01	-2.13e+01	-2.16e+01	-1.54e+01
$T_{11.1}$	5.27e+04	5.26e+04	5.26e+04	5.23e+04	5.20e+04	6.20e+06
$T_{11.2}$	2.24e+07	2.16e+07	1.76e+07	1.76e+07	1.78e+07	5.89e+07
$T_{11.3}$	1.55e+04	1.54e+04	1.55e+04	1.55e+04	1.54e+04	1.57e+04
$T_{11.4}$	1.90e+04	1.85e+04	1.92e+04	1.91e+04	1.81e+04	2.06e+05
$T_{11.5}$	3.30e+04	3.30e+04	3.28e+04	3.28e+04	3.27e+04	1.71e+06
$T_{11.6}$	1.37e+05	1.36e+05	1.35e+05	1.35e+05	1.24e+05	9.51e+05
$T_{11.7}$	1.95e+06	2.04e+06	2.12e+06	1.98e+06	1.85e+06	1.86e+10
$T_{11.8}$	9.43e+05	9.75e+05	1.62e+06	9.98e+05	9.31e+05	1.55e+08
$T_{11.9}$	1.32e+06	1.22e+06	1.98e+06	1.38e+06	9.39e+05	1.56e+08
$T_{11.10}$	9.43e+05	9.70e+05	1.58e+06	1.02e+06	9.32e+05	1.60e+08
T_{12}	1.82e+01	1.62e+01	1.84e+01	1.46e+01	1.61e+01	4.24e+01
T_{13}	1.94e+01	2.08e+01	1.58e+01	2.10e+01	1.43e+01	3.99e+01

TABLE 12. Statistical results of the CEC 2011 problems using Wilcoxon’s Test ($\alpha = 0.05$), where + indicates that IPSO-sono wins, = indicates that both algorithms are the same and - means that other algorithm wins.

Function	GA-MPC	I-Rao	Jaya2	L-SHADE	SS
T_1	6.38e-01(=)	2.30e-02(+)	6.96e-01(=)	5.76e-05(-)	2.26e-05(+)
T_2	1.23e-05(-)	8.09e-05(+)	1.23e-05(-)	1.23e-05(-)	1.23e-05(+)
T_4	9.09e-01(=)	6.89e-01(=)	1.37e-01(=)	9.63e-04(+)	1.23e-05(+)
T_5	2.26e-05(-)	1.23e-05(+)	1.57e-03(-)	1.23e-05(-)	1.23e-05(+)
T_6	6.96e-01(=)	1.23e-05(+)	3.26e-01(=)	1.23e-05(-)	1.23e-05(+)
T_7	1.77e-05(-)	1.23e-05(+)	5.35e-03(+)	1.94e-04(-)	1.23e-05(+)
T_8	1.00e+00(=)	1.00e+00(=)	1.00e+00(=)	1.00e+00(=)	1.18e-05(+)
T_9	8.27e-02(=)	1.23e-05(-)	1.23e-05(-)	1.23e-05(-)	1.23e-05(+)
T_{10}	8.04e-03(-)	4.57e-05(+)	4.76e-01(=)	1.23e-05(-)	1.23e-05(+)
$T_{11.1}$	1.83e-01(=)	5.45e-01(=)	6.93e-02(=)	1.28e-02(-)	1.23e-05(+)
$T_{11.2}$	2.40e-04(-)	1.23e-05(-)	1.23e-05(-)	1.23e-05(-)	1.23e-05(+)
$T_{11.3}$	2.40e-04(-)	2.26e-03(+)	7.37e-01(=)	4.07e-05(-)	1.23e-05(+)
$T_{11.4}$	1.23e-05(-)	1.94e-04(+)	1.10e-02(+)	1.23e-05(-)	1.23e-05(+)
$T_{11.5}$	8.71e-03(-)	1.23e-05(-)	1.23e-05(-)	1.23e-05(-)	1.23e-05(+)
$T_{11.6}$	3.00e-01(=)	8.27e-02(=)	6.31e-03(-)	1.23e-05(-)	1.23e-05(+)
$T_{11.7}$	8.09e-05(+)	2.66e-04(+)	7.42e-03(+)	1.23e-05(-)	1.23e-05(+)
$T_{11.8}$	5.45e-04(+)	1.23e-05(+)	1.40e-04(+)	1.23e-05(-)	1.23e-05(+)
$T_{11.9}$	1.73e-02(-)	1.23e-05(+)	6.00e-01(=)	1.23e-05(-)	1.23e-05(+)
$T_{11.10}$	1.23e-05(+)	1.23e-05(+)	1.23e-05(+)	1.23e-05(-)	1.23e-05(+)
T_{12}	2.16e-04(-)	7.57e-01(=)	1.39e-05(-)	3.62e-05(-)	1.23e-05(+)
T_{13}	7.37e-01(=)	1.73e-02(-)	7.57e-01(=)	1.77e-05(-)	1.23e-05(+)
+ / = / -	3/8/10	12/5/4	5/9/7	1/1/19	21/0/0

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their constructive and helpful comments and suggestions.

