VŠB – Technical University of Ostrava
Faculty of Electrical Engineering
and Computer Science

Swarm Robotics

PHD THESIS

2021

Quoc Bao Diep

VSB TECHNICAL UNIVERSITY OF OSTRAVA | FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Swarm Robotics

Hejnová robotika

Quoc Bao Diep

Ph.D. Thesis

Supervisor: prof. Ing. Ivan Zelinka, Ph.D.

Ostrava, 2021

Abstract

This study analyzes and designs the Swarm intelligence (SI) that Self-organizing migrating algorithm (SOMA) represents to solve industrial practice as well as academic optimization problems, and applies them to swarm robotics. Specifically, the characteristics of SOMA are clarified, shaping the basis for the analysis of SOMA's strengths and weaknesses for the release of SOMA T3A, SOMA Pareto and iSOMA, with outstanding performance, confirmed by well-known test suites from IEEE CEC 2013, 2015, 2017, and 2019. Besides, the dynamic path planning problem for swarm robotics is handled by the proposed algorithms considered as a prime instance. The computational and simulation results on Matlab have proven the performance of the novel algorithms as well as the correctness of the obstacle avoidance method for mobile robots and drones. Furthermore, two out of the three proposed versions achieved the tie for 3^{rd} (the same ranking with HyDE-DF) and 5^{th} place in the 100-Digit Challenge at CEC 2019, GECCO 2019, and SEMCCO 2019 competition, something that any other version of SOMA has yet to do. They show promising possibilities that SOMA and SI algorithms offer.

Keywords: Swarm intelligence; Swarm robotics; Self-organizing migrating algorithm; Optimization; Mobile robot

Abstrakt

Tato práce se zabývá analýzou a vylepšením hejnové inteligence, kterou představuje samoorganizující se migrační algoritmus s možností využití v průmyslové praxi a se zaměřením na hejnovou robotiku. Je analyzován algoritmus SOMA, identifikovány silné a slabé stránky a navrženy nové verze SOMA jako SOMA T3A, SOMA Pareto, iSOMA s vynikajícím výkonem, potvrzeným známými testovacími sadami IEEE CEC 2013, 2015, 2017 a 2019. Tyto verze jsou pak aplikovány na problém s dynamickým plánováním dráhy pro hejnovou robotiku. Výsledky výpočtů a simulace v Matlabu prokázaly výkonnost nových algoritmů a správnost metody umožňující vyhýbání se překážkám u mobilních robotů a dronů. Kromě toho dvě ze tří navržených verzí dosáhly na 3. a 5. místo v soutěži 100-Digit Challenge na CEC 2019, GECCO 2019 a SEMCCO 2019, což je potvrzení navržených inovací. Práce tak demonstruje nejen vylepšení SOMA, ale i slibné možnosti hejnové inteligence.

Klíčová slova: Inteligence roje; hejnová robotika; Samoorganizující se migrační algoritmus; optimalizace; robotika

I would like to thank my supervisor, prof. Ing. Ivan Zelinka, Ph.D. for his invaluable advice, continuous support, and patience during my study; his immense knowledge and plentiful experience have encouraged me in all the time of my academic research life at Czech Republic. I would like to express my gratitude to my parents and relatives for their support and encouragement.

Contents

List of Symbols and Abbreviations

ABC	–	Artificial Bee Colony
ACO	–	Ant Colony Optimization
CEC	–	Congress on Evolutionary Computation
CMAES-RIS	–	A CMA-ES Super-fit Scheme for the Re-sampled Inheritance Search
DE	–	Differential Evolution
DE-APC	–	Differential Evolution with Automatic Parameter Configuration
DES	–	A Differential Evolution Strategy
DEsPA	–	Differential Evolution Algorithm with Success-based Parameter Adaptation
DYYPO	–	Dynamic Yin-Yang Pair Optimization
dynFWACM	–	Dynamic Search Fireworks Algorithm with Covariance Mutation
EA	–	Evolutionary Algorithm
FA	–	Firefly Algorithm
FEs	–	Function Evaluations
fk-PSO	–	A Self-adaptive Heterogeneous Particle Swarm Optimization
GA	–	Genetic Algorithm
HFPSO	–	Hybrid Firefly and Particle Swarm Optimization
ICMLSP	–	An Improved Covariance Matrix Leaning & Searching Preference Algorithm
iSOMA	–	Self-Organizing Migrating Algorithm with Narrowing Search Space Strategy
MaxFEs	–	Maximum of Function Evaluations
PPSO	–	Proactive Particles in Swarm Optimization
PSO	–	Particle Swarm Optimization
SaDPSO	–	A Self-adaptive Dynamic Particle Swarm Optimizer
SAMPE-Jaya	–	Self-Adaptive Multi Population Elitist Jaya
SHADE	–	Success-History Based Parameter Adaptation for Differential Evolution
SI	–	Swarm Intelligence
SMASE	–	Super-fit Multicriteria Adaptive Differential Evolution
SOMA	–	Self-Organizing Migrating Algorithm
SOMA ATO	–	Self-Organizing Migrating Algorithm All To One Version
SOMA ATA	–	Self-Organizing Migrating Algorithm All To All Version
SOMA Pareto	–	Pareto-Based Self-Organizing Migrating Algorithm
SOMA T3A	–	Self-Organizing Migrating Algorithm Team To Team Adaptive
SPSOABC	–	Particle Swarm Optimization and Artificial Bee Colony Hybrid Algorithm
SSA	–	Salp Swarm Algorithm
TEBO	–	Tuning Maturity Model of Ecogeography-Based Optimization
TLBO-FL	–	Teaching Learning Based Optimization with Focused Learning
UAVs	–	Unmanned Aerial Vehicles
WOA	–	Whale Optimization Algorithm

List of Figures

List of Tables

1 INTRODUCTION

With the continuous development of science and technology, many industrial practices and academic problems arise challenging and most of them can be transformed into optimization problems [1]. The swarm intelligence (SI) algorithm is one of the most effective and well-attended methods to find the global optimal solution to such problems, such as artificial bee colony (ABC) [2, 3], particle swarm optimization (PSO) [4, 5], and self-organizing migrating algorithm (SOMA) [6, 7] that is a subject of the reported research here.

