Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Self-Organizing Migrating Algorithm with narrowing search space strategy for robot path planning

Quoc Bao Diep [a,*], Thanh Cong Truong [b], Swagatam Das [c], Ivan Zelinka [d,a]

[a] VSB - Technical University of Ostrava, Faculty of Electrical Engineering and Computer Science, Ostrava, Czech Republic
[b] University of Finance - Marketing, Ho Chi Minh City, Viet Nam
[c] Indian Statistical Institute, Electronics and Communication Sciences Unit, Kolkata, India
[d] Ton Duc Thang University, Faculty of Electrical & Electronics Engineering, Ho Chi Minh City, Viet Nam

ARTICLE INFO

Article history:
Received 27 June 2020
Received in revised form 28 October 2021
Accepted 29 November 2021
Available online 9 December 2021

Keywords:
Self-Organizing Migrating Algorithm
Optimization algorithm
Swarm intelligence
Numerical optimization
Path planning
Drone

ABSTRACT

This article introduces a version of the Self-Organizing Migrating Algorithm with a narrowing search space strategy named iSOMA. Compared to the previous two versions, SOMA T3A and Pareto that ranked 3rd and 5th respectively in the IEEE CEC (Congress on Evolutionary Computation) 2019 competition, the iSOMA is equipped with more advanced features with notable improvements including applying jumps in the order, immediate update, narrowing the search space instead of searching on the intersecting edges of hyperplanes, and the partial replacement of individuals in the population when the global best improved no further. Moreover, the proposed algorithm is organized into processes named initialization, self-organizing, migrating, and replacement. We tested the performance of this new version by using three benchmark test suites of IEEE CEC 2013, 2015, and 2017, which, together contain a total of 73 functions. Not only is it superior in performance to other SOMAs, but iSOMA also yields promising results against the representatives of well-known algorithmic families such as Differential Evolution and Particle Swarm Optimization. Moreover, we demonstrate the application of iSOMA for path planning of a drone, while avoiding static obstacles and catching the target.

1. Introduction

With the continuous development of science and technology, many practical problems arise challenging and most of them can be transformed into optimization problems [1]. The Swarm Intelligence (SI) is one of the most effective and well-attended methods to find the global optimal solution to such problems, such as Differential Evolution (DE) [2,3], Artificial Bee Colony (ABC) [4,5], Particle Swarm Optimization (PSO) [6,7], and Self-Organizing Migrating Algorithm (SOMA) [8,9] that is a subject of the reported research here.

Proposed in the 2000s, SOMA,[1] a representative of the SI, is a population-based optimization algorithm, which mimics the competition–cooperation behavior among individuals in the population of creatures to find the optimal solution. Over many migration loops, the initial candidate solutions are optimized, making these solutions better and better over time. With a non-gradient-based mechanism and flexibility property, i.e., solving complex functions without using complex math equations, SOMA demonstrates its outstanding performance and is applied in many different fields such as the reliability–redundancy allocation problem [10], StarCraft: Brood War — computer games [11,12], and drive robots to avoid dynamics obstacles [13,14].

On the one hand, real-world problems are emerging more and more complex, requiring not only fast real-time computations but also high accuracy of results and capable to escape from local traps. The canonical versions of the SOMA algorithm have been somewhat less performance against these issues.

On the other hand, it requires simplicity, ease of programming, and ease of use for many different application areas. Besides, the adaptation of the control parameters of the algorithm is required, because not all application developers are experts in the field of the optimization algorithm. Therefore, improving the algorithm to satisfy these illustration requirements mentioned above is essential.

Many improved versions have been proposed to boost up the performance of the algorithm and overcome some limitations arising during the application process, such as self-adapting SOMA [15], C-SOMAQI [16], the version of the leader selection in the SOMA [17], SOMA with non-binary perturbation [18], and self-adaptive parameters to SOMA [19]. In particular, the two latest versions, team to team adaptive — SOMA T3 A [20,21] and Pareto-based SOMA [22,23], have made great strides, holding *3rd* (the same ranking with HyDE-DF [24]) and *5th* out of

* Corresponding author.
E-mail addresses: diepquocbao@gmail.com (Q.B. Diep), ttcong@ufm.edu.vn (T.C. Truong), swagatam.das@isical.ac.in (S. Das), ivan.zelinka@tdtu.edu.vn, ivan.zelinka@vsb.cz (I. Zelinka).

[1] http://somaalgorithm.com/.

https://doi.org/10.1016/j.asoc.2021.108270

38 algorithms participating in the 100-Digit Challenge respectively, which reported in [25] including results from the 2019 Congress on Evolutionary Computation (CEC 2019), the 2019 Genetic and Evolutionary Computation Conference (GECCO 2019) and the 2019 Swarm, Evolutionary and Memetic Computing Conference (SEMCCO 2019).

From SOMA 21 years history, it is visible that this algorithm belongs (based on various comparative studies) to the most efficient SI and is also highly applicable to various problems of industrial practice as well as academic problems.

However, these are not the final versions of SOMA. They have proven their effectiveness in the 100-Digit Challenge, which does not mean that those algorithms will be victorious at all different test suites. Furthermore, many real-world applications increasingly require algorithms to be more efficient, solve problems faster, and more accurately. That prompted us to develop and propose the next generation of SOMA T3 A, named iSOMA.

And what about drones? How to apply iSOMA in drones? These questions will be answered in the next part of the article. It can be revealed that one of the most important issues for drone applications is catching up with targets and avoiding multiple obstacles.

The rest of the article is organized into the following sections. Section 2 describes the principles of the SOMA algorithm as well as its strengths and weaknesses, which underlie the proposed algorithm. Section 3 presents the improved version of the self-organizing migrating algorithm, iSOMA. Section 4 shows the experiment setup. Comparison results and discussion are presented in Section 5. Application of the iSOMA for path planning of drones is presented in Section 6. Finally, the work is concluded in Section 7.

2. The canonical SOMA

2.1. The principle

Self-Organizing Migrating Algorithm, a swarm-based intelligence optimization algorithm, works based on the interaction between individuals in the population according to a given rule to find an optimal solution to the given problem [8,9]. The mechanism constituting the SOMA lies in how to select individuals as a leader and migrants, how migrants move to the leader, as well as how to update better individuals into the population and eliminate the bad one. These processes are performed under loops named migration loops. Then, through many migration loops, these solutions are becoming better and better than initial solutions. This section briefly describes the principle of SOMA, shaping the basis for the analysis of SOMA's strengths and weaknesses.

A population is generated at the beginning of the algorithm, containing individuals as candidate solutions to a given problem, according to Eq. (1). Each variable (dimension) of the problem has its boundary, which is also the search range of the algorithm. They are then evaluated by the given fitness function and enter the first migration loop.

$$P = x_j^{(lo)} + rand(x_j^{(hi)} - x_j^{(lo)}) \tag{1}$$

where:

- P: the SOMA's original population,
- $x_j^{(lo)}$: the lowest limit value,
- $x_j^{(hi)}$: the highest limit value,
- $rand$: a random number, from 0 to 1.

In each migration loop, the individual with the lowest fitness value in the population is chosen as the leader, and the remaining individuals are those who are traveling. They will jump by step toward the leader using the *Step* parameter (specified the granularity) before *PathLength* (a limit of distance) is reached. Instead of jumping directly toward the leader, another parameter is used to generate perturbation moves, named $PRTVector_j$, forcing the individuals to move in the $N - k$ subspace where each pair is perpendicular to the original space, as shown in Eq. (2).

$$if\ rand_j < PRT;\ \ PRTVector_j = 1;\quad else,\ 0. \tag{2}$$

The probability of each move is determined by the *PRT* parameter. A number is randomly generated and compared to this threshold. If it is less than *PRT*, the jump in that dimension is performed, and vice versa. This helps to maintain the diversity of the population while creating better new individuals. The Eq. (3) describes this moving process.

$$x_{n,j}^{ML+1} = x_{c,j}^{ML} + (x_{l,j}^{ML} - x_{c,j}^{ML})\ t\ PRTVector_j \tag{3}$$

where:

- $x_{n,j}^{ML+1}$: position in the next migration loop,
- $x_{c,j}^{ML}$: the migrant position in the current migration loop,
- $x_{l,j}^{ML}$: the leader position in the current migration loop,
- t: moving step, from 0, by *Step*, to *PathLength*.

After each individual completes its moves, the best position in the moving trajectory is chosen to compare with the initial position. This one will be replaced by the better new position, otherwise, the algorithm will skip the new position and continue the process for the remaining individuals.

After all individuals have completed the jumping, a new migration loop is started, the new leader will be chosen again, and the migration process will continue until SOMA satisfies the specified termination condition. SOMA AllToOne (SOMA ATO) is the name of that technique. Rather than all individuals moving toward the leader, under another approach, all individuals move toward each other, regardless of whether the individual is better or worse. SOMA AllToAll (SOMA ATA) is the name of this technique.

2.2. Weakness of SOMA

Stopping Criteria:

As described in the previous subsection, after each individual has completed his movement, the best position in this path is selected for comparison with the original. It is clear that to find a better position, the SOMA needs to call the cost function many times (known as function evaluations - *FEs*, each execution of the cost function is considered one *FE*).

For example, with the standard setting of SOMA: $Pathlength = 3.0$ and $Step = 0.11$, each traveling individual has 27 different positions on its jumping path, which means that the SOMA has to spend 27*FEs* to evaluate these positions to find the better one. Meanwhile, other algorithms only use one *FE* to improve their candidate solutions such as DE, PSO, and ABC, or some algorithm-specific parameter-less optimization techniques like Jaya [26–28] and Rao algorithm [29]. This characteristic causes the algorithm to soon face the stop condition of maximum function evaluations (*MaxFEs*) and the premature convergence scenario before it can fine-tune the solution in the exploitation phase at the end of the optimization process.

Move on the Edge:

On the other side, in each variable of the optimization problem, $PRTVector_j$ accepts only one of the two values of 0 and 1.

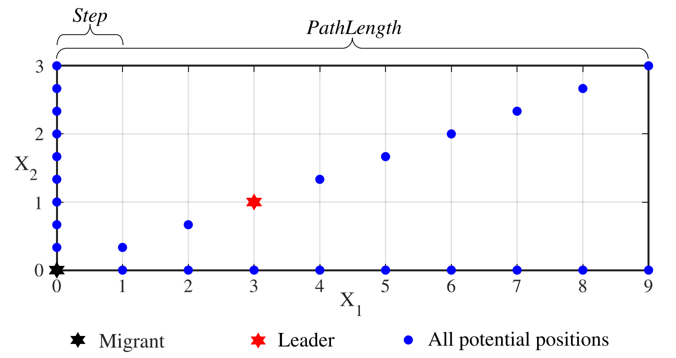


Fig. 1. All possible positions in 2D space. Setting parameters: *Step* = 0.33 and *PathLength* = 3.0.

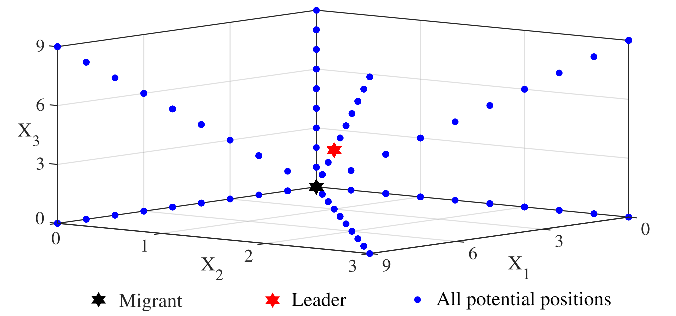


Fig. 2. All possible positions in 3D space. Setting parameters: *Step* = 0.33 and *PathLength* = 3.0.

Therefore, this variable will be updated with a multiple of *Step* if the value of $PRTVector_j$ is 1 (and vice versa it will remain in its position). Geometrically, this means that traveling individuals will move on the intersection edges of hyperplanes created by pairs of sides of variables, as shown in Figs. 1 and 2.

It can be visualized as a line-search strategy. Moving only on the edges of hyperplanes without moving into the inner space highly limits the searching ability of the SOMA and leads to the risk of missing out on the potential search space.

Search for Nonsense:

Besides, one of the major weaknesses of SOMA is that nonsense moves are taken from better individuals to the worse one, as shown in Fig. 3. This leads to a waste of computational time because the algorithm spends a lot of *FEs* on these nonsense moves. It causes the algorithm to face the risk of being stopped before finding the optimal solution to the given problem.

3. The proposed algorithm: iSOMA

The name "Self-Organizing Migrating Algorithm" covers the algorithm's entire operation. Consequently, we divide the algorithmic framework of iSOMA into four processes and call them the initialization process, self-organizing process, migrating process, and replacement (update) process. They work in migration loops. Individuals from the initial population will migrate to each other in each migration loop to explore promising subspaces and then exploit these spaces to find the global optimal solution. As seen in Fig. 4, these procedures were repeated until the specified stop conditions are met.

3.1. The initialization process

The iSOMA starts with the initialization process. Within the control parameters established, an initial population of potential solutions is randomly generated using uniformly distributed random numbers to scatter initial individuals in the whole given search space, by applying Eq. (1).

This population is then evaluated by the given fitness function. The global best optimal solution (the individual with the smallest fitness value) is recorded and the algorithm enters the first migration loop, as described in the next subsection.

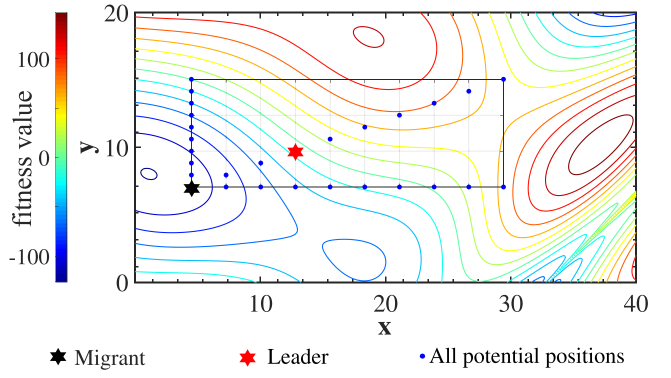


Fig. 3. The meaningless move from the migrant to the leader has a lower fitness value. In this case, there is no position with better fitness value than the migrant itself in the capability search space.

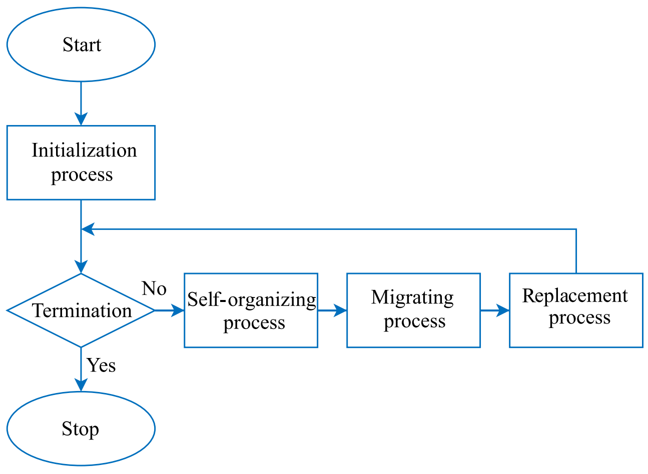


Fig. 4. The flowchart of the iSOMA.

3.2. The self-organizing process

The self-organizing process in the proposed algorithm is the process of determining which individuals will move toward their targets (named as migrants) and which one will become the target (named as leader). In the canonical version of ATO, all individuals are migrants and the best individuals in the population become the leader for each migration loop. This results in limitations as analyzed in the previous section. On the other hand, if all individuals move toward each other as the ATA version, SOMA not only faces the stop condition of *FEs* due to the use of a lot number of jumpings taken place between bad individuals but also faces the premature convergence scenario.

To overcome the mentioned shortcomings, the self-organizing process must both ensure the elimination of bad individuals and maintain the diversity of the population by avoiding only focusing on the global best one. Accordingly, the iSOMA selects the best individuals in a group to move toward the best individual in another group.

To implement this progress, in each migration loop, the iSOMA first randomly selects *m* individuals in the current population as

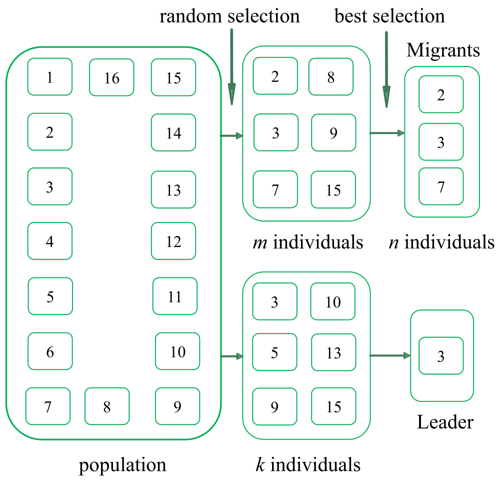


Fig. 5. The self-organizing process.

the subpopulation and then selects the best n out of m ($n \leqslant m$), which will become migrants. For each migrant, the algorithm randomly selects the second subpopulation containing k individuals in the current population (can be overlapped with individuals). The individual with the best fitness value out of k becomes the leader for that migrant. In case the migrant coincides with the leader, the algorithm will choose the second-best individual in k to be the leader. Fig. 5 describes this process.