CONFLICTS OF INTEREST

The authors declare that they have no conflict of interest.

DATA AVAILABILITY STATEMENT

A Matlab implementation of IPSO-sono is publicly posted at <https://www.mathworks.com/matlabcentral/fileexchange/130364-ipso-sono>.

AUTHOR CONTRIBUTION STATEMENT

M.O. and H.W. developed the idea. M.O. implemented the proposed approach. M.O. and M.A. conducted the experiments. M.O. prepared the manuscript. H.W. reviewed the manuscript and provided feedback.

ETHICAL AND INFORMED CONSENT FOR DATA USED

This article does not contain any studies with human participants or animals performed by any of the authors.

REFERENCES

- [1] R. Cheng and Y. Jin, A social learning particle swarm optimization algorithm for scalable optimization. *Inf. Sci.* **291** (2015) 43–60.
- [2] M. Clerc, Guided Randomness in Optimization. Wiley (2015).
- [3] S. Das and P.N. Suganthan, Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems. Jadavpur University, Nanyang Technological University, Kolkata (2010).
- [4] G. Di Caro and M. Dorigo, The ant colony optimization meta-heuristic, in *New Ideas in Optimization*, edited by F. Glover D. Corne and M. Dorigo. McGraw Hill, London (1999) 11–32.
- [5] T. Dokeroglu, E. Sevinc, T. Kucukyilmaz and A. Cosar, A survey on new generation metaheuristic algorithms. *Comput. Ind. Eng.* **137** (2019) 106040.
- [6] A. Engelbrecht, *Computational Intelligence: An Introduction*. Wiley (2007).
- [7] S.M. Elsayed, R.A. Sarker and D.L. Essam, GA with a new multi-parent crossover for solving IEEE-CEC2011 competition problems, in *Congress on Evolutionary Computation (CEC)*. IEEE (2011) 1034–1040.
- [8] F. Glover and M. Laguna, *Tabu Search*. Springer US, Boston, MA (1998) 2093–2229.
- [9] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, New York (1989).
- [10] P. Hansen and N. Mladenović, *An Introduction to Variable Neighborhood Search*. Springer US, Boston, MA (1999) 433–458.
- [11] K. Hussain, M.N. Mohd Salleh, S. Cheng and Y. Shi, Metaheuristic research: a comprehensive survey. *Artif. Intell. Rev.* **52** (2019) 2191–2233.
- [12] J. Kennedy and R. Eberhart, Particle swarm optimization, in *International Joint Conference on Neural Networks (IJCNN)*. IEEE (1995) 1942–1948.
- [13] J. Kennedy and R. Mendes, Population structure and particle swarm performance, in *Congress on Evolutionary Computation (CEC)*. Vol. 2. IEEE (2002) 1671–1676.
- [14] A. Kumar, R. Misra, D. Singh, S. Mishra and S. Das, The spherical search algorithm for bound-constrained global optimization problems. *Appl. Soft Comput.* **85** (2019) 105734.
- [15] A. Kumar, K. Price, A. Mohamed, A. Hadi and P. Suganthan, Problem definitions and evaluation criteria for CEC 2022 competition on single objective bound constrained numerical optimization. Technical report (2021).
- [16] J.J. Liang, A.K. Qin, P.N. Suganthan and S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Trans. Evol. Comput.* **10** (2006) 281–295.
- [17] H. Liu, X. Zhang and L. Tu, A modified particle swarm optimization using adaptive strategy. *Expert Syst. App.* **152** (2020) 533–548.
- [18] N. Lynn and P. Suganthan, Ensemble particle swarm optimizer. *Appl. Soft Comput.* **55** (2017) 533–548.
- [19] N. Lynn, M.Z. Ali and P.N. Suganthan, Population topologies for particle swarm optimization and differential evolution. *Swarm Evol. Comput.* **39** (2018) 24–35.