Proposed in the 2000s, SOMA, 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, let for example mention the solution of a partial differential equation of civil engineering, describing the beam in the wall under statical load, synthesis of electrical circuits (train control, heating of the house and traffic light control) [8], and more.

SOMA has not been used only in various applications or interdisciplinary implementations, and it has also been used in the computer game StarCraft. In this game was SOMA used in a real-time regime so that trajectories of individuals were one-to-one trajectories of game bot warriors. Another game application (currently available on Google Play[1]) is the game Tic Tac Toe powered by SOMA algorithm. It has been used not only in classical optimization tasks but also in interdisciplinary research, for example, in evolutionary dynamics and its relations with complex network structures [9, 10, 11].

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

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 minimal traps. Most canonical versions of SI algorithms have been 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, the research and development of algorithms are essential.

Many 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 self-organizing migrating algorithm [12], self-organizing migrating algorithm with quadratic interpolation crossover operator for constrained global optimization [13], the version of the leader selection in the self-organizing migrating algorithm [14], self-organizing migrating algorithm

[1] https://play.google.com/store/apps/details?id=cz.bukacek.soma_tictactoe

with non-binary perturbation [15], and self-adaptive parameters to self-organizing migrating algorithm [16]. In particular, the two latest versions, team to team adaptive – SOMA T3A [17, 18] and Pareto-based SOMA [19, 20], have made great strides, holding 3^{rd} (the same ranking with HyDE-DF [21]) and 5^{th} out of 36 algorithms participating in the 100-Digit Challenge Competition respectively, which reported in [22] 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).

However, they have proven their effectiveness in the 100-Digit Challenge Competition, which does not mean that the algorithms will be victorious at all different test suites. Furthermore, many academic problems and real-world applications are increasingly complex, requiring algorithms to be more efficient, solving problems faster, and more accurately. ***Is it possible to improve the performance of the SOMA algorithm further, and how?*** That prompted me to research and develop novel algorithms to answer this research question, satisfying these illustration requirements mentioned above.

And what about swarm robotics? ***How to turn the specific problem of swarm robotics into an optimization problem? And how to apply swarm intelligence to solve a particular aspect of mobile robots and even drones?*** These research questions will be answered in the next part of the thesis. It can be revealed that one of the most important issues for swarm robotics applications is catching up with moving targets and avoiding multiple dynamic obstacles on the ground even in space. It's complicated in that it requires the algorithm to work in real-time to avoid dynamic obstacles that are standing or moving in an unknown environment where the robot does not know their position until detecting them by sensors arranged on the robot.

1.1 Main Goals and Contributions

The main goals of the doctoral thesis can be divided as follows:

- State of the art in the research field
- Design the novel self-organizing migrating algorithms to deal with well-known test suites and real-world problems
  - Self-organizing migrating algorithm team to team adaptive
  - Pareto-based self-organizing migrating algorithm
  - Self-organizing migrating algorithm with narrowing search space strategy
- Determine the proposed algorithm position compared to well-known existing algorithms
  - Evaluate the performance on IEEE CEC 2013, 2015, 2017, 2019
  - Assert the ranking against various types of algorithms, even evolutionary algorithms or swarm intelligence
- Apply the proposed algorithms to swarm robotics
  - The movement of mobile robots
  - Unmanned aerial vehicles control

The main contributions of the thesis can be summarized as follows:

- Conduct a comprehensive evaluation on the performance of the SOMA algorithm
- Reach the highest ranking in the history of SOMA on the IEEE CEC Competitions
- Propose an approach to a specific swarm robotics problem based on the swarm intelligence

1.2 Outline of the Thesis

The rest of the thesis is organized into the following sections. Section 2 takes a look at a broad area of research leading to my thesis topic, showing the need to undertake the study. Section 3 describes the principles of the classic self-organizing migrating algorithm as well as its strengths and weaknesses, which underlie the proposed algorithms, including SOMA T3A, Pareto and iSOMA. Section 4 proposes a method to guide the robot to catch the moving target without colliding with any dynamic obstacles based on the proposed algorithms. Section 5 shows the experiment setup for all problems. The results and discussion are presented in Section 6. Finally, the work is concluded in Section 7.

2 BACKGROUND AND MOTIVATION

This section introduces the broad area of research leading to my thesis topic, briefly summarizes previous research to accentuate progress in the field of swarm intelligence and swarm robotics, identifies gaps in knowledge that remain unaddressed, and summarizes what the current study aims to achieve.

The content presented in this section is my scientific research results and co-authors. They have also been originally published in [17, 23, 24, 25].

2.1 Swarm and Evolutionary Computation

Optimization is present in almost all areas of life, from finance to medicine and engineering. Many real-world problems require finding an optimal solution such as finding the shortest trajectory for the travelling salesman problem or finding suitable control parameters to minimize the system's energy consumption, as well as finding global optimal values for the given function. For simple modelled problems, we can apply classical mathematical means to find optimal solutions. However, with complex and constantly changing models, the classical mathematic solver becomes impossible. It challenges researchers to find another way to deal with these practical issues. Thus, the computational optimization was proposed in which the evolutionary algorithm and swarm intelligence were representatives [26].