For problems containing many local traps, the values of m, n, and k should be small. In this context, many moves are performed between random individuals, boosting the exploring ability of the iSOMA in the search space. On the contrary, for simpler problems, the values of m, n, and k should be larger to force the iSOMA to focus on better individuals, increasing the exploiting ability on the promising searched space. These parameters highly impact the performance of the algorithm, besides the other control parameters will be presented in the next subsection.

3.3. The migrating process

The migrating process regulates how the migrant moves toward the leader selected in the previous subsection. This movement type, in the canonical version, is a straight-line-search strategy with dotted-line positions as depicted in Figs. 1 and 2. To enhance the algorithm's search capabilities and restrict the mentioned weakness, we propose the following improvements for the migrating way:

The order of jumps:

Instead of jumping gradually toward the leader as in the canonical version, we propose a method of jumping in order, as shown in Fig. 6. Accordingly, the first position of the migrant is to jump "behind" the leader. After that, the migrant gradually moves toward the leader specified by the given *Step*. In other words, the migrant starts from the farthest step by step approaching its initial position.

Immediately update:

Another valuable improvement derives from terminating the jumping progress of the current migrant and immediately updating its position in the population if the new position is better than the initial position. It is executed by the algorithm that will evaluate the new position found during the migrant's migration and compare it to the initial. It will immediately replace the initial and stop its migration, going to the next migrant.

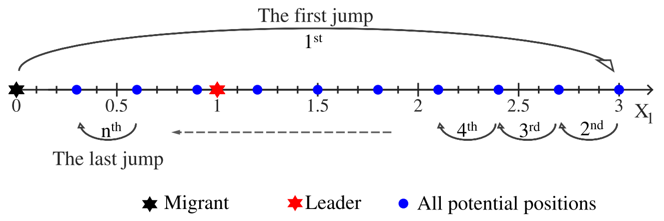


Fig. 6. The order of the jumps in the iSOMA. Setting parameters: $Step = 0.3$, $N_{jump} = 10$, and $PathLength = 3.0$.

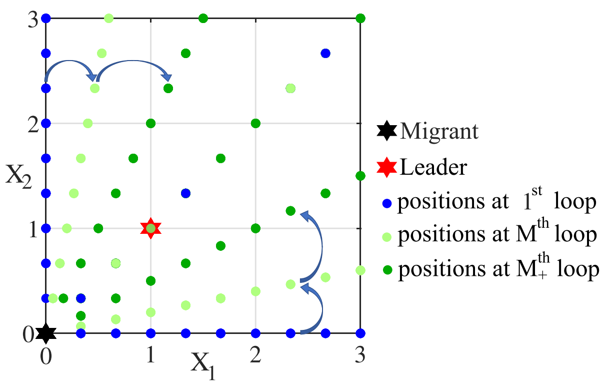


Fig. 7. All possible positions of the offspring over migration loops. Setting parameters: $Step = 0.33$, $PathLength = 3.0$, with adaptive $PRTVector$.

This improvement, incorporated with jumping in order, not only makes the algorithm spend fewer *FEs* to get a better position, but also helps the population preserve diversity, avoiding premature convergence scenarios.

Narrow the search space:

Fig. 7 depicts the narrowing of the search space. In the early stages of the optimization progress, the algorithm should prefer to explore promising subspaces rather than focus on exploiting them. Thus, individuals move on the edges of hyperplanes created by pairs of sides of variables (specified by small *PRT*, resulting in more $PRTVector_j$ equals zero).

Toward the end of optimization progress, iSOMA is more inclined to exploit these promising subspaces. Therefore, the adaptive *PRTVector* parameter is proposed so that individuals can move inside the space created by intersection hyperplanes, instead of just moving on the edges like the canonical version. Eq. (4) is used to enable this feature.

$$if\ rand_j < PRT;\ PRTVector_j = 1;\ else, PRTVector_j = \frac{FEs}{MaxFEs}. \tag{4}$$

where:

- *FEs*: the current function evaluation,
- *MaxFEs*: the maximum of function evaluations.

Besides, the adaptive *PRT* parameter is leveraged in the iSOMA, which was introduced in [20], given in Eq. (5). The *PRT* starts with a number close to 0 and ends with a number close to 1 to avoid the meaningless comparison of $rand_j < PRT$ in Eq. (4). In this version, the *Step* parameter is fixed.

$$PRT = 0.05 + 0.90\ \frac{FEs}{MaxFEs} \tag{5}$$

3.4. The replacement process

The process is to replace some individuals in the current population with new ones. This is a necessary progression to be taken when the algorithm cannot find a better global optimal solution after a certain amount of searching time which can be measured by the number of function evaluations.

Accordingly, after several *FEs*, if the algorithm does not discover a better position than the global best, the iSOMA will randomly replace a certain percentage (10% for example) of the existing individuals in the current population (excluding the global best individual) by the same number of randomly generated individuals in the whole search space (according to Eq. (1)). Using random individuals instead of recorded historical individuals found during the searching progress prevents the algorithm from falling into the current local traps.

The proposed iSOMA is described in Algorithm 1.

Algorithm 1 : iSOMA

```
1: Create and evaluate the initial population $P$
2: Store the best individual as the global best one
3: while stop condition not reached do
4:   Select random $m$ individuals from $P$
5:   Pick the best $n$ out of $m$ individuals as migrants
6:   for $i = 1$ to $n$ migrants do
7:     Select random $k$ individuals from $P$
8:     Picked out the best of $k$ as the leader.
9:     if the leader is the migrant then
10:      Change the leader to the second-best one in $k$.
11:    end if
12:    while ($n_{jump} \leqslant N_{jump}$) and (no better position) do
13:      Update $PRT$ values
14:      The migrant moves to the leader
15:      Checking boundary
16:      Re-evaluate fitness function
17:      Updated this migrant
18:      Updated the global best position
19:    end while
20:    if the global best is not updated after $t$ $FEs$ then
21:      Randomly replace $x$% of the population $P$
22:    end if
23:  end for
24: end while
25: return
```

Fig. 8 visually illustrates how the operating optimization process of the three algorithms SOMA ATO, SHADE, and iSOMA, implemented on the Rotated Composition Function (F.26) of the CEC17. It clearly shows how the individuals of the classical SOMA move along the edges while SHADE's movement is spread evenly in the search space. The searching capability has been improved in the iSOMA version by applying the above-mentioned processes providing iSOMA's balanced power as evidenced by "spreading" individuals throughout the search space and then "focusing" toward the best individual.

4. Experimental setup

4.1. Test functions

To thoroughly evaluate the iSOMA performance, three common test suites of the IEEE Congress on Evolutionary Computation (IEEE CEC) were used, including a total of 73 functions as listed in Tables 1, 2, and 3 and presented below:

- The first benchmark set is the IEEE CEC 2013 Special Session on Real Parameter Single Objective Optimization, consisting of 28 functions (CEC13, see detail at [30]);
- The second is the IEEE CEC 2015 Competition on Learning-based Real Parameter Single Objective Optimization (CEC15, see detail at [31]);
- And the last one is the IEEE CEC 2017 Special Session and Competition on Single Objective Real Parameter Numerical Optimization (CEC17, see detail at [32]).

These single objective benchmark problems were used for evaluation because they are the basis of research on more complex optimization problems such as multi-objective, dynamic, niching composition, computationally expensive, and so on. They are categorized into various types of functions including unimodal, basic multimodal, simple multimodal, hybrid, and composition, (non-)separable, shifted, and rotated functions that are challenging enough to evaluate an algorithm. Definitions and details can be found in [30–32].

4.2. Comparison algorithms

To demonstrate improvement over previous versions of the iSOMA, the results were compared to the original and latest versions of SOMA, as listed below.

On the SOMA family:

- Self-organizing migrating algorithm AllToOne and AllToAll (SOMA ATO; SOMA ATA) [8,9];
- Pareto-based self-organizing migrating algorithm (SOMA Pareto) [22];
- Self-organizing migrating algorithm team to team adaptive (SOMA T3 A) [20].

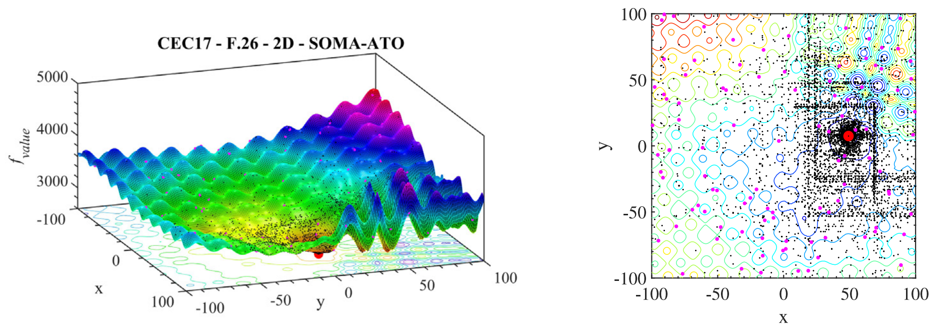
To investigate the iSOMA level of performance and effectiveness compared to some well-known existing algorithms, we carry out experiments on the various types of algorithms such as DE, PSO, and ABC shown below, including algorithms that have participated in the corresponding years' competitions. Compare iSOMA with other SOMAs to figure out the impact of the improvements we have proposed and compare with other algorithms outside the SOMAs to ascertain the position of iSOMA on the optimization algorithm map.

IEEE CEC 2013 (CEC13):

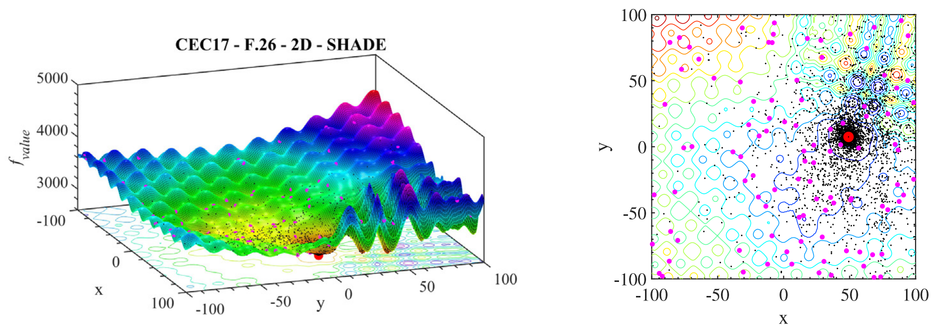
- Success-history based parameter adaptation for differential evolution (SHADE) [33];
- Super-fit multicriteria adaptive differential evolution (SMADE) [34];
- A CMA-ES super-fit scheme for the re-sampled inheritance search (CMAES-RIS) [35];
- A particle swarm optimization and artificial bee colony hybrid algorithm (SPSOABC) [36];
- A genetic algorithm for solving the CEC'2013 competition problems on real-parameter optimization (TPC-GA) [37].

IEEE CEC 2015 (CEC15):

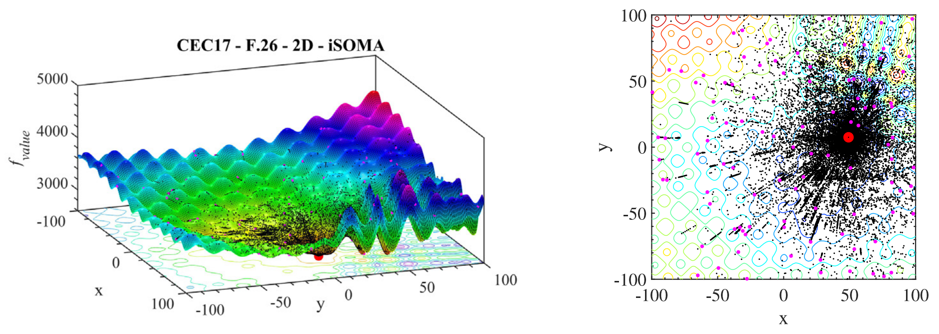
- A differential evolution algorithm with success-based parameter adaptation for CEC2015 learning-based optimization (DEsPA) [38];
- Tuning maturity model of ecogeography-based optimization on CEC 2015 single-objective optimization test problems (TEBO) [39];
- A Self-adaptive Dynamic Particle Swarm Optimizer (SaDPSO) [40];
- An improved covariance matrix leaning and searching preference algorithm for solving CEC 2015 benchmark problems (ICMLSP) [41];
- Dynamic search fireworks algorithm with covariance mutation for solving the CEC 2015 learning based competition problems (dynFWACM) [42].



(a) Distribution of individuals of SOMA ATO, presented in 3D and 2D.



(b) Distribution of individuals of SHADE, presented in 3D and 2D.



(c) Distribution of individuals of iSOMA, presented in 3D and 2D.

◎ Contour map • All positions obtained during the search ● Initial population ● Best positon

Fig. 8. The operation of three algorithms SOMA-ATO, SHADE, and iSOMA on the function 26*th* of CEC17 tested on 2D.

Table 1
The list of the IEEE CEC 2013 special session on real parameter single objective optimization functions.

No.	Functions	F^*	No.	Functions	F^*
1	Sphere function	−1400	15	Rotated Schwefel's function	100
2	Rotated high conditioned elliptic function	−1300	16	Rotated Katsuura function	200
3	Rotated Bent Cigar function	−1200	17	Lunacek Bi-Rastrigin function	300
4	Rotated discus function	−1100	18	Rotated Lunacek Bi-Rastrigin function	400
5	Different powers function	−1000	19	Expanded Griewank's plus Rosenbrock's function	500
6	Rotated Rosenbrock's function	−900	20	Expanded Scaffer's F6 function	600
7	Rotated Schaffers F7 function	−800	21	Composition Function 1 (n = 5,Rotated)	700
8	Rotated Ackley's function	−700	22	Composition Function 2 (n = 3,Unrotated)	800
9	Rotated Weierstrass function	−600	23	Composition Function 3 (n = 3,Rotated)	900
10	Rotated Griewank's function	−500	24	Composition Function 4 (n = 3,Rotated)	1000
11	Rastrigin's function	−400	25	Composition Function 5 (n = 3,Rotated)	1100
12	Rotated Rastrigin's function	−300	26	Composition Function 6 (n = 5,Rotated)	1200
13	Non-continuous rotated Rastrigin's function	−200	27	Composition Function 7 (n = 5,Rotated)	1300
14	Schwefel's function	−100	28	Composition Function 8 (n = 5,Rotated)	1400

IEEE CEC 2017 (CEC17):

- A differential evolution strategy (DES) [43];
- A version of IPOP-CMA-ES algorithm with midpoint for CEC 2017 single objective bound constrained problems (RB-IPOP-CMA-ES) [44];

Table 2
The list of the IEEE CEC 2015 competition on learning-based real parameter single objective optimization functions.

No.	Functions	F^*	No.	Functions	F^*
1	Rotated high conditioned elliptic function	100	9	Composition Function 1 (N = 3)	900
2	Rotated Cigar function	200	10	Composition Function 2 (N = 3)	1000
3	Shifted and rotated Ackley's function	300	11	Composition Function 3 (N = 5)	1100
4	Shifted and rotated Rastrigin's function	400	12	Composition Function 4 (N = 5)	1200
5	Shifted and rotated Schwefel's function	500	13	Composition Function 5 (N = 5)	1300
6	Hybrid Function 1 (N = 3)	600	14	Composition Function 6 (N = 7)	1400
7	Hybrid Function 2 (N = 4)	700	15	Composition Function 7 (N = 10)	1500
8	Hybrid Function 3 (N = 5)	800	–	–	–

Table 3
The list of the IEEE CEC 2017 special session and competition on single objective real parameter numerical optimization functions.

No.	Functions	F^*	No.	Functions	F^*
1	Shifted and rotated Bent Cigar function	100	16	Hybrid Function 6 (N = 4)	1600
2	Shifted and rotated sum of different power function	200	17	Hybrid Function 6 (N = 5)	1700
3	Shifted and rotated Zakharov function	300	18	Hybrid Function 6 (N = 5)	1800
4	Shifted and rotated Rosenbrock's function	400	19	Hybrid Function 6 (N = 5)	1900
5	Shifted and rotated Rastrigin's function	500	20	Hybrid Function 6 (N = 6)	2000
6	Shifted and rotated expanded Scaffer's F6 function	600	21	Composition Function 1 (N = 3)	2100
7	Shifted and rotated Lunacek Bi-Rastrigin function	700	22	Composition Function 2 (N = 3)	2200
8	Shifted and rotated non-continuous Rastrigin's function	800	23	Composition Function 3 (N = 4)	2300
9	Shifted and rotated Levy function	900	24	Composition Function 4 (N = 4)	2400
10	Shifted and rotated Schwefel's function	1000	25	Composition Function 5 (N = 5)	2500
11	Hybrid Function 1 (N = 3)	1100	26	Composition Function 6 (N = 5)	2600
12	Hybrid Function 2 (N = 3)	1200	27	Composition Function 7 (N = 6)	2700
13	Hybrid Function 3 (N = 3)	1300	28	Composition Function 8 (N = 6)	2800
14	Hybrid Function 4 (N = 4)	1400	29	Composition Function 9 (N = 3)	2900
15	Hybrid Function 5 (N = 4)	1500	30	Composition Function 10 (N = 3)	3000

- Proactive particles in swarm optimization: A settings-free algorithm for real-parameter single objective optimization problems (PPSO) [45];
- Dynamic Yin–Yang pair optimization and its performance on single objective real parameter problems of cec 2017 (DYYPO) [46];
- Teaching learning based optimization with focused learning and its performance on CEC2017 functions (TLBO-FL) [47].