- [20] Z. Meng, Y. Zhong, G. Mao and Y. Liang, PSO-sono: a novel PSO variant for single-objective numerical optimization. *Inf. Sci.* **586** (2022) 176–191.
- [21] M. Nasir, S. Das, D. Maity, U. Sengupta, S. Halder and P. Suganthan, A dynamic neighborhood learning based particle swarm optimizer for global numerical optimization. *Inf. Sci.* **209** (2012) 16–36.
- [22] M. Omran and M. Clerc, Laplace’s rule of succession: a simple and efficient way to compare metaheuristics. *Neural Comput. App.* **35** (2023) 11807–11814.
- [23] M. Omran and G. Iacca, An improved jaya optimization algorithm with ring topology and population size reduction. *J. Intell. Syst.* **31** (2022) 1178–1210.
- [24] E. Osaba, E. Villar-Rodriguez, J. Del Ser, A. Nebro, D. Molina, A. LaTorre, P. Suganthan, C. Coello Coello and F. Herrera, A tutorial on the design, experimentation and application of metaheuristic algorithms to real-world optimization problems. *Swarm Evol. Comput.* **64** (2021) 100888.
- [25] J. Peng, Y. Li, H. Kang, Y. Shen, X. Sun and Q. Chen, Impact of population topology on particle swarm optimization and its variants: an information propagation perspective. *Swarm Evol. Comput.* **69** (2022) 100990.
- [26] A. Piotrowski, J. Napiorkowski and A. Piotrowska, Particle swarm optimization or differential evolution – a comparison. *Eng. App. Artif. Intell.* **121** (2023) 106008.
- [27] R. Poláková, J. Tvrdik and P. Bujok, Adaptation of population size according to current population diversity in differential evolution, in Proceedings of the IEEE 2017 Symposium Series on Computational Intelligence (SSCI). IEEE (2017) 2627–2634.
- [28] S. Rahnamayan, J. Jesuthasan, F. Bourennani, H. Salehinejad and G. Naterer, Computing opposition by involving entire population, in Congress on Evolutionary Computation (CEC). IEEE (2014) 1800–1807.
- [29] R. Rao, Rao algorithms: three metaphor-less simple algorithms for solving optimization problems. *Int. J. Ind. Eng. Comput.* **11** (2020) 107–130.
- [30] R.V. Rao, Jaya: a simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *Int. J. Ind. Eng. Comput.* **7** (2016) 19–34.
- [31] R.V. Rao and R.B. Pawar, Improved rao algorithm: a simple and effective algorithm for constrained mechanical design optimization problems. *Soft Comput.* **27** (2022) 3847–3868.
- [32] T.M. Shami, A.A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M.A. Summakieh and S. Mirjalili, Particle swarm optimization: a comprehensive survey. *IEEE Access* **10** (2022) 10031–10061.
- [33] T. Si, D. Bhattacharya, S. Nayak, P.B.C. Miranda, U. Nandi, S. Mallik, U. Maulik and H. Qin, Pcobl: a novel opposition-based learning strategy to improve metaheuristics exploration and exploitation for solving global optimization problems. *IEEE Access* **11** (2023) 46413–46440.
- [34] K. Socha and M. Dorigo, Ant colony optimization for continuous domains. *Eur. J. Oper. Res.* **185** (2008) 1155–1173.
- [35] R. Storn and K. Price, Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, ICSI, Berkeley (1995).
- [36] B. Suman and P. Kumar, A survey of simulated annealing as a tool for single and multiobjective optimization. *J. Oper. Res. Soc.* **57** (2021) 1143–1160.
- [37] R. Tanabe and A. Fukunaga, Improving the search performance of SHADE using linear population size reduction, in Congress on Evolutionary Computation (CEC). IEEE (2014) 1658–1665.
- [38] F. Wilcoxon, Individual comparisons by ranking methods. *Biometrics Bull.* **1** (1945) 80–83.
- [39] X.-S. Yang, Nature-Inspired Optimization Algorithms. Elsevier (2014).

Please help to maintain this journal in open access!



This journal is currently published in open access under the Subscribe to Open model (S2O). We are thankful to our subscribers and supporters for making it possible to publish this journal in open access in the current year, free of charge for authors and readers.

Check with your library that it subscribes to the journal, or consider making a personal donation to the S2O programme by contacting subscribers@edpsciences.org.

More information, including a list of supporters and financial transparency reports, is available at <https://edpsciences.org/en/subscribe-to-open-s2o>.