2.1.1 Evolutionary Algorithm

The evolutionary algorithm (EA) is a common name for optimization algorithms that have mechanisms essentially based on the evolutionary theory of Charles Darwin. From the individuals in the initial population representing the candidate solution of the given problem, through generations of survival, crossover, mutation, and heredity, individuals evolved, changing their genomes to adapt to the environment, helping the population get closer and closer to the optimal solution of that problem [27].

Although implemented in many different forms, those algorithms have a common characteristic of the evolution of organisms, and can be summarized in the following processes:

- Initialization: This process aims to initialize an initial population of individuals that represents candidate solutions to the given problem in the form of the computer code.
- Selection: The main objective of this process is to select one or several individuals to be activated, i.e., individuals will evolve or participate in the evolution of other individuals.
- Crossover: A combination of selected individuals in a given rule to create new ones with the heredity characteristics of the parent - selected individuals. Crossover is one of the two important points of the evolution of creatures as well as evolutionary algorithms. It combines and maintains the good characteristics of organisms from generation to generation.
- Mutation: Mutation is the important rest component, indispensable in the evolution of organisms in nature, as well as in evolutionary algorithms. This process creates "new

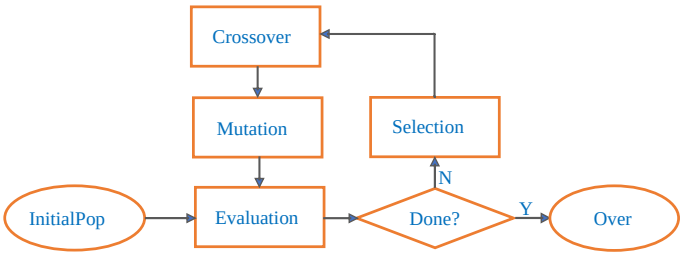


Figure 1: The evolutionary algorithm flowchart.

characteristics" that have not existed in the previous individuals in the whole population. Depending on each algorithm, the mutation process will take place in different ways.

- Evaluation: Over the long evolutionary period, under the influence of harsh environments, weak individuals are eliminated, and healthy individuals continue their survival and growth. In the evolutionary algorithm, this is done by assessing individuals in the population by the given fitness function. Individuals created after the selection, crossover, and mutation processes will be evaluated. Inappropriate individuals that have bad fitness values will be removed from the population and replaced by better individuals.

These processes are repeated until the algorithm finds an optimization value that meets the requirements of the practice, as depicted in Fig. 1.

2.1.2 Swarm Intelligence

The term "Swarm Intelligence" was first introduced by Beni and Wang in the context of cellular robotics system [28]. However, much more sooner, the idea of swarm robots has been introduced (1964) by Stanislaw Lem in his novel *Invictible* [29]. The swarm intelligence algorithm, different from the evolutionary algorithms, is inspired by the cooperation-competition behaviours of intelligent creatures to solve their problems such as foraging, attacking enemies, or defending the nest. These behaviours reflect the action between individuals in a population, or between individuals and the environment, taking place within the same generation. In other words, the mechanism of swarm intelligence algorithm is the interaction between individuals with each other or the environment according to a given rule to find the global optimal solution to the given problem.

Slightly different from the evolutionary algorithms, which operate on Darwin's theory of evolution, individuals in the SI population do not inherit the genetic properties from generation to generation, but rather will share the knowledge with each other in the same generation under loops. This sharing of information is the key for the SI algorithms.

However, there are many similarities between the evolutionary algorithm and swarm intelligence algorithm. They are all population-based algorithms, meaning that one population or some sub-populations need to be initialized at the beginning of algorithm; they select individuals in the population to be activated individuals; they use the fitness function as a basis for selecting and evaluating individuals; they create new ones based on previously selected individuals - but the mechanisms for reproducing new individuals are completely different.

To date, many swarm intelligence algorithms have been proposed in the literature and successfully applied in practice, including function optimization problems, finding optimal routes, scheduling, structural optimization, and image analysis [30]. Examples of swarm intelligence algorithms are: ant colony optimization [31], particle swarm optimization [32], artificial bee colony [33], bacterial foraging [34], firefly algorithm [35], bat algorithm [36], self-organizing migrating algorithm [6], and whale optimization algorithm [37].

2.2 SOMA - A Representative

In recent years, swarm intelligence-based algorithms have been increasingly developed. It is attractive not only because of its simplicity and efficiency, but it can solve complex problems without many complex equations. Some of the most regarded-known algorithms such as PSO, ACO, ABC, and another algorithm that are confirming its position, which is the self-organizing migrating algorithm.

The SOMA was inspired by the competitive-cooperative intelligent behavior of individuals in the population, through many migration loops, to find the global optimal solution of the problem [6, 8]. First introduced in 1999 [38], over two decades of development, SOMA has attracted many researchers and many different variants were proposed, besides the original methods of SOMA ATO and AllToAll. C-SOMGA [39] is an example. It takes advantage of the genetic algorithm and SOMA features to solve constrained nonlinear optimization problems, which works with less than population size and function evaluations. In another publication [40], the authors proposed a hybrid binary coded of GA with real coded SOMA. Its effectiveness has also been affirmed by the set of well-known 25 test problems test. Or CSOMA, a combination of the SOMA approach and Cultural algorithm, presented better results when compared to others algorithm to solving the economic load dispatch problem with the valve-point effect [41]. In addition, other versions of SOMA such as hybridization of self-organizing migrating algorithm with quadratic approximation and non uniform mutation for function optimization [42], Modified Nelder-Mead self organizing migrating algorithm for function optimization and its application [43], Hybrid self-organizing migrating algorithm based on estimation of distribution [44] and Self organizing migrating algorithm with quadratic interpolation for solving large scale global optimization problems [45] have been proposed and demonstrated their effectiveness.