4.3. Parameter settings

Tests on 10D and 30D are carried out, with a search range of $[-100, 100]^D$ for those testing problems. The *MaxFEs* was used at $10000 * D$ (*MaxFEs* for 10D = 100000; for 30D = 300000). Error value smaller than 10^{-8} will be taken as zero. Each algorithm was independently run 51 times for each function, as the experimental settings requested in [30–32]. To determine if the gaps between the findings are meaningful, the Wilcoxon rank-sum test (WRT) was used at the 5% significance [48,49].

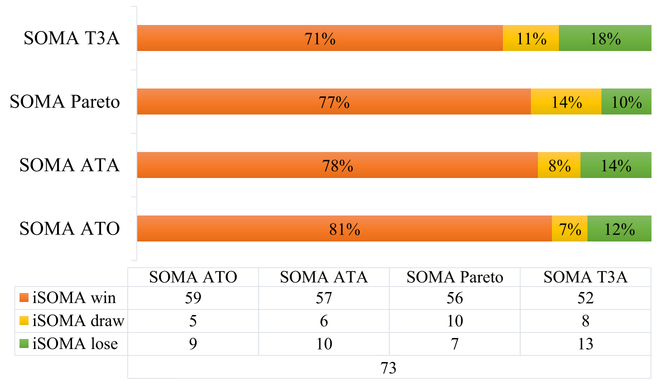
The control parameter of iSOMA: $PopSize = 100$, $N_{jump} = 10$, $n = 5$, $m = 10$, $k = 15$, $Step = 0.3$, and PRT as in Eq. (5). The control parameter values of the rest algorithms were used just as they were in the original articles in the citations, with no modifications.

The iSOMA is available at MathWorks site here.

5. Comparison results

The error values of 51 continuous runs are used as a basis for comparing the performance of algorithms with dimensions $D = 10$ and $D = 30$. It is obtained by the difference between the best value found by the algorithm and the global optimal value within the given search ranges of those functions ($F(x) - F(x^*)$). Note that the error value is considered zero when smaller than 10^{-8} as mentioned in the contest rules of [30–32].

The results of comparisons between the iSOMA algorithm with others were recorded in tables where each row represents the values of the mean and standard deviation of 51 runs for each testing function. The signs (+), (−), and (≈) show the comparison outcome at 5% of the WRT where it is significantly better (iSOMA loses), significantly worse (iSOMA wins), and not significantly better or worse (draw) compared to iSOMA [48,49]. Without statistical checks, the best outcomes for each row in the participating algorithms were bold. The last three rows in each table show the total of (+), (−), and (≈).



	SOMA ATO	SOMA ATA	SOMA Pareto	SOMA T3A
🟧 iSOMA win	59	57	56	52
🟨 iSOMA draw	5	6	10	8
🟩 iSOMA lose	9	10	7	13
	73			

Fig. 9. The summarized comparison results between the iSOMA and other SOMAs tested on 73 benchmark functions.

5.1. Outperform other SOMAs

Tables 4, 5, and 6 present the comparison results between some latest versions of the SOMA family, in turn, performed on the CEC13, CEC15, and CEC17 test suites within only 30D.

In particular, compared to SOMA ATO and ATA versions, iSOMA has significantly better results on 3 test suites with a total of 59 and 57 over 73 cases wins, while iSOMA only loses 9 and 10, draws 5 and 6, respectively, as summarized in Fig. 9. These results clearly show that the improvements of the iSOMA bring superior performance compared to the classical version as well as the SOMA Pareto and T3 A.

Table 4
Comparison of iSOMA with SOMA family on the CEC13 benchmark functions (30 dimensions, 51 runs).

F	iSOMA Mean (Std Dev)	SOMA ATO Mean (Std Dev)	SOMA ATA Mean (Std Dev)	SOMA Pareto Mean (Std Dev)	SOMA T3A Mean (Std Dev)
F_1	**0.00e+00 (0.00e+00)**	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈
F_2	9.80e+04 (4.13e+04)	1.70e+07 (4.03e+06)−	1.40e+07 (2.62e+06)−	**8.79e+04 (3.68e+04)**≈	3.48e+05 (2.01e+05)−
F_3	**3.13e+06 (4.17e+06)**	9.75e+07 (1.20e+08)−	1.89e+08 (1.60e+08)−	3.60e+07 (5.49e+07)−	2.06e+07 (3.02e+07)−
F_4	**3.23e+02 (1.65e+02)**	2.49e+04 (5.35e+03)−	2.08e+04 (4.85e+03)−	7.43e+02 (1.12e+03)≈	4.79e+02 (3.26e+02)−
F_5	**0.00e+00 (0.00e+00)**	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈
F_6	1.95e+01 (1.70e+01)	3.24e+01 (2.34e+01)−	3.28e+01 (2.01e+01)−	**1.71e+01 (1.95e+01)**+	3.13e+01 (2.46e+01)−
F_7	**1.37e+01 (4.39e+00)**	8.21e+01 (1.24e+01)−	8.66e+01 (1.55e+01)−	3.74e+01 (7.86e+00)−	3.48e+01 (9.38e+00)−
F_8	2.09e+01 (5.36e−02)	2.09e+01 (5.24e−02)≈	2.10e+01 (4.79e−02)≈	2.09e+01 (4.89e−02)≈	**2.09e+01 (4.62e−02)**≈
F_9	**2.10e+01 (3.43e+00)**	3.11e+01 (1.30e+00)−	2.82e+01 (2.60e+00)−	2.35e+01 (4.33e+00)−	2.85e+01 (2.33e+00)−
F_{10}	1.91e−01 (7.57e−02)	3.88e−01 (2.44e−01)−	3.03e−01 (1.21e−01)−	3.00e−01 (1.48e−01)−	**1.87e−01 (9.30e−02)**≈
F_{11}	7.33e+00 (2.10e+00)	7.02e−01 (8.74e−01)+	**2.93e−01 (5.73e−01)**+	1.55e+01 (4.86e+00)−	2.58e+00 (1.40e+00)+
F_{12}	**1.84e+01 (5.98e+00)**	1.65e+02 (1.87e+01)−	1.22e+02 (2.02e+01)−	3.57e+01 (8.45e+00)−	3.91e+01 (1.16e+01)−
F_{13}	**4.42e+01 (1.61e+01)**	1.80e+02 (1.41e+01)−	1.60e+02 (2.17e+01)−	8.06e+01 (2.32e+01)−	8.03e+01 (2.67e+01)−
F_{14}	1.00e+03 (3.27e+02)	1.19e+01 (6.64e+00)+	**4.00e+00 (3.26e+00)**+	1.26e+03 (4.01e+02)−	1.56e+01 (8.22e+00)+
F_{15}	**2.84e+03 (6.80e+02)**	5.51e+03 (3.16e+02)−	4.62e+03 (3.52e+02)−	3.34e+03 (6.66e+02)−	3.87e+03 (6.38e+02)−
F_{16}	2.44e+00 (2.58e−01)	2.13e+00 (2.29e−01)+	**1.73e+00 (3.15e−01)**+	1.87e+00 (3.60e−01)+	2.08e+00 (5.02e−01)+
F_{17}	4.31e+01 (4.42e+00)	3.14e+01 (6.21e−01)+	**3.07e+01 (2.44e−01)**+	5.22e+01 (5.37e+00)−	3.36e+01 (1.19e+00)+
F_{18}	**5.01e+01 (8.52e+00)**	2.12e+02 (1.43e+01)−	1.85e+02 (1.77e+01)−	5.25e+01 (8.29e+00)≈	6.37e+01 (1.27e+01)−
F_{19}	2.42e+00 (4.96e−01)	1.88e+00 (3.01e−01)+	**1.48e+00 (2.66e−01)**+	2.78e+00 (7.47e−01)−	1.97e+00 (3.23e−01)+
F_{20}	**9.18e+00 (6.49e−01)**	1.33e+01 (5.48e−01)−	1.34e+01 (5.34e−01)−	9.86e+00 (6.76e−01)−	1.05e+01 (7.97e−01)−
F_{21}	3.03e+02 (6.21e+01)	3.21e+02 (8.91e+01)−	**2.73e+02 (5.81e+01)**≈	3.28e+02 (7.87e+01)−	3.16e+02 (8.90e+01)−
F_{22}	**6.51e+02 (2.49e+02)**	1.42e+02 (5.39e+01)+	5.33e+01 (3.75e+01)+	1.30e+03 (4.06e+02)−	1.29e+02 (4.65e+01)+
F_{23}	**2.84e+03 (6.93e+02)**	6.27e+03 (3.41e+02)−	5.39e+03 (3.87e+02)−	3.48e+03 (6.40e+02)−	4.61e+03 (8.02e+02)−
F_{24}	**2.21e+02 (5.18e+00)**	2.76e+02 (9.04e+00)−	2.75e+02 (7.61e+00)−	2.27e+02 (4.44e+00)−	2.50e+02 (1.06e+01)−
F_{25}	**2.74e+02 (7.54e+00)**	3.05e+02 (3.74e+00)−	2.97e+02 (4.48e+00)−	2.78e+02 (8.97e+00)−	2.95e+02 (6.61e+00)−
F_{26}	2.00e+02 (3.09e−03)	2.01e+02 (3.01e−01)−	2.01e+02 (3.97e−01)−	**2.00e+02 (2.11e−03)**≈	2.00e+02 (7.75e−03)−
F_{27}	**5.54e+02 (5.88e+01)**	1.04e+03 (2.08e+02)−	9.13e+02 (2.70e+02)−	6.38e+02 (8.02e+01)−	1.01e+03 (9.22e+01)−
F_{28}	**3.00e+02 (0.00e+00)**	**3.00e+02 (0.00e+00)**−	**3.00e+02 (0.00e+00)**−	**3.00e+02 (0.00e+00)**≈	**3.00e+02 (0.00e+00)**−
	+	6	6	2	6
	−	19	18	18	18
	≈	3	4	8	4

Table 5
Comparison of iSOMA with SOMA family on the CEC15 benchmark functions (30 dimensions, 51 runs).

F	iSOMA Mean (Std Dev)	SOMA ATO Mean (Std Dev)	SOMA ATA Mean (Std Dev)	SOMA Pareto Mean (Std Dev)	SOMA T3A Mean (Std Dev)
F_1	**5.61e+03 (5.25e+03)**	2.00e+06 (7.21e+05)−	1.79e+06 (6.16e+05)−	1.11e+04 (8.30e+03)−	5.20e+04 (4.99e+04)−
F_2	1.14e−01 (4.52e−01)	3.19e+03 (2.99e+03)−	9.12e+02 (1.21e+03)−	3.14e−01 (1.10e+00)−	**1.93e−04 (9.21e−04)**+
F_3	2.10e+01 (4.44e−02)	**2.03e+01 (3.12e−02)**+	2.03e+01 (3.82e−02)+	2.08e+01 (9.95e−02)+	2.04e+01 (1.06e−01)+
F_4	**1.47e+01 (3.76e+00)**	6.80e+01 (7.81e+00)−	4.89e+01 (7.40e+00)−	2.95e+01 (6.53e+00)−	5.11e+01 (1.52e+01)−
F_5	**1.91e+03 (5.53e+02)**	2.78e+03 (2.58e+02)−	2.17e+03 (2.44e+02)−	2.59e+03 (5.71e+02)−	2.16e+03 (4.96e+02)−
F_6	**2.43e+03 (1.61e+03)**	1.19e+06 (6.92e+05)−	1.07e+06 (5.29e+05)−	5.47e+03 (5.55e+03)−	1.30e+04 (9.02e+03)−
F_7	**2.60e+00 (7.40e−01)**	9.85e+00 (1.46e+00)−	8.46e+00 (1.80e+00)−	4.00e+00 (9.00e−01)−	3.79e+00 (1.05e+00)−
F_8	**1.88e+03 (2.41e+03)**	2.70e+05 (1.28e+05)−	2.49e+05 (1.32e+05)−	5.71e+03 (5.02e+03)−	6.47e+03 (5.73e+03)−
F_9	**1.02e+02 (1.23e−01)**	1.03e+02 (2.13e−01)−	1.04e+02 (3.29e−01)−	1.03e+02 (1.61e−01)−	1.03e+02 (1.55e−01)−
F_{10}	**2.45e+03 (1.71e+03)**	3.89e+05 (2.05e+05)−	4.35e+05 (2.12e+05)−	4.55e+03 (4.50e+03)−	4.95e+03 (3.57e+03)−
F_{11}	3.10e+02 (3.25e+01)	3.21e+02 (9.10e+00)−	3.35e+02 (5.25e+01)−	3.14e+02 (5.70e+01)−	**3.03e+02 (1.99e+00)**+
F_{12}	**1.04e+02 (4.28e−01)**	1.07e+02 (5.68e−01)−	1.07e+02 (6.28e−01)−	1.04e+02 (4.13e−01)−	1.05e+02 (5.77e−01)−
F_{13}	**9.68e+01 (5.16e+00)**	1.04e+02 (2.67e+00)−	1.01e+02 (3.53e+00)−	1.03e+02 (5.72e+00)−	1.07e+02 (4.87e+00)−
F_{14}	3.26e+04 (5.55e+02)	**3.19e+04 (6.16e+02)**+	3.23e+04 (5.69e+02)+	3.27e+04 (4.54e+02)≈	3.21e+04 (7.25e+02)+
F_{15}	**1.00e+02 (1.22e−13)**	1.00e+02 (1.35e−13)−	1.00e+02 (1.09e−13)−	1.00e+02 (2.59e−13)−	1.00e+02 (5.50e−13)−
	+	2	2	1	4
	−	13	13	13	11
	≈	0	0	1	0

5.2. Compete against other algorithms

For CEC13:

Tables 7 and 8 show the comparison results on 10D and 30D with well-known DE versions of SHADE and SMADE, and other well-attended algorithms such as PSO, GA and ABC listed in the previous section and summarized in Fig. 10. For 10D problems, iSOMA proved weaker than the two versions of DE, when losing 16 and 13 out of 28 functions, winning only 4 and 9 functions. However, the situation changed for 30D problems when iSOMA won 14 and lost 9 compared to SMADE. These results show promising potential.

Compared to TPC-GA and CMAES-RIS, it is clear that iSOMA is on par with them on 10D, and outperforms on 30D problems. This shows that TPC-GA and CMAES-RIS are more effective on unimodal functions compared to iSOMA, as well as fast convergence but potentially be trapped in local optima of the complex functions. In contrast, iSOMA has proven its synergy on basic multimodal and composition functions.

For CEC15:

Tables 9 and 10, in turn, show the simulation results between iSOMA compared to DEsPA, TEBO, SaDPSO, ICMLSP and dynFWACM, on both 10D and 30D and summarized in Fig. 11. Confronted with another DE representative, iSOMA was a bit weaker to lose 8 out of 15 cases on both 10D and 30D, winning only 4 and 6 cases.

For TEBO, iSOMA has comparable optimization results on 30D problems and is somewhat weaker on 10D. The results were moderately better meanwhile antagonizing to SaDPSO, ICMLSP, and

Table 6
Comparison of iSOMA with SOMA family on the CEC17 benchmark functions (30 dimensions, 51 runs).