During its development, SOMA has been tested, compared [46], and applied on various practical issues. However, problems arise more and more complicated, not only requires the speed of calculation but also requires accuracy, different versions of SOMA family do not seem to keep up with that development. Therefore, more advanced versions are required to adapt and solve those problems.

These observations have inspired me to research and develop faster, more powerful, and efficient algorithms to solve more emerging complex problems, and apply them to solve a specific field in swarm robotics.

2.3 The Path Planning Problem of Swarm Robots

Avoiding collisions with dynamic obstacles and other robots to catch the given target is one of the most issues in the field of swarm robotics. That is, in an unknown environment where many robots are active, each robot not only catches its moving target but also must be able to avoid each other and avoid dynamic obstacles that the robot does not know about them until the sensors detect them. Various methods have been proposed to solve this problem such as [47], [48], and [49]. Each method has its advantages in some specific environments. There is no doubt that considering the movement of the robot as an optimization problem is one of the most common methods.

In this study, a method of constructing an imaginary map was proposed to deal with that problem, in which the start position, the target, the obstacles, and the robot, in turn, are considered as the highest position, the lowest position, the small hills and the spherical ball. This ball can be moved on the imaginary map due to gravity, corresponding to the movement of the robot in the actual environment. The SOMA algorithm, which acts as gravity on this map, is used to solve the problem.

Moreover, the study proposes a method for the drone to catch the given target and avoid detected obstacles in its path based on the self-organizing migrating algorithm. In particular, a two-component fitness function is proposed based on the principle that the closer the target, the lower the fitness value, and the closer the obstacle, the higher the fitness value. SOMA is used to predict the next positions that the drone will move to. These positions both satisfy the requirement to avoid obstacles and shorten the distance to the target.

3 SELF-ORGANIZING MIGRATING ALGORITHM

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 [6, 7]. The mechanism constituted the SOMA lies in how to select individuals as a Leader and Migrants, how Migrants move toward 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 the initial. The next subsection describes the principle of SOMA, shaping the basis for the analysis of SOMA's strengths and weaknesses.

The content presented in this section is my scientific research results and co-authors. They have also been originally published in [17, 19, 50].

3.1 The Canonical SOMA

At the beginning of the algorithm, a population is generated, containing individuals as candidate solutions to the 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 initial population of the algorithm,
- $x_j^{(lo)}$: the lowest boundary value,
- $x_j^{(hi)}$: the highest boundary value,
- $rand$: uniformly distributed random numbers from 0 to 1.

In each migration loop, an individual with the best fitness value in the population is selected as the Leader, all the remaining individuals will be moving individuals. They will jump step by step toward the Leader using the $Step$ parameter (specified the granularity) until $PathLength$ (a limit of distance) is reached. Instead of jumping directly towards the Leader, another parameter is used to generate perturbation moves, named $PRTVector_j$, causing the individuals to move in the $N - k$ subspace that 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 both helps 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:

- $x_{n,j}^{ML+1}$: the new position in the next migration loop,
- $x_{c,j}^{ML}$: the position in current migration loop,
- $x_{l,j}^{ML}$: the leader position in current migration loop,
- t: jumping 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. It 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 the 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 the algorithm reaches the given stop condition. This strategy is named SOMA AllToOne (SOMA ATO).

Instead of all individuals moving toward the Leader, in another strategy, all individuals move towards each other, regardless of whether it is a better or bad individual. This strategy is called SOMA AllToAll (SOMA ATA).

3.2 The Key Features of SOMA

As described in the previous subsection, after each individual has completed its movement, the best position in this trajectory is selected for comparison with the original. It is clear that to find the better one, the algorithm needs to call a lot of times of function evaluations (FEs). For example, with the standard setting of SOMA: $Pathlength = 3.0$ and $Step = 0.11$, each traveling individual has to take 27 times of jumps to complete its movement, which means the algorithm spends 27 times of FEs to find the better position. Meanwhile, other algorithms only use one time of FE to improve their candidate solutions such as DE, PSO, and ABC. This characteristic causes the algorithm to soon face the stop condition of $MaxFEs$ and the premature convergence scenario before it searches for detail in the exploitation phase at the end of the optimization process.

On the other side, in each variable of the optimization problem, $PRTVector_j$ accepts only one of the two values of 0 and 1. Thus, this variable will be updated with a multiple of $Step$ if the value of $PRTVector_j$ is 1, and vice versa it will remain its position. Geometrically, this means that traveling individuals will move on intersection edges of hyperplanes created by pairs of sides of variables, as shown in Fig. 2 and Fig. 3 (setting parameters: $Step = 0.33$ and $PathLength = 3.0$).

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 potential search space.