F	iSOMA Mean (Std Dev)	SOMA ATO Mean (Std Dev)	SOMA ATA Mean (Std Dev)	SOMA Pareto Mean (Std Dev)	SOMA T3A Mean (Std Dev)
F_1	6.63e−10 (3.55e−09)	1.48e+03 (2.41e+03)−	5.52e+02 (1.14e+03)−	2.49e−08 (1.48e−07)≈	**0.00e+00 (0.00e+00)**≈
F_2	**3.92e−02 (2.80e−01)**	3.61e+08 (2.47e+09)−	9.10e+04 (5.96e+05)−	1.67e+03 (9.51e+03)−	2.97e+09 (1.46e+10)−
F_3	9.71e−04 (2.82e−03)	1.54e+04 (4.40e+03)−	9.89e+03 (3.34e+03)−	**3.81e−05 (1.45e−04)**+	1.90e−02 (6.43e−02)−
F_4	4.52e+01 (3.20e+01)	8.46e+01 (2.78e+01)−	8.54e+01 (2.19e+01)−	**3.77e+00 (9.58e+00)**+	5.12e+01 (3.12e+01)≈
F_5	**1.41e+01 (4.02e+00)**	6.81e+01 (6.19e+00)−	4.99e+01 (9.35e+00)−	3.00e+01 (7.20e+00)−	5.20e+01 (1.70e+01)−
F_6	3.68e−06 (1.39e−05)	**0.00e+00 (0.00e+00)**+	**0.00e+00 (0.00e+00)**+	1.41e−03 (1.83e−03)−	4.22e−04 (5.76e−04)−
F_7	**4.35e+01 (3.73e+00)**	1.06e+02 (7.50e+00)−	8.27e+01 (8.37e+00)−	5.13e+01 (7.60e+00)−	7.77e+01 (1.48e+01)−
F_8	**1.56e+01 (4.00e+00)**	7.07e+01 (6.25e+00)−	5.46e+01 (8.48e+00)−	2.85e+01 (7.89e+00)−	5.65e+01 (1.48e+01)−
F_9	**3.63e−01 (5.77e−01)**	7.17e−01 (1.25e+00)−	3.05e+00 (4.67e+00)−	6.37e+00 (5.03e+00)−	4.14e+00 (4.32e+00)−
F_{10}	**2.13e+03 (4.94e+02)**	3.08e+03 (2.21e+02)−	2.35e+03 (2.98e+02)−	2.84e+03 (5.85e+02)−	2.51e+03 (4.67e+02)−
F_{11}	**1.26e+01 (1.53e+01)**	6.00e+01 (2.77e+01)−	1.75e+01 (1.32e+01)−	2.80e+01 (1.94e+01)−	2.35e+01 (2.15e+01)−
F_{12}	**7.00e+03 (4.23e+03)**	3.96e+05 (2.70e+05)−	5.09e+05 (3.26e+05)−	1.13e+04 (5.68e+03)−	1.04e+04 (5.79e+03)−
F_{13}	**2.61e+01 (1.44e+01)**	1.31e+04 (1.39e+04)−	8.30e+03 (7.59e+03)−	7.26e+01 (5.11e+01)−	1.63e+02 (2.24e+02)−
F_{14}	**4.35e+01 (1.64e+01)**	4.28e+04 (3.36e+04)−	8.75e+04 (1.14e+05)−	1.21e+02 (3.14e+02)−	6.86e+01 (7.42e+01)≈
F_{15}	**1.79e+02 (6.81e+02)**	7.45e+03 (7.70e+03)−	2.12e+03 (2.42e+03)−	1.49e+02 (3.22e+02)+	**2.52e+01 (1.77e+01)**+
F_{16}	**3.54e+02 (2.09e+02)**	7.83e+02 (1.27e+02)−	5.89e+02 (1.72e+02)−	6.93e+02 (2.43e+02)−	5.61e+02 (1.66e+02)−
F_{17}	**3.80e+01 (2.62e+01)**	2.34e+02 (7.51e+01)−	1.45e+02 (8.89e+01)−	8.78e+01 (8.76e+01)−	9.63e+01 (7.21e+01)−
F_{18}	**9.13e+03 (6.75e+03)**	2.09e+05 (1.09e+05)−	2.04e+05 (1.17e+05)−	1.15e+04 (6.72e+03)−	1.24e+04 (1.34e+04)≈
F_{19}	**1.46e+01 (6.05e+00)**	7.99e+03 (8.78e+03)−	2.92e+03 (3.60e+03)−	4.29e+01 (4.28e+01)−	1.91e+01 (9.23e+00)−
F_{20}	**1.28e+02 (4.87e+01)**	2.91e+02 (9.03e+01)−	1.85e+02 (8.71e+01)−	1.75e+02 (8.22e+01)−	1.57e+02 (8.33e+01)−
F_{21}	**2.17e+02 (4.66e+00)**	2.79e+02 (8.92e+00)−	2.50e+02 (2.86e+01)−	2.28e+02 (8.31e+00)−	2.45e+02 (4.41e+01)−
F_{22}	**1.00e+02 (3.44e−01)**	5.43e+02 (1.02e+03)−	6.45e+02 (1.08e+03)−	1.45e+02 (3.21e+02)−	3.85e+02 (8.73e+02)−
F_{23}	**3.65e+02 (8.16e+00)**	4.26e+02 (9.35e+00)−	4.04e+02 (1.05e+01)−	3.82e+02 (8.72e+00)−	4.01e+02 (1.70e+01)−
F_{24}	**4.37e+02 (6.12e+00)**	5.49e+02 (1.40e+01)−	5.12e+02 (4.29e+01)−	4.52e+02 (7.26e+00)−	4.74e+02 (1.79e+01)−
F_{25}	**3.87e+02 (3.12e−01)**	3.87e+02 (1.09e+00)≈	3.87e+02 (9.42e−01)≈	3.88e+02 (3.32e+00)−	3.88e+02 (1.11e+00)−
F_{26}	1.15e+03 (8.28e+01)	1.18e+03 (6.92e+02)≈	1.00e+03 (5.68e+02)≈	1.41e+03 (2.01e+02)−	**6.61e+02 (5.68e+02)**+
F_{27}	5.17e+02 (5.00e+00)	5.20e+02 (6.22e+00)−	**5.12e+02 (6.48e+00)**+	5.34e+02 (6.75e+00)−	5.12e+02 (6.49e+00)+
F_{28}	3.17e+02 (4.13e+01)	4.04e+02 (1.14e+01)−	4.02e+02 (4.98e+00)−	**3.04e+02 (2.05e+01)**+	3.23e+02 (4.25e+01)−
F_{29}	**4.61e+02 (3.84e+01)**	6.67e+02 (7.73e+01)−	5.29e+02 (7.50e+01)−	5.20e+02 (9.75e+01)−	5.63e+02 (9.71e+01)−
F_{30}	**2.82e+03 (6.75e+02)**	7.12e+03 (2.83e+03)−	4.60e+03 (9.44e+02)−	3.22e+03 (2.87e+02)−	4.34e+03 (2.00e+03)−
	+	1	2	4	3
	−	27	26	25	23
	≈	2	2	1	4

Table 7
Comparison of iSOMA with well-known algorithms on the CEC13 benchmark functions (10 dimensions, 51 runs).

F	iSOMA Mean (Std Dev)	SHADE Mean (Std Dev)	SMADE Mean (Std Dev)	CMAES-RIS Mean (Std Dev)	SPSOABC Mean (Std Dev)	TPC-GA Mean (Std Dev)
F_1	**0.00e+00 (0.00e+00)**	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈
F_2	1.67e+03 (1.92e+03)	**0.00e+00 (0.00e+00)**+	**0.00e+00 (0.00e+00)**+	**0.00e+00 (0.00e+00)**+	1.47e+05 (1.65e+05)−	**0.00e+00 (0.00e+00)**+
F_3	2.77e+05 (1.15e+06)	1.27e−01 (8.84e−01)+	2.48e−01 (1.24e+00)+	7.04e−01 (4.61e+00)+	1.27e+05 (6.22e+05)+	**0.00e+00 (0.00e+00)**+
F_4	1.83e+01 (4.14e+01)	**0.00e+00 (0.00e+00)**+	**0.00e+00 (0.00e+00)**+	**0.00e+00 (0.00e+00)**+	1.37e+03 (1.46e+03)−	**0.00e+00 (0.00e+00)**+
F_5	**0.00e+00 (0.00e+00)**	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈
F_6	4.04e+00 (4.88e+00)	7.89e+00 (3.93e+00)≈	5.41e+00 (4.81e+00)−	1.10e+00 (2.88e+00)≈	**0.00e+00 (0.00e+00)**+	**0.00e+00 (0.00e+00)**+
F_7	2.30e+00 (2.66e+00)	3.26e−03 (4.54e−03)+	2.27e+00 (4.50e+00)+	5.33e+01 (4.68e+01)−	**0.00e+00 (0.00e+00)**+	4.24e−02 (2.10e−01)+
F_8	2.03e+01 (7.12e−02)	2.04e+01 (8.95e−02)≈	2.03e+01 (1.04e−01)≈	2.03e+01 (1.37e−01)≈	**0.00e+00 (0.00e+00)**+	2.04e+01 (8.44e−02)−
F_9	2.73e+00 (8.60e−01)	3.39e+00 (7.35e−01)−	2.29e+00 (7.26e−01)+	3.59e+00 (1.04e+00)−	**0.00e+00 (0.00e+00)**+	3.39e+00 (2.88e+00)≈
F_{10}	3.33e−01 (2.05e−01)	1.20e−02 (8.99e−03)+	1.42e−02 (9.67e−03)+	1.24e−02 (1.35e−02)+	**0.00e+00 (0.00e+00)**+	3.87e−02 (2.83e−02)+
F_{11}	1.16e+00 (1.10e+00)	**0.00e+00 (0.00e+00)**+	9.75e−02 (2.99e−01)+	3.57e+00 (1.48e+00)−	**0.00e+00 (0.00e+00)**+	2.73e−01 (4.91e−01)+
F_{12}	4.97e+00 (2.24e+00)	3.14e+00 (9.73e−01)+	7.80e+00 (4.14e+00)−	1.29e+01 (5.42e+00)−	**0.00e+00 (0.00e+00)**+	6.03e+00 (2.18e+00)−
F_{13}	7.73e+00 (5.20e+00)	3.77e+00 (1.85e+00)+	1.21e+01 (6.47e+00)−	2.56e+01 (1.08e+01)−	**0.00e+00 (0.00e+00)**+	9.87e+00 (6.24e+00)≈
F_{14}	8.58e+01 (8.20e+01)	4.90e−03 (1.70e−02)+	3.64e+00 (4.44e+00)+	1.02e+02 (7.39e+01)≈	**0.00e+00 (0.00e+00)**+	2.45e+01 (2.47e+01)+
F_{15}	5.13e+02 (2.99e+02)	**4.21e+02 (1.14e+02)**≈	7.36e+02 (2.63e+02)−	6.17e+02 (1.74e+02)−	5.96e+02 (1.37e+02)−	7.34e+02 (2.44e+02)−
F_{16}	1.16e+00 (2.08e−01)	7.08e−01 (2.12e−01)+	4.04e−01 (3.17e−01)+	**1.64e−01 (7.56e−02)**+	2.00e+02 (1.25e−01)−	1.25e+00 (3.29e−01)−
F_{17}	1.31e+01 (1.69e+00)	**1.01e+01 (0.00e+00)**+	1.03e+01 (1.56e−01)+	1.04e+01 (3.73e+00)+	3.10e+02 (1.96e+00)−	1.12e+01 (7.76e−01)+
F_{18}	2.02e+01 (5.81e+00)	**1.69e+01 (1.54e+00)**+	2.46e+01 (4.73e+00)−	2.98e+01 (6.16e+00)−	4.17e+02 (1.95e+00)−	1.80e+01 (3.13e+00)≈
F_{19}	7.05e−01 (2.21e−01)	**3.44e−01 (4.90e−02)**+	3.95e−01 (1.26e−01)+	8.14e−01 (2.74e−01)≈	5.00e+02 (5.21e−02)−	5.01e−01 (1.21e−01)+
F_{20}	**2.02e+00 (5.98e−01)**	2.16e+00 (3.52e−01)≈	2.65e+00 (4.52e−01)−	4.16e+00 (3.99e−01)−	6.02e+02 (4.82e−01)−	3.17e+00 (4.81e−01)−
F_{21}	3.98e+02 (1.25e+01)	4.00e+02 (0.00e+00)−	3.83e+02 (5.56e+01)+	**1.61e+02 (6.03e+01)**+	1.10e+03 (2.80e+01)−	2.90e+02 (5.00e+01)+
F_{22}	9.20e+01 (8.29e+01)	**4.84e+00 (6.20e+00)**+	4.93e+01 (5.38e+01)+	2.44e+02 (1.09e+02)−	8.13e+02 (5.48e+00)−	9.07e+01 (6.14e+01)≈
F_{23}	**3.84e+02 (2.45e+02)**	4.61e+02 (1.78e+02)≈	5.78e+02 (3.20e+02)−	8.35e+02 (1.90e+02)−	1.50e+03 (1.81e+02)−	8.40e+02 (2.83e+02)−
F_{24}	1.45e+02 (4.32e+01)	1.93e+02 (2.46e+01)−	2.02e+02 (1.78e+01)−	**1.19e+02 (5.69e+00)**≈	1.20e+03 (2.33e+01)−	2.13e+02 (6.62e+00)−
F_{25}	2.01e+02 (1.02e+01)	2.00e+02 (7.02e−01)+	2.02e+02 (1.93e+00)≈	**1.93e+02 (3.42e+01)**+	1.30e+03 (2.16e+01)−	2.17e+02 (6.59e+00)−
F_{26}	**1.05e+02 (2.26e+00)**	1.33e+02 (4.36e+01)−	1.26e+02 (3.73e+01)−	1.61e+02 (4.06e+01)−	1.33e+03 (3.99e+01)−	1.96e+02 (1.81e+01)−
F_{27}	3.03e+02 (3.65e+00)	**3.00e+02 (1.46e−08)**+	3.37e+02 (5.29e+01)≈	3.13e+02 (2.30e+01)−	1.65e+03 (7.13e+01)−	4.24e+02 (6.83e+01)−
F_{28}	2.92e+02 (3.92e+01)	3.00e+02 (0.00e+00)≈	3.17e+02 (6.94e+01)≈	**2.06e+02 (1.07e+02)**+	1.69e+03 (7.01e+01)−	2.92e+02 (3.92e+01)≈
	+	16	13	9	10	11
	−	4	9	12	16	10
	≈	8	6	7	2	7

Table 8
Comparison of iSOMA with well-known algorithms on the CEC13 benchmark functions (30 dimensions, 51 runs).