Besides, one of the major weaknesses of the canonical SOMA is that nonsense moves are taken from better individuals to the worse one, as shown in Fig. 4. This leads to a waste of

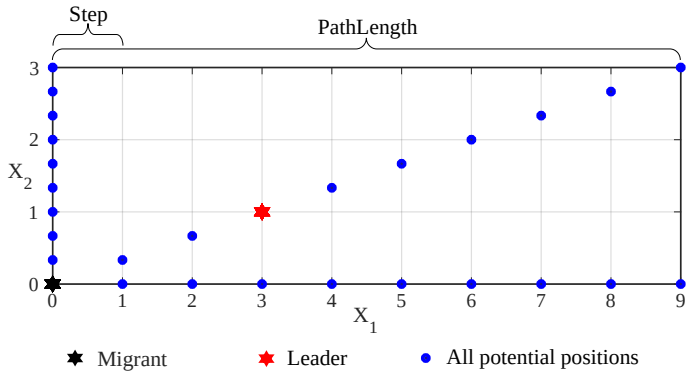


Figure 2: All possible positions of the offspring in 2D space.

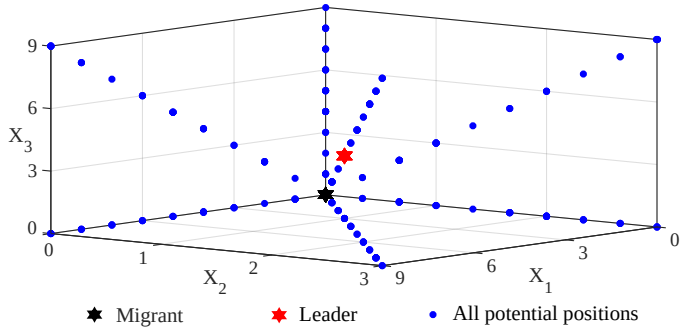


Figure 3: All possible positions of the offspring in 3D space.

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. In this case, there is no position with better fitness value than the migrant itself in the capability search space.

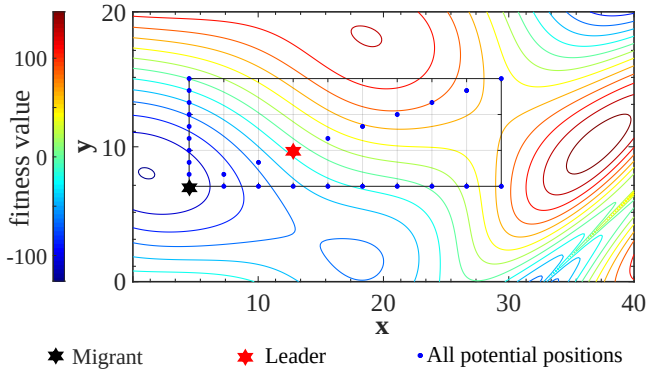


Figure 4: The meaningless move from the migrant to the leader having lower fitness value.

3.3 SOMA T3A

The name "Self-organizing migrating algorithm" fully covers the operating processes of the algorithm. Accordingly, the proposed algorithm is divided into processes, which are named the

initialization process, organization process, migration process, and update process. They operate in loops called migration loops. In each migration loop, individuals in the initial population will move towards the other to explore promising subspaces and then exploit these spaces to find the global optimum solution. These processes are repeated until the given stop conditions are satisfied, as described in Fig. 5.

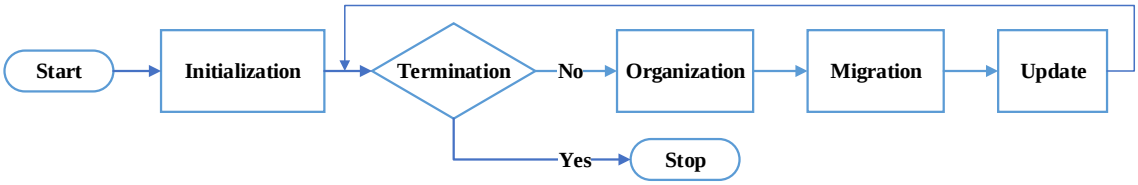


Figure 5: The flowchart of the algorithm.

Depending on how the above processes work, different versions of SOMA are developed, which will be presented in the next subsections.

In SOMA Team To Team Adaptive (SOMA T3A) [17], the initial population was generated according to Eq. 1 and evaluated by the given fitness function, similar to the original version of SOMA. The proposed algorithm then enters the first migration loop.

3.3.1 Organization

The two tasks of this process are to choose which individuals will move, named Migrants, and which individual will be the Leader. To determine Migrants, the algorithm randomly selects m individuals from the population and then selects the best n out of m individuals ($n \leqslant m$). The n individuals are Migrants. To choose the Leader, k individuals are randomly selected from the population, and then the best of the k individuals becomes the Leader. Fig. 6 describes this process. In case the Leader is one of the Migrants, the moving of this coincident Migrant is skipped.

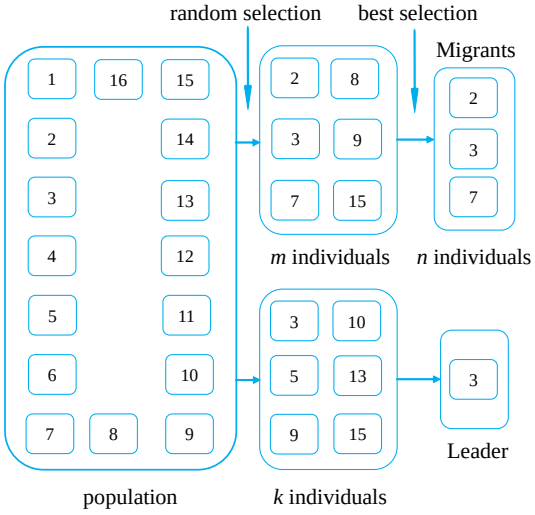


Figure 6: The organization process of SOMA T3A.