F	iSOMA Mean (Std Dev)	SHADE Mean (Std Dev)	SMADE Mean (Std Dev)	CMAES-RIS Mean (Std Dev)	SPSOABC Mean (Std Dev)	TPC-GA Mean (Std Dev)
F_1	**0.00e+00 (0.00e+00)**	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈
F_2	9.80e+04 (4.13e+04)	9.00e+03 (7.47e+03)+	**0.00e+00 (0.00e+00)**+	**0.00e+00 (0.00e+00)**+	8.77e+05 (1.69e+06)−	2.44e+05 (1.60e+05)−
F_3	3.13e+06 (4.17e+06)	**4.02e+01 (2.13e+02)**+	9.82e+03 (4.99e+04)+	2.24e+03 (1.10e+04)+	5.16e+07 (8.00e+07)−	3.80e+07 (7.22e+07)−
F_4	3.23e+02 (1.65e+02)	1.92e−04 (3.01e−04)+	**0.00e+00 (0.00e+00)**+	**0.00e+00 (0.00e+00)**+	4.92e+03 (2.30e+03)−	1.38e+01 (2.17e+01)+
F_5	**0.00e+00 (0.00e+00)**	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈
F_6	1.95e+01 (1.70e+01)	5.96e−01 (3.73e+00)+	2.67e+00 (7.92e+00)+	6.94e−04 (2.01e−03)+	**0.00e+00 (0.00e+00)**+	2.43e+01 (1.26e+01)−
F_7	1.37e+01 (4.39e+00)	4.60e+00 (5.39e+00)+	3.25e+01 (1.63e+01)−	4.48e+01 (2.96e+01)−	**0.00e+00 (0.00e+00)**+	2.91e+01 (2.20e+01)−
F_8	2.09e+01 (5.36e−02)	2.07e+01 (1.76e−01)+	2.10e+01 (4.85e−02)≈	2.09e+01 (8.19e−02)+	**0.00e+00 (0.00e+00)**+	2.10e+01 (5.44e−02)−
F_9	2.10e+01 (3.43e+00)	2.75e+01 (1.77e+00)−	2.23e+01 (3.61e+00)≈	2.37e+01 (1.95e+00)−	**0.00e+00 (0.00e+00)**+	3.61e+01 (8.55e+00)−
F_{10}	1.91e−01 (7.57e−02)	7.69e−02 (3.58e−02)+	1.84e−02 (1.35e−02)+	8.31e−03 (5.46e−03)+	**0.00e+00 (0.00e+00)**+	8.68e−02 (4.78e−02)+
F_{11}	7.33e+00 (2.10e+00)	**0.00e+00 (0.00e+00)**+	1.09e+01 (4.23e+00)−	2.54e+01 (6.36e+00)−	**0.00e+00 (0.00e+00)**+	2.39e+01 (8.18e+00)−
F_{12}	1.84e+01 (5.98e+00)	2.30e+01 (3.73e+00)−	5.72e+01 (1.72e+01)−	7.94e+01 (4.39e+01)−	**0.00e+00 (0.00e+00)**+	4.14e+01 (8.94e+00)−
F_{13}	4.42e+01 (1.61e+01)	5.03e+01 (1.34e+01)≈	1.28e+02 (3.53e+01)−	1.56e+02 (5.42e+01)−	**0.00e+00 (0.00e+00)**+	8.41e+01 (2.06e+01)−
F_{14}	1.00e+03 (3.27e+02)	3.18e−02 (2.33e−02)+	1.33e+02 (1.28e+02)+	7.92e+02 (2.21e+02)+	**0.00e+00 (0.00e+00)**+	9.25e+02 (3.94e+02)≈
F_{15}	**2.84e+03 (6.80e+02)**	3.22e+03 (2.64e+02)−	4.10e+03 (8.55e+02)−	3.13e+03 (4.57e+02)−	3.65e+03 (3.04e+02)−	3.97e+03 (6.25e+02)−
F_{16}	2.44e+00 (2.58e−01)	9.13e−01 (1.85e−01)+	1.31e−01 (7.65e−02)+	**1.07e−01 (6.78e−02)**+	2.01e+02 (2.01e−01)−	2.50e+00 (5.94e−01)−
F_{17}	4.31e+01 (4.42e+00)	**3.04e+01 (0.00e+00)**+	3.48e+01 (1.54e+00)+	5.50e+01 (5.24e+00)−	3.31e+02 (1.23e−01)−	5.44e+01 (1.05e+01)−
F_{18}	**5.01e+01 (8.52e+00)**	7.25e+01 (5.58e+00)−	8.33e+01 (2.08e+01)−	1.89e+02 (2.73e+01)−	4.90e+02 (8.95e+00)−	6.96e+01 (1.33e+01)−
F_{19}	2.42e+00 (4.96e−01)	**1.36e+00 (1.20e−01)**+	2.55e+00 (5.23e−01)≈	2.80e+00 (6.42e−01)−	5.02e+02 (4.68e−01)−	3.28e+00 (1.29e+00)−
F_{20}	**9.18e+00 (6.49e−01)**	1.05e+01 (6.04e−01)−	1.05e+01 (8.15e−01)−	1.43e+01 (5.75e−01)−	6.11e+02 (7.60e−01)−	1.37e+01 (4.54e−01)−
F_{21}	3.03e+02 (6.21e+01)	3.09e+02 (5.65e+01)−	3.27e+02 (8.73e+01)−	**1.86e+02 (4.01e+01)**+	1.02e+03 (7.53e+01)−	2.92e+02 (8.74e+01)+
F_{22}	6.51e+02 (2.49e+02)	**9.81e+01 (2.52e+01)**+	1.79e+02 (4.54e+01)+	1.17e+03 (2.93e+02)−	8.84e+02 (3.90e+01)−	1.27e+03 (5.56e+02)−
F_{23}	**2.84e+03 (6.93e+02)**	3.51e+03 (4.11e+02)−	4.22e+03 (8.83e+02)−	4.03e+03 (5.43e+02)−	5.08e+03 (5.62e+02)−	4.33e+03 (8.56e+02)−
F_{24}	2.21e+02 (5.18e+00)	**2.05e+02 (5.29e+00)**+	2.32e+02 (2.60e+01)−	2.59e+02 (1.76e+01)−	1.25e+03 (1.43e+01)−	2.74e+02 (1.68e+01)−
F_{25}	2.74e+02 (7.54e+00)	**2.59e+02 (1.96e+01)**+	2.78e+02 (1.00e+01)−	2.82e+02 (8.50e+00)−	1.38e+03 (9.76e+00)−	2.98e+02 (9.14e+00)−
F_{26}	2.00e+02 (3.09e−03)	2.02e+02 (1.48e+01)−	2.15e+02 (5.30e+01)−	**1.97e+02 (1.21e+01)**≈	1.46e+03 (7.62e+01)−	3.25e+02 (5.96e+01)−
F_{27}	5.54e+02 (5.88e+01)	**3.88e+02 (1.09e+02)**+	6.47e+02 (1.39e+02)−	7.49e+02 (1.87e+02)−	2.21e+03 (1.62e+02)−	1.03e+03 (1.92e+02)−
F_{28}	**3.00e+02 (0.00e+00)**	**3.00e+02 (0.00e+00)**≈	3.88e+02 (3.27e+02)−	5.39e+02 (1.33e+03)−	1.73e+03 (2.32e+02)−	**3.00e+02 (0.00e+00)**≈
	+	16	9	9	9	3
	−	8	14	16	17	21
	≈	4	5	3	2	4

Table 9
Comparison of iSOMA with well-known algorithms on the CEC15 benchmark functions (10 dimensions, 51 runs).

F	iSOMA Mean (Std Dev)	DEsPA Mean (Std Dev)	TEBO Mean (Std Dev)	SaDPSO Mean (Std Dev)	ICMLSP Mean (Std Dev)	dynFWACM Mean (Std Dev)
F_1	4.69e+00 (1.85e+01)	**0.00e+00 (0.00e+00)**+	**0.00e+00 (0.00e+00)**+	3.61e+01 (1.40e+02)−	**0.00e+00 (0.00e+00)**+	1.11e+05 (9.88e+04)−
F_2	**0.00e+00 (0.00e+00)**	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	1.36e+02 (3.19e+02)−	**0.00e+00 (0.00e+00)**≈	8.86e+03 (8.90e+03)−
F_3	1.83e+01 (6.10e+00)	**1.67e+01 (7.24e+00)**+	1.73e+01 (6.95e+00)+	2.00e+01 (1.32e−02)−	2.00e+01 (5.30e−05)−	2.00e+01 (2.33e−04)−
F_4	**3.49e+00 (1.89e+00)**	3.56e+00 (1.30e+00)≈	5.07e+00 (2.36e+00)−	4.49e+00 (1.84e+00)−	3.87e+01 (1.89e+01)−	1.67e+01 (6.65e+00)−
F_5	1.78e+02 (1.26e+02)	**5.19e+01 (6.01e+01)**+	9.82e+01 (1.24e+02)+	1.31e+02 (8.54e+01)≈	1.03e+03 (3.10e+02)−	5.18e+02 (2.40e+02)−
F_6	1.37e+01 (3.22e+01)	**1.59e+00 (1.21e+00)**+	1.88e+01 (3.97e+01)≈	3.00e+02 (2.23e+02)−	5.24e+02 (2.63e+02)−	1.74e+03 (1.97e+03)−
F_7	2.31e−01 (3.40e−01)	3.42e−01 (2.85e−01)−	**1.40e−01 (2.67e−01)**+	6.63e−01 (3.75e−01)−	2.90e+00 (1.09e+00)−	1.45e+00 (2.31e−01)−
F_8	1.24e+01 (1.42e+01)	**1.95e−01 (2.22e−01)**+	5.68e+00 (1.44e+01)+	5.61e+01 (6.03e+01)−	3.44e+02 (1.89e+02)−	1.96e+03 (2.13e+03)−
F_9	**1.00e+02 (3.22e−02)**	1.06e+02 (1.35e+01)−	1.00e+02 (3.05e−01)−	1.00e+02 (3.72e−02)−	1.02e+02 (1.46e+00)−	1.00e+02 (4.59e−01)−
F_{10}	2.87e+02 (8.92e+01)	**7.56e+00 (3.96e−01)**+	1.61e+02 (3.89e+01)+	3.80e+02 (1.68e+02)−	5.25e+04 (2.60e+05)−	5.47e+02 (2.37e+02)−
F_{11}	**6.28e+01 (1.19e+02)**	9.46e+01 (1.00e+02)≈	1.42e+02 (1.50e+02)≈	1.54e+02 (1.48e+02)−	3.59e+02 (1.83e+02)−	1.85e+02 (1.47e+02)−
F_{12}	**1.01e+02 (2.00e−01)**	1.01e+02 (2.84e−01)−	1.01e+02 (3.16e−01)−	6.17e+01 (5.00e+01)≈	1.19e+02 (1.71e+01)−	1.13e+02 (1.52e+00)−
F_{13}	3.04e+01 (2.95e+00)	1.77e+01 (2.88e+00)+	2.86e+01 (2.98e+00)+	2.79e+01 (5.97e+00)+	2.26e−01 (1.51e−01)+	**1.21e−01 (1.66e−02)**+
F_{14}	3.57e+03 (1.59e+03)	**3.09e+02 (6.95e+02)**+	2.92e+03 (2.18e+03)≈	9.01e+02 (1.13e+03)+	7.23e+03 (1.25e+03)−	6.30e+03 (2.12e+03)−
F_{15}	1.00e+02 (1.42e−13)	2.05e+02 (1.35e−03)−	**1.00e+02 (0.00e+00)**+	1.00e+02 (1.46e−13)≈	**1.00e+02 (0.00e+00)**+	**1.00e+02 (0.00e+00)**+
	+	8	8	2	3	2
	−	4	3	10	11	13
	≈	3	4	3	1	0

dynFWACM when iSOMA had won more than half the number of winning cases compared to the number of losing cases.

For CEC17:

Compared to DES and RB-IPOP-CMA-ES, iSOMA prevailed on 10*D* problems, winning 17 and 13 cases over 30 functions, losing 9 and 10 cases, respectively. However, the situation was reversed when iSOMA lost its position on 30*D*. DES then asserts the balance control over all three unimodal, basic multimodal, and composition function classes.

As for the PPSO, DYYPO, and TLBO-FL algorithms, iSOMA completely dominated and almost absolutely won over them on 10 and 30 dimensions, as shown in Tables 11 and 12 and summarized in Fig. 12.

These results clearly show that iSOMA has good performance against some representatives of the GA and PSO algorithms, but is a little inferior to well-known DE versions due to SOMA and DE belonging to two different algorithm classes.

It is worth noting that the algorithms involved in the comparison have been carefully refined to participate in the corresponding years of CEC competitions. To be fair, iSOMA should use its own set of control parameters for each CEC test suite. That will greatly increase the comparison results, which is more beneficial for iSOMA. However, we use the same setting parameters for all CEC competitions. This gives iSOMA users a broad view of the iSOMA performance level.

6. Application to drones

In this subsection, the application of iSOMA for drones to hit the target and avoid the detected obstacles on its path is

Table 10
Comparison of iSOMA with well-known algorithms on the CEC15 benchmark functions (30 dimensions, 51 runs).

F	iSOMA Mean (Std Dev)	DEsPA Mean (Std Dev)	TEBO Mean (Std Dev)	SaDPSO Mean (Std Dev)	ICMLSP Mean (Std Dev)	dynFWACM Mean (Std Dev)
F_1	5.61e+03 (5.25e+03)	**0.00e+00 (0.00e+00)**+	3.69e+02 (7.71e+02)+	1.93e−02 (8.42e−02)+	0.00e+00 (0.00e+00)+	6.17e+05 (2.49e+05)−
F_2	1.14e−01 (4.52e−01)	**0.00e+00 (0.00e+00)**+	4.54e−07 (3.02e−06)+	2.88e+02 (1.09e+03)−	4.05e−05 (1.44e−04)+	3.31e+03 (3.59e+03)−
F_3	2.10e+01 (4.44e−02)	2.01e+01 (4.36e−02)+	2.00e+01 (6.32e−02)+	2.00e+01 (4.15e−05)+	2.00e+01 (7.57e−03)+	**2.00e+01 (5.75e−06)**+
F_4	**1.47e+01 (3.76e+00)**	8.64e+01 (2.87e−14)−	4.41e+01 (1.06e+01)−	4.25e+01 (9.31e+00)−	2.31e+02 (5.66e+01)−	1.30e+02 (3.80e+01)−
F_5	1.91e+03 (5.53e+02)	**1.85e+03 (3.97e+02)**≈	1.96e+03 (6.32e+02)≈	2.52e+03 (3.58e+02)−	4.03e+03 (6.56e+02)−	3.38e+03 (6.98e+02)−
F_6	2.43e+03 (1.61e+03)	**1.61e+02 (8.00e+01)**+	6.98e+02 (6.52e+02)+	1.38e+03 (6.04e+02)+	1.47e+03 (4.08e+02)+	2.69e+04 (1.90e+04)−
F_7	**2.60e+00 (7.40e−01)**	3.09e+00 (7.41e−01)−	4.42e+00 (1.41e+00)−	9.52e+00 (1.93e+00)−	2.07e+01 (1.45e+01)−	1.46e+01 (2.57e+00)−
F_8	1.88e+03 (2.41e+03)	**2.55e+01 (2.29e+01)**+	1.21e+02 (1.48e+02)+	1.62e+03 (1.35e+03)≈	9.42e+02 (2.50e+02)≈	2.40e+04 (1.32e+04)−
F_9	**1.02e+02 (1.23e−01)**	1.80e+02 (3.60e+01)−	1.08e+02 (1.22e+00)−	1.03e+02 (1.86e−01)−	1.63e+02 (1.32e+02)−	1.08e+02 (9.01e−01)−
F_{10}	2.45e+03 (1.71e+03)	**1.71e+02 (7.08e+01)**+	6.21e+02 (9.31e+01)+	6.52e+03 (4.66e+03)−	1.43e+03 (3.36e+02)+	3.15e+04 (2.01e+04)−
F_{11}	**3.10e+02 (3.25e+01)**	3.11e+02 (5.52e+01)−	4.81e+02 (1.95e+02)−	3.20e+02 (8.88e+00)−	1.12e+03 (2.72e+02)−	6.72e+02 (1.54e+02)−
F_{12}	**1.04e+02 (4.28e−01)**	1.08e+02 (3.19e−01)−	1.06e+02 (1.04e+00)−	1.05e+02 (4.90e−01)−	1.61e+02 (4.11e+01)−	1.17e+02 (1.23e+01)−
F_{13}	9.68e+01 (5.16e+00)	8.13e+01 (5.60e+00)+	9.87e+01 (5.46e+00)≈	1.01e+02 (4.06e+00)−	8.53e−02 (1.26e−01)+	**2.62e−02 (7.46e−03)**+
F_{14}	3.26e+04 (5.55e+02)	2.81e+04 (1.71e+03)+	3.45e+04 (4.04e+03)−	**1.87e+04 (5.27e+03)**+	4.21e+04 (4.67e+03)−	4.49e+04 (1.02e+03)−
F_{15}	1.00e+02 (1.22e−13)	2.73e+02 (1.50e−01)−	**1.00e+02 (0.00e+00)**+	1.00e+02 (1.19e−13)−	1.27e+02 (1.62e+01)−	**1.00e+02 (0.00e+00)**+
	+	8	7	4	6	3
	–	6	6	10	8	12
	≈	1	2	1	1	0

Table 11
Comparison of iSOMA with well-known algorithms on the CEC17 benchmark functions (10 dimensions, 51 runs).

F	iSOMA Mean (Std Dev)	DES Mean (Std Dev)	RB-IPOP-CMA-ES Mean (Std Dev)	PPSO Mean (Std Dev)	DYYPO Mean (Std Dev)	TLBO-FL Mean (Std Dev)
F_1	**0.00e+00 (0.00e+00)**	1.20e−07 (3.04e−08)−	2.93e−10 (2.09e−09)≈	2.39e+02 (2.01e+02)−	2.86e+03 (3.27e+03)−	2.02e+03 (2.46e+03)−
F_2	**0.00e+00 (0.00e+00)**	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	1.96e−02 (1.40e−01)≈
F_3	**0.00e+00 (0.00e+00)**	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	1.17e−05 (4.30e−05)−	1.11e−04 (7.84e−04)−
F_4	2.80e−05 (1.68e−04)	**0.00e+00 (0.00e+00)**+	**0.00e+00 (0.00e+00)**+	1.20e+00 (9.36e−01)−	2.07e+00 (8.25e+00)−	3.03e+00 (1.17e+00)−
F_5	3.51e+00 (1.61e+00)	**1.54e+00 (9.40e−01)**+	1.58e+00 (1.96e+00)+	1.81e+01 (5.10e+00)−	1.12e+01 (4.23e+00)−	8.75e+00 (5.57e+00)−
F_6	**0.00e+00 (0.00e+00)**	1.17e−01 (3.48e−01)−	2.01e−07 (6.21e−07)−	2.26e−01 (3.08e−01)−	6.36e−05 (6.02e−05)−	8.39e−08 (4.43e−07)≈
F_7	1.41e+01 (1.91e+00)	1.19e+01 (7.04e−01)+	**1.01e+01 (2.69e+00)**+	1.69e+01 (2.21e+00)−	2.18e+01 (6.02e+00)−	2.76e+01 (3.97e+00)−
F_8	3.59e+00 (1.58e+00)	**1.56e+00 (1.04e+00)**+	1.97e+00 (2.32e+00)+	9.95e+00 (2.37e+00)−	1.32e+01 (4.50e+00)−	1.23e+01 (4.40e+00)−
F_9	**0.00e+00 (0.00e+00)**	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	1.96e−02 (9.82e−02)−	8.91e−03 (6.36e−02)≈
F_{10}	1.88e+02 (1.78e+02)	**5.66e+00 (1.71e+01)**+	4.35e+02 (1.90e+02)−	5.03e+02 (1.55e+02)−	3.67e+02 (1.69e+02)−	9.55e+02 (2.14e+02)−
F_{11}	1.32e+00 (1.46e+00)	**1.17e−01 (3.24e−01)**+	1.71e−01 (4.10e−01)+	1.69e+01 (5.31e+00)−	9.28e+00 (4.79e+00)−	4.12e+00 (1.46e+00)−
F_{12}	**8.35e+01 (6.28e+01)**	4.41e+02 (1.94e+02)−	1.10e+02 (9.37e+01)−	4.55e+03 (2.51e+03)−	1.35e+04 (1.20e+04)−	6.56e+04 (5.49e+04)−
F_{13}	5.28e+00 (3.25e+00)	**3.31e+00 (3.00e+00)**+	4.17e+00 (3.61e+00)+	1.39e+03 (1.33e+03)−	5.08e+03 (5.64e+03)−	2.45e+03 (2.16e+03)−
F_{14}	**1.41e+00 (9.69e−01)**	1.23e+01 (1.02e+01)−	1.59e+01 (1.25e+01)−	3.73e+01 (1.17e+01)−	2.07e+01 (2.20e+01)−	6.73e+01 (1.83e+01)−
F_{15}	9.23e−01 (7.73e−01)	3.25e+00 (4.09e+00)−	**4.91e−01 (4.76e−01)**+	5.33e+01 (2.27e+01)−	4.36e+01 (1.11e+02)−	1.26e+02 (4.34e+01)−
F_{16}	**6.14e−01 (1.61e−01)**	6.09e+00 (2.34e+01)−	9.71e+01 (1.03e+02)−	8.30e+01 (7.25e+01)−	4.38e+01 (5.79e+01)−	8.91e+00 (2.19e+01)−
F_{17}	**5.06e+00 (6.60e+00)**	2.10e+01 (1.03e+01)−	5.25e+01 (3.36e+01)−	2.46e+01 (7.43e+00)−	1.41e+01 (1.39e+01)−	3.83e+01 (7.83e+00)−
F_{18}	**9.30e−01 (6.57e−01)**	2.91e+01 (2.46e+01)−	1.97e+01 (2.35e+01)−	8.78e+02 (7.14e+02)−	8.76e+03 (6.42e+03)−	6.15e+03 (5.63e+03)−
F_{19}	**3.09e−01 (4.74e−01)**	2.52e+00 (2.34e+00)−	1.82e+00 (3.30e+00)−	2.25e+01 (1.56e+01)−	9.27e+01 (2.94e+02)−	6.06e+01 (3.18e+01)−
F_{20}	**1.38e+00 (3.24e+00)**	1.22e+01 (9.86e+00)−	1.06e+02 (6.95e+01)−	2.78e+01 (8.96e+00)−	8.01e+00 (9.07e+00)−	1.46e+01 (9.45e+00)−
F_{21}	1.09e+02 (2.87e+01)	2.02e+02 (4.62e+00)−	1.37e+02 (4.92e+01)−	1.04e+02 (2.15e+01)+	**1.00e+02 (9.03e−01)**+	1.42e+02 (5.17e+01)−
F_{22}	**8.52e+01 (3.16e+01)**	1.00e+02 (1.50e−08)≈	9.93e+01 (5.57e+00)≈	9.67e+01 (1.68e+01)−	9.75e+01 (1.99e+01)−	9.33e+01 (2.27e+01)≈
F_{23}	2.99e+02 (3.86e+01)	3.01e+02 (1.98e+00)−	**2.75e+02 (7.06e+01)**+	3.42e+02 (1.05e+01)−	3.09e+02 (4.46e+01)−	3.07e+02 (3.84e+00)−
F_{24}	1.71e+02 (1.07e+02)	3.03e+02 (6.11e+01)−	1.98e+02 (1.02e+02)−	2.27e+02 (1.35e+02)−	**1.17e+02 (4.97e+01)**+	3.10e+02 (6.90e+01)−
F_{25}	4.29e+02 (2.20e+01)	4.08e+02 (1.89e+01)+	**4.02e+02 (6.54e+01)**+	4.04e+02 (1.45e+01)+	4.23e+02 (2.33e+01)≈	4.26e+02 (2.25e+01)+
F_{26}	2.88e+02 (5.86e+01)	2.96e+02 (1.96e+01)−	2.73e+02 (1.51e+02)+	**2.67e+02 (7.66e+01)**+	3.03e+02 (3.14e+01)−	3.01e+02 (4.63e+01)≈
F_{27}	3.94e+02 (2.67e+00)	3.96e+02 (2.33e+00)−	3.95e+02 (1.09e+00)≈	4.27e+02 (1.35e+01)−	3.96e+02 (3.76e+00)−	**3.93e+02 (3.28e+00)**+
F_{28}	3.00e+02 (2.99e−13)	5.26e+02 (1.23e+02)−	4.02e+02 (1.64e+02)−	**2.94e+02 (4.20e+01)**+	3.01e+02 (5.11e+01)−	4.47e+02 (1.58e+02)−
F_{29}	2.50e+02 (7.72e+00)	**2.36e+02 (7.67e+00)**+	2.66e+02 (4.53e+01)≈	2.78e+02 (1.35e+01)−	2.60e+02 (1.89e+01)−	2.74e+02 (1.38e+01)−
F_{30}	**5.01e+02 (8.96e+01)**	1.54e+05 (3.69e+05)−	2.05e+03 (1.04e+04)−	2.99e+03 (8.99e+02)−	6.70e+03 (6.52e+03)−	2.79e+05 (4.92e+05)−
	+	9	10	4	2	2
	–	17	13	23	26	23
	≈	4	7	3	2	5

proposed. Specifically, how to turn the path planning problem into an optimization problem will be presented, based on the principle that the closer the target, the lower the fitness value, and the closer the obstacle, the higher the fitness value. The iSOMA task in this problem is to point out optimal solutions in real-time for that built function, as shown in Fig. 13 and depicted in [50]. The following subsections will clarify the problem.

6.1. Problem formulation

Fig. 14 depicts the working model within a space consisting of 3 drones with 3 respective targets and 4 static obstacles. The main purpose of drones is to move toward their targets and avoid any detected obstacles on the way. To handle the task, the following necessary assumptions are proposed:

- *Drone*: In the framework of this study, drones are assumed to be able to fly smoothly from a given point to another nearby without any problems, named r_{step}. They are different depending on each drone, and it should be noted that the iSOMA performance is not affected by this parameter. Moreover, drones are assumed to be equipped with a detecting obstacles sensor system with a working range of $r_{detector}$.
- *Obstacles*: Obstacles are assumed that their entire physical size is surrounded by spheres with radius $r_{obstacles}$. These

Table 12
Comparison of iSOMA with well-known algorithms on the CEC17 benchmark functions (30 dimensions, 51 runs).

F	iSOMA Mean (Std Dev)	DES Mean (Std Dev)	RB-IPOP-CMA-ES Mean (Std Dev)	PPSO Mean (Std Dev)	DYYPO Mean (Std Dev)	TLBO-FL Mean (Std Dev)
F_1	**6.63e−10 (3.55e−09)**	5.37e−07 (7.42e−08)−	3.15e−08 (2.75e−08)−	7.48e+02 (6.06e+02)−	3.71e+03 (5.04e+03)−	3.51e+03 (3.61e+03)−
F_2	3.92e−02 (2.80e−01)	5.88e−02 (2.38e−01)≈	**0.00e+00 (0.00e+00)**≈	5.32e+01 (6.91e+01)−	3.18e+09 (2.27e+10)−	8.52e+16 (5.81e+17)−
F_3	9.71e−04 (2.82e−03)	2.02e−09 (4.44e−09)+	**0.00e+00 (0.00e+00)**+	1.13e+00 (4.83e−01)−	5.27e+02 (3.76e+03)≈	2.99e+03 (1.08e+03)−
F_4	4.52e+01 (3.20e+01)	5.69e+01 (1.17e+01)≈	5.53e+01 (1.65e+01)≈	**4.39e+01 (3.19e+01)**≈	9.12e+01 (2.49e+01)−	9.01e+01 (2.37e+01)−
F_5	1.41e+01 (4.02e+00)	4.64e+00 (1.42e+00)+	**1.65e+00 (1.37e+00)**+	1.12e+02 (1.33e+01)−	9.09e+01 (2.43e+01)−	3.95e+01 (2.07e+01)−
F_6	3.68e−06 (1.39e−05)	**4.50e−07 (1.05e−07)**+	1.21e−07 (3.97e−08)≈	2.03e+01 (4.15e+00)−	8.59e−01 (7.17e−01)−	4.87e−01 (4.24e−01)−
F_7	4.35e+01 (3.73e+00)	3.57e+01 (1.32e+00)+	**3.43e+01 (1.28e+00)**+	1.35e+02 (1.63e+01)−	1.44e+02 (3.08e+01)−	1.39e+02 (4.75e+01)−
F_8	1.56e+01 (4.00e+00)	4.55e+00 (1.66e+00)+	**1.76e+00 (1.65e+00)**+	8.10e+01 (1.04e+01)−	9.63e+01 (2.45e+01)−	3.67e+01 (1.84e+01)−
F_9	3.63e−01 (5.77e−01)	2.28e−09 (4.69e−09)+	**0.00e+00 (0.00e+00)**+	1.36e+03 (2.82e+02)−	6.54e+02 (7.73e+02)−	3.45e+01 (2.71e+01)−
F_{10}	2.13e+03 (4.94e+02)	**1.39e+02 (1.10e+02)**+	1.44e+03 (5.83e+02)+	3.13e+03 (3.46e+02)−	2.84e+03 (6.02e+02)−	6.69e+03 (2.77e+02)−
F_{11}	**1.26e+01 (1.53e+01)**	2.73e+01 (2.89e+01)≈	4.11e+01 (4.76e+01)≈	8.43e+01 (1.84e+01)−	1.16e+02 (4.11e+01)−	8.16e+01 (4.14e+01)−
F_{12}	7.00e+03 (4.23e+03)	1.21e+03 (3.72e+02)+	**1.09e+03 (2.81e+02)**+	2.77e+04 (8.55e+03)−	1.50e+06 (1.19e+06)−	5.75e+04 (8.99e+04)−
F_{13}	**2.61e+01 (1.44e+01)**	4.87e+01 (3.15e+01)−	1.19e+02 (4.00e+02)≈	3.21e+03 (2.88e+03)−	9.58e+03 (1.28e+04)−	2.02e+04 (1.79e+04)−
F_{14}	4.35e+01 (1.64e+01)	**2.66e+01 (4.24e+00)**+	9.08e+01 (5.62e+01)−	2.32e+03 (1.52e+03)−	2.11e+03 (2.28e+03)−	7.10e+03 (5.85e+03)−
F_{15}	1.79e+02 (6.81e+02)	**3.24e+01 (1.97e+01)**+	2.18e+02 (1.84e+02)−	2.13e+03 (1.63e+03)−	1.06e+04 (9.40e+03)−	2.16e+04 (2.27e+04)−
F_{16}	3.54e+02 (2.09e+02)	**7.64e+01 (9.51e+01)**+	5.02e+02 (2.54e+02)−	8.46e+02 (1.53e+02)−	6.66e+02 (2.26e+02)−	4.92e+02 (3.53e+02)≈
F_{17}	**3.80e+01 (2.62e+01)**	5.54e+01 (4.00e+01)≈	1.32e+02 (9.54e+01)−	3.31e+02 (1.13e+02)−	2.55e+02 (1.55e+02)−	1.41e+02 (6.59e+01)−
F_{18}	9.13e+03 (6.75e+03)	**3.51e+01 (1.43e+01)**+	1.60e+02 (1.14e+02)+	6.99e+04 (3.06e+04)−	1.25e+05 (1.01e+05)−	3.67e+05 (1.67e+05)−
F_{19}	**1.46e+01 (6.05e+00)**	1.63e+01 (7.38e+00)≈	1.15e+02 (6.59e+01)−	1.71e+03 (1.69e+03)−	1.37e+04 (1.56e+04)−	1.07e+04 (1.10e+04)−
F_{20}	1.28e+02 (4.87e+01)	**7.06e+01 (5.32e+01)**+	2.97e+02 (1.19e+02)−	3.48e+02 (9.16e+01)−	2.55e+02 (1.47e+02)−	2.21e+02 (1.25e+02)−
F_{21}	2.17e+02 (4.66e+00)	**2.07e+02 (4.27e+00)**+	2.09e+02 (1.67e+01)+	3.05e+02 (3.30e+01)−	2.98e+02 (2.31e+01)−	2.34e+02 (1.16e+01)−
F_{22}	1.00e+02 (3.44e−01)	**1.00e+02 (1.10e−07)**+	6.72e+02 (7.63e+02)−	1.00e+02 (5.05e−07)+	1.00e+02 (9.87e−01)−	1.01e+02 (1.94e+00)−
F_{23}	3.65e+02 (8.16e+00)	3.50e+02 (7.57e+00)+	**3.39e+02 (4.93e+01)**+	6.81e+02 (3.79e+01)−	4.53e+02 (3.19e+01)−	3.96e+02 (1.62e+01)−
F_{24}	4.37e+02 (6.12e+00)	**4.18e+02 (4.73e+00)**+	4.19e+02 (3.06e+00)+	7.39e+02 (4.57e+01)−	5.65e+02 (5.05e+01)−	4.69e+02 (1.62e+01)−
F_{25}	3.87e+02 (3.12e−01)	3.87e+02 (7.55e−03)+	3.87e+02 (1.47e−02)+	**3.85e+02 (1.77e+00)**+	3.86e+02 (1.41e+00)+	4.02e+02 (1.76e+01)−
F_{26}	1.15e+03 (8.28e+01)	5.74e+02 (2.72e+02)+	**3.94e+02 (2.17e+02)**+	2.04e+03 (1.73e+03)≈	2.17e+03 (7.28e+02)−	1.42e+03 (4.69e+02)−
F_{27}	5.17e+02 (5.00e+00)	**5.10e+02 (7.85e+00)**+	5.12e+02 (1.13e+01)+	7.08e+02 (5.42e+01)−	5.39e+02 (1.62e+01)−	5.32e+02 (2.07e+01)−
F_{28}	3.17e+02 (4.13e+01)	3.18e+02 (4.21e+01)−	**3.09e+02 (2.96e+01)**+	3.27e+02 (3.17e+01)−	3.87e+02 (4.33e+01)−	4.30e+02 (2.67e+01)−
F_{29}	4.61e+02 (3.84e+01)	**4.43e+02 (4.55e+01)**+	4.93e+02 (1.04e+02)≈	7.80e+02 (1.21e+02)−	7.48e+02 (1.96e+02)−	6.15e+02 (9.09e+01)−
F_{30}	2.82e+03 (6.75e+02)	**2.16e+03 (1.65e+02)**+	2.84e+03 (1.94e+03)−	3.32e+03 (3.89e+02)−	3.31e+04 (3.13e+04)−	2.57e+04 (2.78e+04)−
	+	22	15	2	1	0
	–	3	9	26	28	29
	≈	5	6	2	1	1

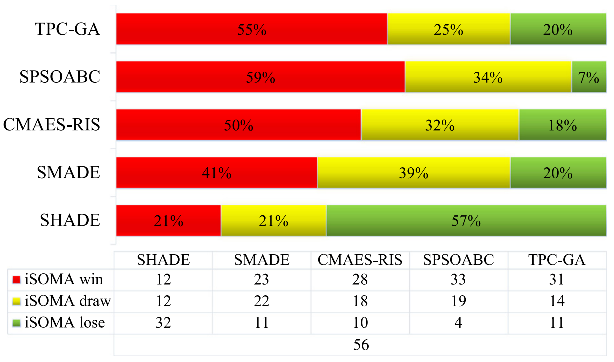


Fig. 10. The summarized comparison results between the iSOMA and other algorithms tested on the CEC13 (for both 10*D* and 30*D*).

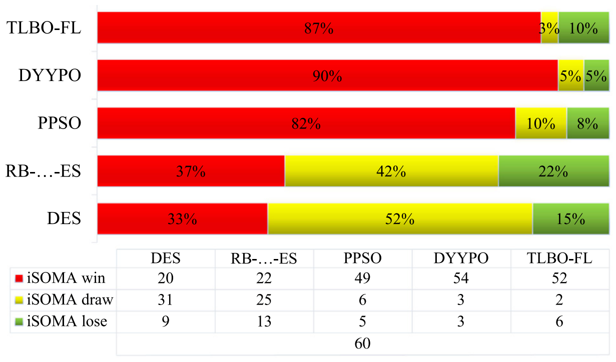


Fig. 12. The summarized comparison results between the iSOMA and other algorithms tested on the CEC17 (for both 10*D* and 30*D*).

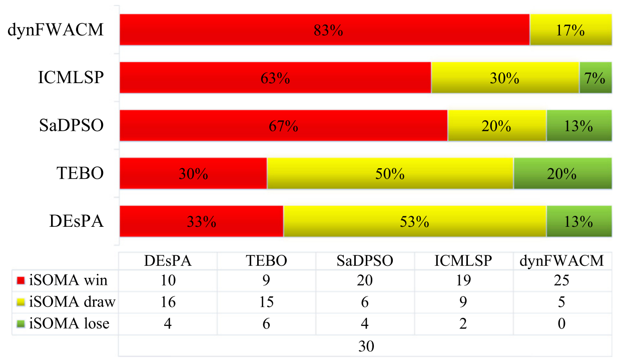


Fig. 11. The summarized comparison results between the iSOMA and other algorithms tested on the CEC15 (for both 10*D* and 30*D*).

obstacles are static and will be detected by the drone sensor system.