The organization process plays an important role in exploring a promising subspace. The larger the values of m, n, and k, the smaller the search subspace. That is, the algorithm only focuses on the best group of individuals in the population. This seriously affects the diversity of populations when the algorithm goes through migration loops. For simple functions and a small number of dimensions, the global minimum position is found quickly and accurately. However, for complex functions and large dimensions, the algorithm can be trapped in the local minima, and no longer able to go beyond the local search subspace. In other words, the algorithm tends to exploitation phase. In the case of m, n, and k equal to the population size, the algorithm will return to its original version, which is SOMA ATO.

On the other hand, if the values of m, n, and k are smaller, the algorithm will search in a large space in a rambling and inefficient way. These parameters determine the balance of the two phases of exploration and exploitation, besides the PRT parameter.

3.3.2 Migration

Migration is the process of moving Migrant towards the Leader. In the original version, individuals jump towards the Leader with fixed steps until a given $Pathlength$ is reached. Instead of jumping straight to the Leader, individuals move in the $N - k$ dimensional subspace, which is perpendicular to the original space, with the probability of each jump is decided by the fixed PRT parameter. In this version, we propose adaptive PRT and $Step$ parameters.

PRT is a sensitive parameter of the algorithm. The closer to 1 the value of PRT, the more individuals tend to jump straight towards the Leader. This leads to the diversity of populations being not maintained. On the other hand, if the PRT is too small, the algorithm will be inclined to the exploration phase. Therefore, PRT should be started with a low value, and incremented by the number of migration loops. Such adjustable calculation of the PRT is given by Eq. 4.

$$PRT = 0.05 + 0.90 \ \frac{FEs}{MaxFEs} \tag{4}$$

where:

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

To find a promising subspace, large steps are recommended to speed up the algorithm and reduce the number of function evaluations. When focusing on a promising subspace, i.e., exploitation phase, small steps are recommended to apply. For that reason, descending steps by the number of migration loops were proposed as in Eq.5.

$$Step = 0.15 - 0.08 \ \frac{FEs}{MaxFEs} \tag{5}$$

In this version, the number of jumps is fixed instead of fixing the $PathLength$ of the original version. That is each individual moves towards the Leader with a given number of jumps N_{jumps}, and it is then evaluated by the fitness function.

3.3.3 Update

The best position on the way of the Migrant is chosen to compare with the original position. If it is better, it will replace the original position of this migrant.

3.4 SOMA Pareto

In the beginning, a population containing candidate solutions to the problem is generated according to Eq. 1, similar to other versions of SOMA. The given fitness function then evaluates the population and the algorithm goes into the migration loop with main processes, namely the organization, migration, and update process [19].

3.4.1 The Organization Process

The organization process determines which individuals in the population will interact with each other to create new candidate solutions that are different from the initial individuals. In other words, it is the process of selecting the Migrant and the Leader, and this Migrant will then move towards the Leader to search the better positions during the move.

This process plays an essential role in exploring promising subspaces and then focusing on exploiting these promising subspaces. Therefore, an individual selected to become the Leader is an individual with good fitness value, but should not always be the best. Because if it is the best, the population will only focus on exploiting the subspace around the best Leader, ignoring other promising subspaces around other good individuals, which may contain global minima. And so the algorithm will converge quickly in local minima. The Migrant should be an individual with the worst fitness value than the Leader's fitness value to avoid the meaningless moves, as pointed out in Fig. 4.

To choose the Migrant and Leader as analyzed, we propose to apply the Pareto Principle [51]. Accordingly, 20% of the number of individuals (marked as A) hold 80% of the population value, and 80% of the number of individuals (marked as B) hold 20% of the population value only. That means the Leader should be in A and the Migrant should be in B. However, not all individuals in B are Migrants, and not all individuals in A are Leaders. Instead, we propose to randomly select an individual in 20% of A (marked as C) as the Leader and randomly select an individual in 20% of B (marked as D) as the Migrant.

In other words, in each loop, the population will be sorted in increasing order of fitness value. And the population is then divided into two parts, the first part is A containing 20% of the number of individuals with the best fitness value, and part B containing 80% of the remaining individuals. After that, the best 20% of A will be selected and marked as C, and the best 20% of B will be selected and marked as D. And then, the algorithm randomly selects an individual from C to be the Leader and randomly selects an individual from D to be the Migrant. The entire process is described in Fig. 7.

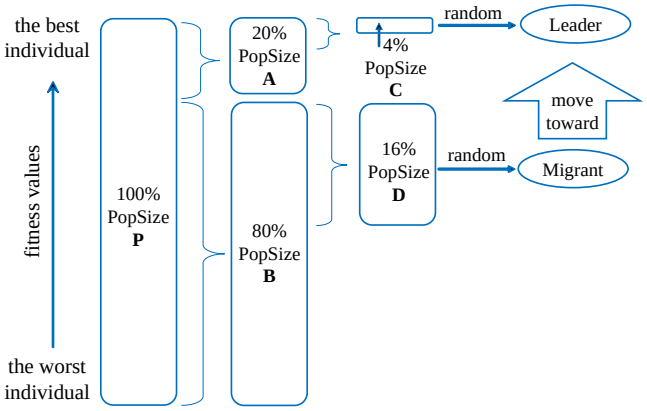


Figure 7: The organization process of SOMA Pareto.