To deal with the path planning problem of drones as an optimization problem, the first thing to do is build a fitness function. This function was proposed in our previous publication in [50] and given in Eq. (6). Accordingly, the mission of the iSOMA is to create all consecutive positions from the starting position, forming a discrete point set that the drone must pass through in real-time.

$$f_{value} = a_1 * e^{a_2 * dis_{tar}^{a_3}} + \sum_{n=0}^{n_{obs}} b_1 * e^{b_2 * dis_{obs}^{b_3}} \tag{6}$$

where:

- f_{value} : the cost value of the drone problem,

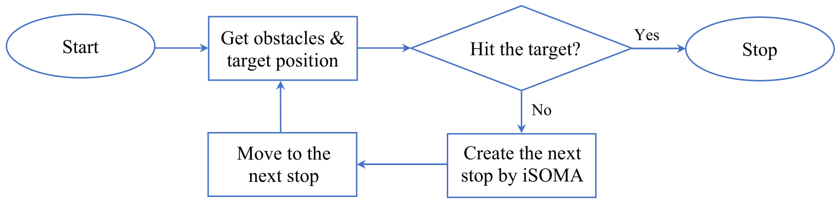

Start
Get obstacles & target position
Hit the target?
Yes
Stop
No
Create the next stop by iSOMA
Move to the next stop

Fig. 13. The operation flowchart of drones using iSOMA algorithm.

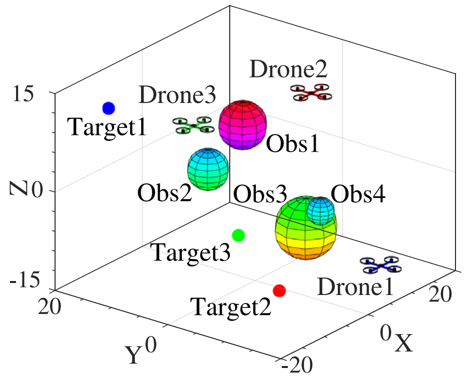


Fig. 14. The working space of drones and obstacles.

Table 13
Obstacles position and their radius (in meter).

Obstacle$_i$	1	2	3	4
x_i	−3.5	1.5	4.5	−2.5
y_i	−5	9.5	−5.5	−13.5
z_i	10	−1	−7	0
r_i	3.5	3	4.5	2

Table 14
Intentional placement of drones and their targets (in meter).

The object	Drone 1	Drone 2	Drone 3	Target 1	Target 2	Target 3
x_i	11	6	−1	−5	−10	−9
y_i	−14	−5	10	14	−12	−4
z_i	−13	13	6	9	−10	−4

- n_{obs} : amount of obstacles found,
- dis_{tar} : distance between the drone and the target, in Eq. (7),
- dis_{obs} : distance between the drone and the found obstacle, in Eq. (8),
- a_x, b_x : equilibrium coefficients ($x = 1, 2, 3.$).

$$dis_{tar} = \sqrt{(x_{target} - x_{drone})^2 + (y_{target} - y_{drone})^2 + (z_{target} - z_{drone})^2} \tag{7}$$

$$dis_{obs} = \sqrt{(x_{obs} - x_{drone})^2 + (y_{obs} - y_{drone})^2 + (z_{obs} - z_{drone})^2} \tag{8}$$

The drones' problem of catching the target and avoiding obstacles has now evolved into an optimization problem.

6.2. Experimental setup

The obstacle positions are given in Table 13. They are intentionally arranged to prevent the direct movement of the drones to the targets. Table 14 presents the drones starting position and respective targets.

In this simulation, we have assumed that the drones can move within a radius $r_{movingstep} = 0.3$ m without any problems. Sensors on drones are capable of detecting obstacles within a radius of 2 m. The drones are provided with the location of the target as indicated above.

6.3. Results and discussions

Three-dimensional trajectories of the drone are shown in Fig. 15. The defined obstacles are illustrative by the four spheres, whose positions and radius are given in Table 13. The first, second, and third drones are characterized by blue, red, and green, respectively. The trajectories are plotted in dotted lines with respective colors. The obstacles and targets in this study are stationary objects as stated, and the drone's goal is to reach its target without hitting any detected obstacles as well as without hitting each other. To explain the movement, the 25*th*, 66*th*, 131*st* and 168*th* steps were chosen. The move is continuous, despite it being viewed as discrete steps.

Step 25*th* in Fig. 15 reveals that all drones have progressed 25 steps. The iSOMA algorithm serves as a reference point creator for drones at each step, providing the next stop they would move to based on the current data, such as target position and detected obstacles. It is worth noting that drones are also obstacles to each other. However, further analysis of dynamic obstacles as well as moving targets for drone problems is not within the scope of this research and will be covered in our upcoming publications.

Return to the first step, there is no obstacle in the sensor system's detection region when the movement begins due to the intentional arrangement. Thus, there is only the first part in Eq. (6) pulling the drone toward the target. As a result, the drone will fly directly to the target. However, the second component of the cost function appears in the following stages, when the drone's sensors detect obstacles. Due to the cost value being inversely proportional to the distance to the drone, if the next step of the drone is still moving forward, approaching the obstacles, the value will increase. On the other hand, if the drone moves backward, away from obstacles, the distance between the drone and the target increases accordingly, and thus the cost value increases as well. A suitable position will be calculated and proposed by the iSOMA algorithm to obtain the minimum cost value. This process will be repeated continuously until the drone hits the target at step 168*th* as shown in Fig. 15.

Figs. 16, 17, and 18 show various views of the trajectories for a thorough observation of the movement process. They are captured in two-dimensional space at the same time as in Fig. 15. Since the distances from the starting points to the targets are different, their travel times differ as well. The third drone has the shortest distance, so it catches the target firstly, at step 94*th*. It is followed by the second drone at step 154*th* and the first drone at step 168*th*.

7. Conclusion

In this article, the iSOMA is proposed, with the algorithm divided into four processes: the initialization, self-organizing, migrating, and replacement process. The spotlight of the self-organizing and migrating process is to pick migrants and leaders and propose the jumps in order. It guarantees the algorithm's

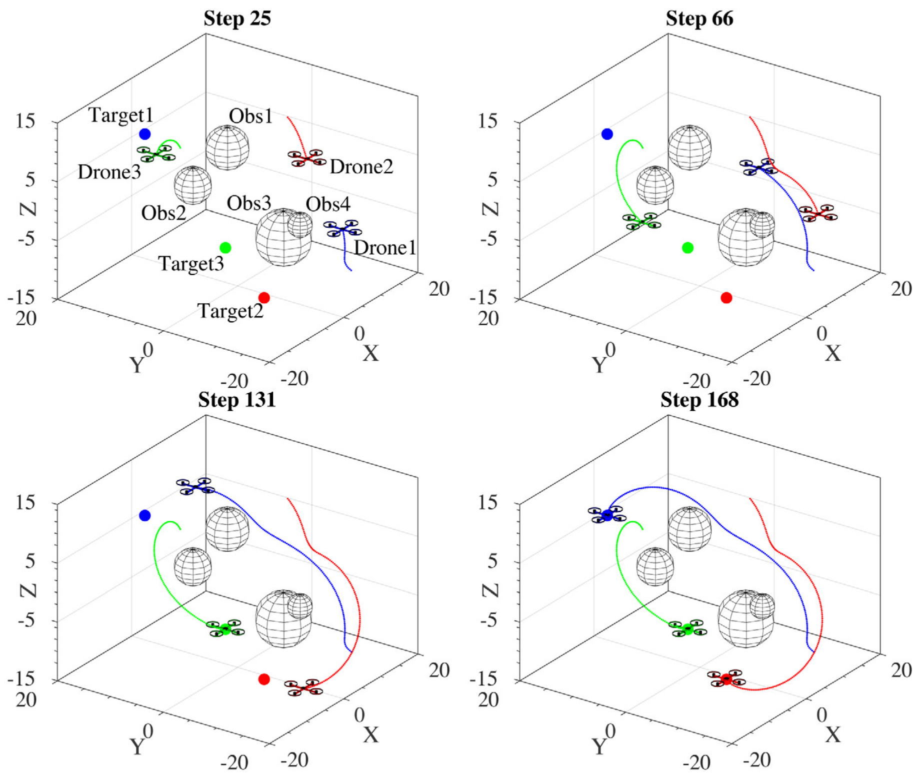


Fig. 15. The operation of the drones in three-dimensional space, captured at four representative steps.

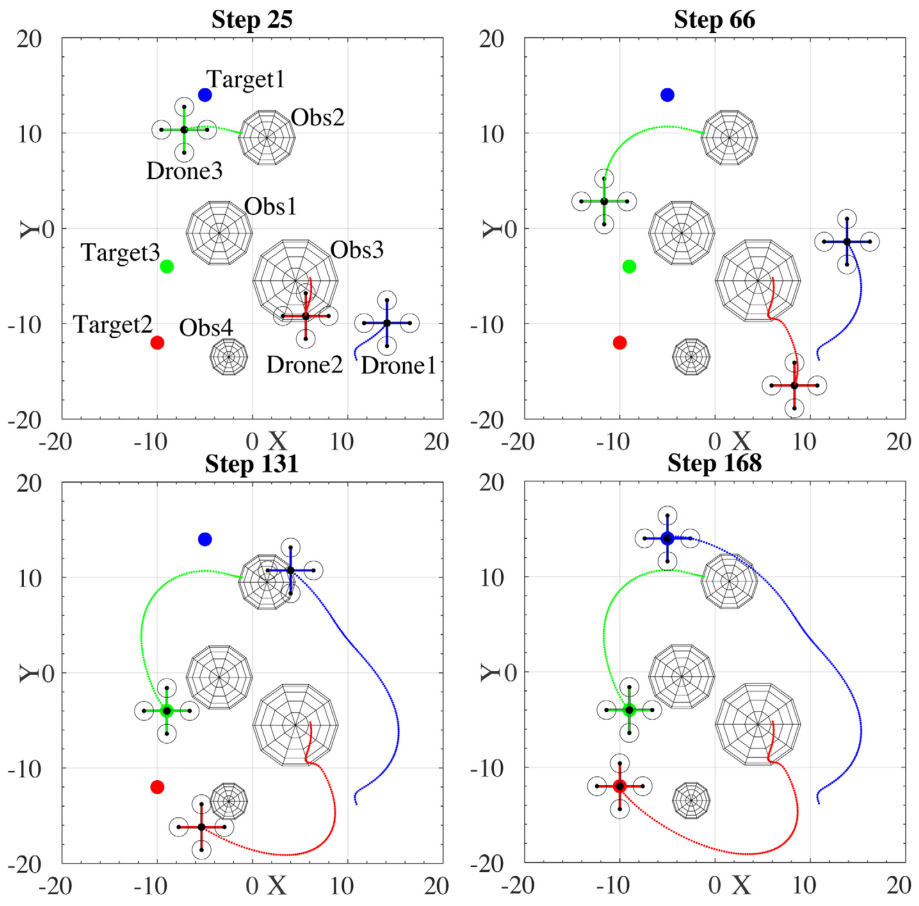


Fig. 16. Drone trajectory simulation in X-Y view.

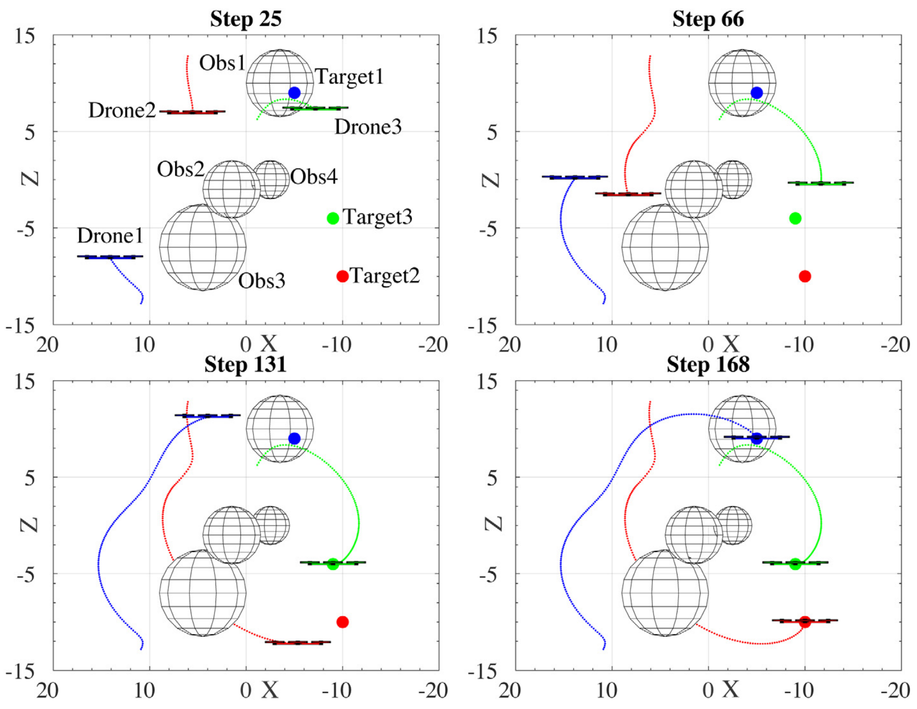


Fig. 17. Drone trajectory simulation in X-Z view.

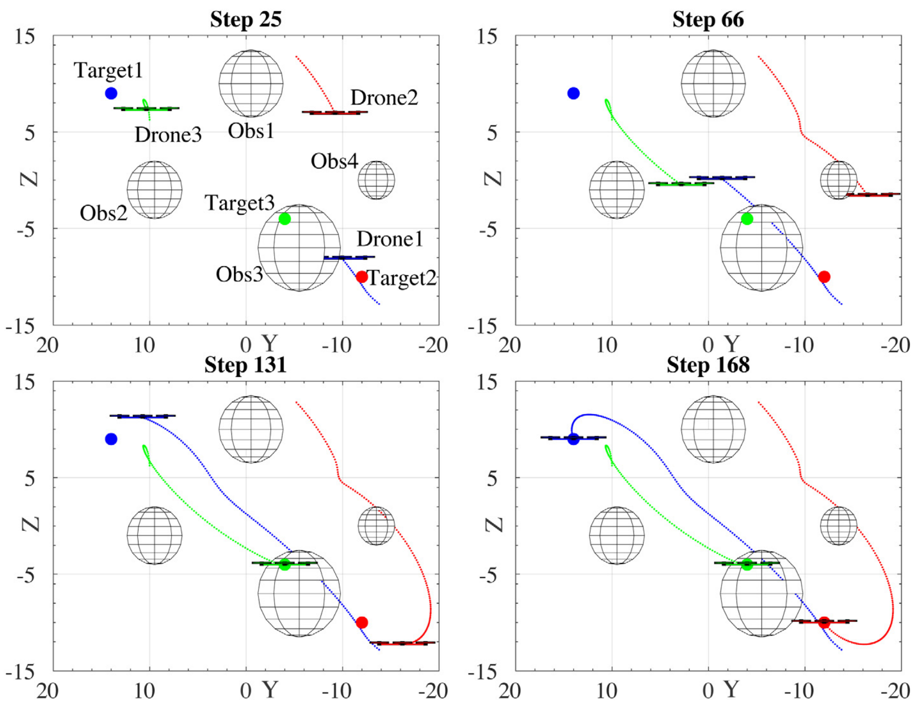


Fig. 18. Drone trajectory simulation in Y-Z view.

speed as well as the equilibrium between the two phases of exploration and exploitation. Furthermore, the application of immediate updates, as well as the narrowing of the search space and the replacement of individuals when the global best is not updated, make iSOMA have superior performance than previous versions.

On three IEEE CEC benchmark suites, including CEC13, CEC15, and CEC17, the iSOMA has been proved and outperformed other variants of optimization algorithms. These comparative outcomes show the usefulness of the proposed algorithm. Consequently, it validates efficiency, competitiveness, and promise against traditional and well-known algorithms such as ABC, PSO, GA, and DE.

Furthermore, this study solved the real-time navigation problem for drones using swarm intelligence, of which the iSOMA algorithm is representative. In particular, how to consider the drone-movement problem as an optimization problem was specified. In which the optimal function was proposed based on the simple but effective principle of being close to the target, far from the obstacle. The trajectory of the drone is a set of real-time continuous points generated by the iSOMA for each drone member, independent of any central controller. However, multiple dynamic obstacles and even moving targets have not been investigated in this study. What if the number of obstacles and drones is relatively large? Will it lead to the drones being trapped and unable to move or collide with each other? The solutions to such issues will be addressed in our subsequent studies.

With outstanding performance, real-world applications that use the iSOMA algorithm will promise to deliver superior power, catching up with the ever-increasing demands of technical development.

CRediT authorship contribution statement

Quoc Bao Diep: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Visualization, Writing – original draft, Writing – review & editing. **Thanh Cong Truong:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Visualization, Writing – original draft, Writing – review & editing. **Swagatam Das:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Visualization, Writing – original draft, Writing – review & editing. **Ivan Zelinka:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Visualization, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The following grants are acknowledged for the financial support provided for this research: Grant of SGS, Czech Republic No. SP2021/72, VSB-Technical University of Ostrava.