3.4.2 The Migration Process

The migration process searches for a better position by moving the Migrant towards the selected Leader. In the original version, individuals move in the $N - k$ dimensional subspace with the given fixed step until reaching the given $PathLength$. The ratio of each move is determined by the PRT. Using fixed parameters of $Step$, $PathLength$ and PRT lead to limiting the flexibility of the algorithm. Instead, it is better to start by exploring promising subspaces and then focus on exploiting these promising subspaces. That has been overcome in the SOMA T3A version [17], which is to use the adaptive PRT and $Step$ parameters and use the fixed number of jumps (N_{jump}) instead of the $PathLength$, but not yet thorough because $PRTVector$ still only is the fixed value, 0 or 1.

In this version, we propose the adaptive PRT and $Step$ as given in Eq. 6 and Eq. 7, and shown in Fig. 8. These parameters help maintain the diversity of the population, balance between two phases of exploration and exploitation, helping individuals move far away from the trapped area. In this case, $PathLength = Step \times N_{jump}$.

$$PRT = 0.50 + 0.45\ cos(T_1\pi\ \frac{FEs}{MaxFEs} + \pi) \tag{6}$$

$$Step = 0.35 + 0.15\ cos(T_2\pi\ \frac{FEs}{MaxFEs}) \tag{7}$$

Not only that, we propose to change the $PRTVector$ parameter. It not only receives two values 0 and 1 like all previous versions but also adapts to each migration loop. It means that the Migrant instead of moving in a $N - k$ dimensional subspace that is perpendicular to the original space, which will move in a narrow $N - k$ dimensional subspace gradually with each migration loop to focus on the Leader. $PRTVector$ is given in Eq. 8.

$$if\ rand_j < PRT;\ PRTVector_j = 1;\ \ else,\ PRTVector_j = \frac{FEs}{MaxFEs}. \tag{8}$$

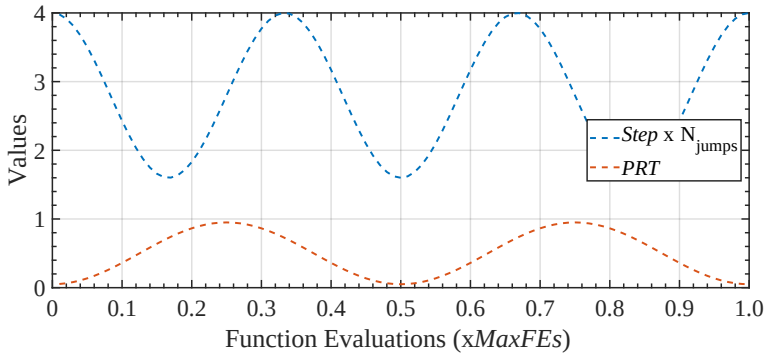


Figure 8: The control parameters of SOMA Pareto - An example.

3.4.3 The Update Process

The update process is the process of reviewing and deciding whether to replace the initial Migrant. To do that, all jumping positions are evaluated by the fitness function and then one of the best positions is selected to compare with the initial position of that Migrant. If the new position is better, it will replace the initial position and become a new individual in the population. On the contrary, it is ignored, and that migration loop has no improvement.

3.5 iSOMA

3.5.1 The Initialization Process

The iSOMA starts with the initialization process. Within the control parameters established, an initial population containing candidate solutions to the problem 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 below.

3.5.2 The Self-Organizing Process

The self-organizing process in the proposed algorithm is the process of determining which individuals will move towards their targets (named as Migrants) and which one will become the 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 analysed in the previous subsection. On the other hand, if all individuals move towards 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, similar to the organization process of SOMA T3A.

For problems containing many local minimum 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 on the performance of the algorithm, besides the other control parameters will be presented below.

3.5.3 The Migrating Process

The migrating process regulates how the Migrant moves towards 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 Fig. 2 and Fig. 3. To enhance the algorithm's search capabilities and restrict the mentioned weakness, we propose the following improvements for the migrating way.

3.5.4 The Order of Jumps

Instead of jumping gradually towards the Leader as in the canonical version, we propose a method of jumping in order, as shown in Fig. 9 (setting parameters: $Step = 0.3$, $N_{jump} = 10$, and $PathLength = 3.0$.). 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.

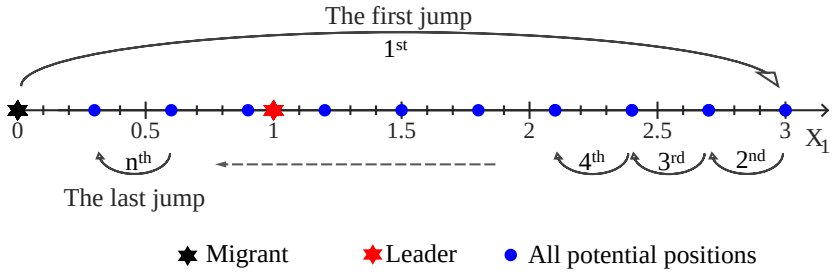


Figure 9: The order of the jumps in the iSOMA.

3.5.5 Immediately Update

Another valuable improvement derives from terminating the jumping progress of the current Migrant and immediately update its position to the population if the new 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.

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.

3.5.6 Narrow Search Space

Fig. 10 depicts the narrowing of the search space (with $Step = 0.33$, $PathLength = 3.0$, and adaptive $PRTVector$). In the early stages of the optimization progress, the algorithm should prefer to explore promising subspace rather than focus on exploiting them. So, 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).