References

[1] J.D. Ser, E. Osaba, D. Molina, X.-S. Yang, S. Salcedo-Sanz, D. Camacho, S. Das, P.N. Suganthan, C.A.C. Coello, F. Herrera, Bio-inspired computation: Where we stand and what's next, Swarm Evol. Comput. 48 (2019) 220–250, http://dx.doi.org/10.1016/j.swevo.2019.04.008, URL http://www.sciencedirect.com/science/article/pii/S2210650218310277.

[2] K.V. Price, Differential evolution, in: I. Zelinka, V. Snášel, A. Abraham (Eds.), Handbook of Optimization: From Classical to Modern Approach, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 187–214, http://dx.doi.org/10.1007/978-3-642-30504-7_8, URL https://link.springer.com/chapter/10.1007/978-3-642-30504-7_8.

[3] S. Das, P.N. Suganthan, Differential evolution: A survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (1) (2011) 4–31, http://dx.doi.org/10.1109/TEVC.2010.2059031, URL https://ieeexplore.ieee.org/abstract/document/5601760.

[4] W.-f. Gao, S.-y. Liu, A modified artificial bee colony algorithm, Comput. Oper. Res. 39 (3) (2012) 687–697, http://dx.doi.org/10.1016/j.cor.2011.06.007, URL http://www.sciencedirect.com/science/article/pii/S0305054811001699.

[5] D. Karaboga, B. Basturk, Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems, in: P. Melin, O. Castillo, L.T. Aguilar, J. Kacprzyk, W. Pedrycz (Eds.), Foundations of Fuzzy Logic and Soft Computing, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 789–798, http://dx.doi.org/10.1007/978-3-540-72950-1_77, URL https://link.springer.com/chapter/10.1007/978-3-540-72950-1_77.

[6] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, Vol. 4, 1995, pp. 1942–1948, http://dx.doi.org/10.1109/ICNN.1995.488968, URL https://ieeexplore.ieee.org/abstract/document/488968.

[7] J.C. Bansal, Particle swarm optimization, in: J.C. Bansal, P.K. Singh, N.R. Pal (Eds.), Evolutionary and Swarm Intelligence Algorithms, Springer International Publishing, Cham, 2019, pp. 11–23, http://dx.doi.org/10.1007/978-3-319-91341-4_2, URL https://link.springer.com/chapter/10.1007/978-3-319-91341-4_2.

[8] I. Zelinka, SOMA — Self-organizing migrating algorithm, in: New Optimization Techniques in Engineering, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 167–217, http://dx.doi.org/10.1007/978-3-540-39930-8_7.

[9] I. Zelinka, SOMA—Self-organizing migrating algorithm, in: D. Davendra, I. Zelinka (Eds.), Self-Organizing Migrating Algorithm: Methodology and Implementation, Springer International Publishing, Cham, 2016, pp. 3–49, http://dx.doi.org/10.1007/978-3-319-28161-2_1.

[10] L. dos Santos Coelho, Self-organizing migrating strategies applied to reliability-redundancy optimization of systems, IEEE Trans. Reliab. 58 (3) (2009) 501–510, http://dx.doi.org/10.1109/TR.2009.2019514, URL https://ieeexplore.ieee.org/abstract/document/4967917.

[11] I. Zelinka, M. Bukacek, SOMA swarm algorithm in computer games, in: L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L.A. Zadeh, J.M. Zurada (Eds.), Artificial Intelligence and Soft Computing, Springer International Publishing, Cham, 2016, pp. 395–406, http://dx.doi.org/10.1007/978-3-319-39384-1_34, URL https://link.springer.com/chapter/10.1007/978-3-319-39384-1_34.

[12] I. Zelinka, L. Sikora, StarCraft: Brood war — Strategy powered by the SOMA swarm algorithm, in: 2015 IEEE Conference on Computational Intelligence and Games (CIG), 2015, pp. 511–516, http://dx.doi.org/10.1109/CIG.2015.7317903, URL https://ieeexplore.ieee.org/abstract/document/7317903.

[13] D.Q. Bao, I. Zelinka, Obstacle avoidance for swarm robot based on self-organizing migrating algorithm, Procedia Comput. Sci. 150 (2019) 425–432, http://dx.doi.org/10.1016/j.procs.2019.02.073, Proceedings of the 13th International Symposium "Intelligent Systems 2018" (INTELS'18), 22-24 October, 2018, St. Petersburg, Russia. URL http://www.sciencedirect.com/science/article/pii/S187705091930420X.

[14] Q.B. Diep, I. Zelinka, R. Senkerik, An algorithm for swarm robot to avoid multiple dynamic obstacles and to catch the moving target, in: L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, J.M. Zurada (Eds.), Artificial Intelligence and Soft Computing, Springer International Publishing, Cham, 2019, pp. 666–675, http://dx.doi.org/10.1007/978-3-030-20915-5_59, URL https://link.springer.com/chapter/10.1007/978-3-030-20915-5_59.

[15] L. Skanderova, T. Fabian, I. Zelinka, Self-adapting self-organizing migrating algorithm, Swarm Evol. Comput. 51 (2019) 100593, http://dx.doi.org/10.1016/j.swevo.2019.100593, URL http://www.sciencedirect.com/science/article/pii/S2210650219300756.

[16] D. Singh, S. Agrawal, K. Deep, C-SOMAQI: Self organizing migrating algorithm with quadratic interpolation crossover operator for constrained global optimization, in: D. Davendra, I. Zelinka (Eds.), Self-Organizing Migrating Algorithm: Methodology and Implementation, Springer International Publishing, Cham, 2016, pp. 147–165, http://dx.doi.org/10.1007/978-3-319-28161-2_7, URL https://link.springer.com/chapter/10.1007/978-3-319-28161-2_7.

[17] L. Tomaszek, I. Zelinka, M. Chadli, On the leader selection in the self-organizing migrating algorithm, MENDEL 25 (1) (2019) 171–178, http://dx.doi.org/10.13164/mendel.2019.1.171, URL https://mendel-journal.org/index.php/mendel/article/view/95.

[18] M. Pluhacek, R. Senkerik, A. Viktorin, T. Kadavy, Self-organizing migrating algorithm with non-binary perturbation, in: A. Zamuda, S. Das, P.N. Suganthan, B.K. Panigrahi (Eds.), Swarm, Evolutionary, and Memetic Computing and Fuzzy and Neural Computing, Springer International Publishing, Cham, 2020, pp. 43–57, http://dx.doi.org/10.1007/978-3-030-37838-7_5, URL https://link.springer.com/chapter/10.1007/978-3-030-37838-7_5.

[19] T. Kadavy, M. Pluhacek, R. Senkerik, A. Viktorin, Introducing self-adaptive parameters to self-organizing migrating algorithm, in: 2019 IEEE Congress on Evolutionary Computation (CEC), 2019, pp. 2908–2914, http://dx.doi.org/10.1109/CEC.2019.8790283, URL https://ieeexplore.ieee.org/abstract/document/8790283.

[20] Q.B. Diep, Self-organizing migrating algorithm team to team adaptive – SOMA T3A, in: 2019 IEEE Congress on Evolutionary Computation (CEC), 2019, pp. 1182–1187, http://dx.doi.org/10.1109/CEC.2019.8790202, URL https://ieeexplore.ieee.org/abstract/document/8790202.

[21] Q.B. Diep, I. Zelinka, S. Das, R. Senkerik, SOMA T3A for solving the 100-digit challenge, in: A. Zamuda, S. Das, P.N. Suganthan, B.K. Panigrahi (Eds.), Swarm, Evolutionary, and Memetic Computing and Fuzzy and Neural Computing, Springer International Publishing, Cham, 2020, pp. 155–165, http://dx.doi.org/10.1007/978-3-030-37838-7_14, URL https://link.springer.com/chapter/10.1007/978-3-030-37838-7_14.

[22] Q. Diep, I. Zelinka, S. Das, Self-organizing migrating algorithm Pareto, MENDEL 25 (1) (2019) 111–120, http://dx.doi.org/10.13164/mendel.2019.1.111, URL https://mendel-journal.org/index.php/mendel/article/view/87.

[23] T.C. Truong, Q.B. Diep, I. Zelinka, R. Senkerik, Pareto-based self-organizing migrating algorithm solving 100-digit challenge, in: A. Zamuda, S. Das, P.N. Suganthan, B.K. Panigrahi (Eds.), Swarm, Evolutionary, and Memetic Computing and Fuzzy and Neural Computing, Springer International Publishing, Cham, 2020, pp. 13–20, http://dx.doi.org/10.1007/978-3-030-37838-7_2, URL https://link.springer.com/chapter/10.1007/978-3-030-37838-7_2.

[24] F. Lezama, J.a. Soares, R. Faia, Z. Vale, Hybrid-adaptive differential evolution with decay function (HyDE-DF) applied to the 100-digit challenge competition on single objective numerical optimization, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, in: GECCO '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 7–8, http://dx.doi.org/10.1145/3319619.3326747, URL https://dl.acm.org/doi/abs/10.1145/3319619.3326747.

[25] K. Price, N. Awad, M. Ali, P. Suganthan, The 2019 100-digit challenge on real-parameter, single objective optimization: Analysis of results, 2019, URL https://github.com/P-N-Suganthan/CEC2019.

[26] R. Rao, Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems, Int. J. Ind. Eng. Comput. 7 (1) (2016) 19–34, http://dx.doi.org/10.5267/j.ijiec.2015.8.004.

[27] R. Venkata Rao, A. Saroj, A self-adaptive multi-population based Jaya algorithm for engineering optimization, Swarm Evol. Comput. 37 (2017) 1–26, http://dx.doi.org/10.1016/j.swevo.2017.04.008, URL https://www.sciencedirect.com/science/article/pii/S2210650216303510.

[28] R. Venkata Rao, Jaya optimization algorithm and its variants, in: Jaya: An Advanced Optimization Algorithm and its Engineering Applications, Springer International Publishing, Cham, 2019, pp. 9–58, http://dx.doi.org/10.1007/978-3-319-78922-4_2.

[29] R. Rao, Rao algorithms: Three metaphor-less simple algorithms for solving optimization problems, Int. J. Ind. Eng. Comput. 11 (1) (2020) 107–130, http://dx.doi.org/10.5267/j.ijiec.2019.6.002.

[30] J. Liang, B. Qu, P. Suganthan, A.G. Hernández-Díaz, Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization, Technical Report 201212 (34), Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, 2013, pp. 281–295.

[31] J. Liang, B. Qu, P. Suganthan, Q. Chen, Problem Definitions and Evaluation Criteria for the CEC 2015 Competition on Learning-Based Real-Parameter Single Objective Optimization, Vol. 29, Technical Report201411A, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, 2014, pp. 625–640.

[32] N. Awad, M. Ali, J. Liang, B. Qu, P. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Bound Constrained Real-Parameter Numerical Optimization, Technical Report, Nanyang Technological University Singapore, 2016.

[33] R. Tanabe, A. Fukunaga, Evaluating the performance of SHADE on CEC 2013 benchmark problems, in: 2013 IEEE Congress on Evolutionary Computation, 2013, pp. 1952–1959, http://dx.doi.org/10.1109/CEC.2013.6557798, URL https://ieeexplore.ieee.org/abstract/document/6557798.

[34] F. Caraffini, F. Neri, J. Cheng, G. Zhang, L. Picinali, G. Iacca, E. Mininno, Super-fit multicriteria adaptive differential evolution, in: 2013 IEEE Congress on Evolutionary Computation, 2013, pp. 1678–1685, http://dx.doi.org/10.1109/CEC.2013.6557763, URL https://ieeexplore.ieee.org/abstract/document/6557763.

[35] F. Caraffini, G. Iacca, F. Neri, L. Picinali, E. Mininno, A CMA-ES super-fit scheme for the re-sampled inheritance search, in: 2013 IEEE Congress on Evolutionary Computation, 2013, pp. 1123–1130, http://dx.doi.org/10.1109/CEC.2013.6557692, URL https://ieeexplore.ieee.org/abstract/document/6557692.

[36] M. El-Abd, Testing a particle swarm optimization and artificial bee colony hybrid algorithm on the CEC13 benchmarks, in: 2013 IEEE Congress on Evolutionary Computation, 2013, pp. 2215–2220, http://dx.doi.org/10.1109/CEC.2013.6557832, URL https://ieeexplore.ieee.org/abstract/document/6557832.

[37] S.M. Elsayed, R.A. Sarker, D.L. Essam, A genetic algorithm for solving the CEC'2013 competition problems on real-parameter optimization, in: 2013 IEEE Congress on Evolutionary Computation, 2013, pp. 356–360, http://dx.doi.org/10.1109/CEC.2013.6557591, URL https://ieeexplore.ieee.org/abstract/document/6557591.

[38] N. Awad, M.Z. Ali, R.G. Reynolds, A differential evolution algorithm with success-based parameter adaptation for CEC2015 learning-based optimization, in: 2015 IEEE Congress on Evolutionary Computation (CEC), 2015, pp. 1098–1105, http://dx.doi.org/10.1109/CEC.2015.7257012, URL https://ieeexplore.ieee.org/abstract/document/7257012.

[39] Y. Zheng, X. Wu, Tuning maturity model of ecogeography-based optimization on CEC 2015 single-objective optimization test problems, in: 2015 IEEE Congress on Evolutionary Computation (CEC), 2015, pp. 1018–1024, http://dx.doi.org/10.1109/CEC.2015.7257001, URL https://ieeexplore.ieee.org/abstract/document/7257001.

[40] J.J. Liang, L. Guo, R. Liu, B.Y. Qu, A self-adaptive dynamic particle swarm optimizer, in: 2015 IEEE Congress on Evolutionary Computation (CEC), 2015, pp. 3206–3213, http://dx.doi.org/10.1109/CEC.2015.7257290, URL https://ieeexplore.ieee.org/abstract/document/7257290.

[41] L. Chen, C. Peng, H. Liu, S. Xie, An improved covariance matrix leaning and searching preference algorithm for solving CEC 2015 benchmark problems, in: 2015 IEEE Congress on Evolutionary Computation (CEC), 2015, pp. 1041–1045, http://dx.doi.org/10.1109/CEC.2015.7257004, URL https://ieeexplore.ieee.org/abstract/document/7257004.

[42] C. Yu, L.C. Kelley, Y. Tan, Dynamic search fireworks algorithm with covariance mutation for solving the CEC 2015 learning based competition problems, in: 2015 IEEE Congress on Evolutionary Computation (CEC), 2015, pp. 1106–1112, http://dx.doi.org/10.1109/CEC.2015.7257013, URL https://ieeexplore.ieee.org/abstract/document/7257013.

[43] D. Jagodziński, J. Arabas, A differential evolution strategy, in: 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 1872–1876, http://dx.doi.org/10.1109/CEC.2017.7969529, URL https://ieeexplore.ieee.org/abstract/document/7969529.

[44] R. Biedrzycki, A version of IPOP-CMA-ES algorithm with midpoint for CEC 2017 single objective bound constrained problems, in: 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 1489–1494, http://dx.doi.org/10.1109/CEC.2017.7969479, URL https://ieeexplore.ieee.org/abstract/document/7969479.

[45] A. Tangherloni, L. Rundo, M.S. Nobile, Proactive particles in swarm optimization: A settings-free algorithm for real-parameter single objective optimization problems, in: 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 1940–1947, http://dx.doi.org/10.1109/CEC.2017.7969538, URL https://ieeexplore.ieee.org/abstract/document/7969538.

[46] D. Maharana, R. Kommadath, P. Kotecha, Dynamic yin-yang pair optimization and its performance on single objective real parameter problems of CEC 2017, in: 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 2390–2396, http://dx.doi.org/10.1109/CEC.2017.7969594, URL https://ieeexplore.ieee.org/abstract/document/7969594.

[47] R. Kommadath, P. Kotecha, Teaching learning based optimization with focused learning and its performance on CEC2017 functions, in: 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 2397–2403, http://dx.doi.org/10.1109/CEC.2017.7969595, URL https://ieeexplore.ieee.org/abstract/document/7969595.

[48] J. Derrac, S. García, D. Molina, 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. 1 (1) (2011) 3–18, http://dx.doi.org/10.1016/j.swevo.2011.02.002, URL http://www.sciencedirect.com/science/article/pii/S2210650211000034.

[49] J. Carrasco, S. García, M. Rueda, S. Das, F. Herrera, Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review, Swarm Evol. Comput. 54 (2020) 100665, http://dx.doi.org/10.1016/j.swevo.2020.100665, URL https://www.sciencedirect.com/science/article/pii/S2210650219302639.

[50] Q.B. Diep, T.C. Truong, I. Zelinka, Obstacle avoidance for drones based on the self-organizing migrating algorithm, in: L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, J.M. Zurada (Eds.), Artificial Intelligence and Soft Computing, Springer International Publishing, Cham, 2020, pp. 376–386, http://dx.doi.org/10.1007/978-3-030-61401-0_35, URL https://link.springer.com/chapter/10.1007/978-3-030-61401-0_35.