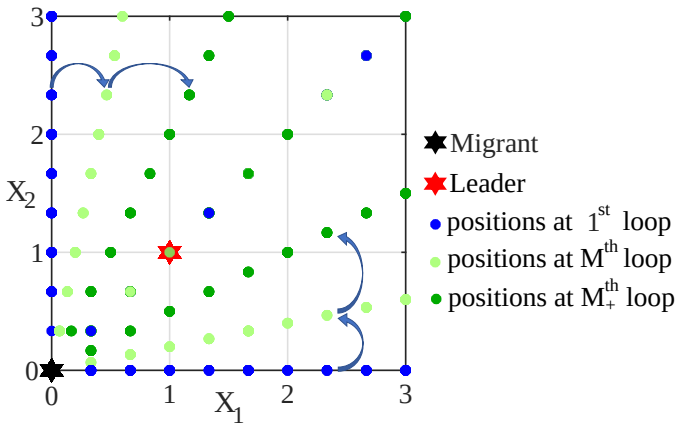


Figure 10: All possible positions of the offspring over migration loops.

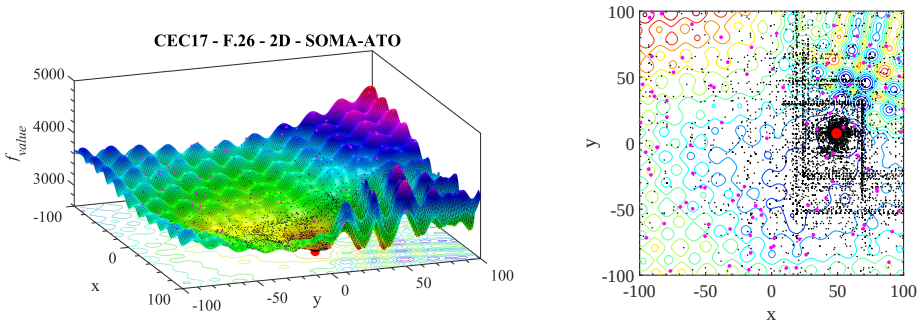
Towards the end of optimization progress, the iSOMA is more inclined to exploit these promising subspace. Therefore, the adaptive $PRTVector$ parameter is proposed so that individuals can move inside the space created by intersection hyperplanes, instead of just moving on the edges like the canonical version. Eq. 8 is used to enable this feature.

Besides, the adaptive PRT parameter is leveraged in the iSOMA, which was introduced in [17], given in Eq. 4. In this version, the $Step$ parameter is fixed.

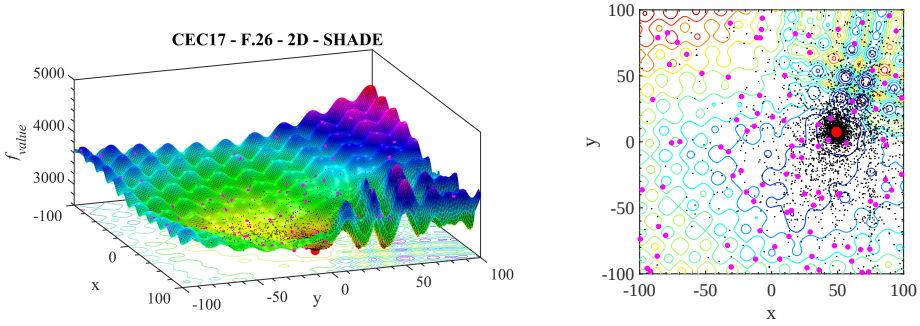
3.5.7 The Replacement Process

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 measured by the number of function evaluations.

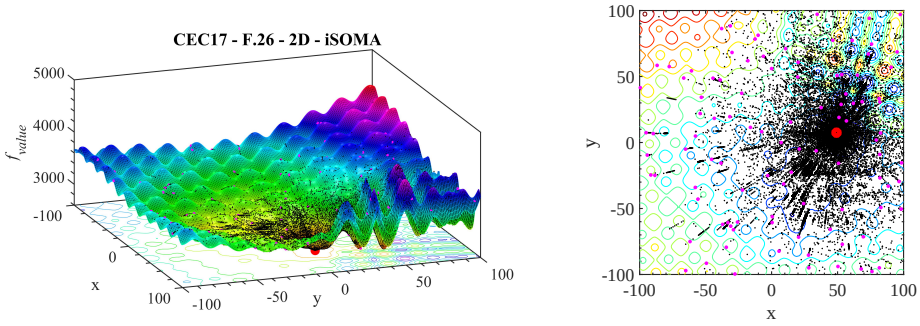
Accordingly, after several FEs, if the algorithm does not discover a better position than the global best, the iSOMA will randomly replace some (10% for example) of the existing individuals in the curent population (excluding the global best individual) by the same number of randomly generated individuals 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 falling into the current local traps.



(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.

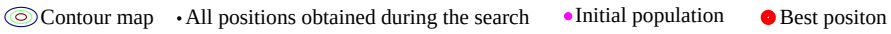


Figure 11: The operation of three algorithms SOMA ATO, SHADE, and iSOMA on the function 26^{th} of CEC17 tested on 2D.

Fig. 11 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" towards the best individual.

4 APPLICATIONS IN SWARM ROBOTICS

One of the most important issues for swarm robotics applications is catching up with moving targets and avoiding multiple dynamic obstacles. It is complicated in that it requires the algorithm to work in real-time to avoid obstacles that are standing or moving in an unknown environment where the robot does not know their position until detecting them by sensors arranged on robot.

Besides the long-standing methods such as potential field method [52], and the vector field histogram [53], several new methods such as follow the gap method [54], and barrier function [55], or artificial intelligence methods such as genetic algorithm [56], and neural network [57] also demonstrate their effectiveness. Among the methods of artificial intelligence used to solve the problem as a function optimization problem, SOMA emerges as a fast, powerful, and efficient algorithm [6, 7, 8].

In this section, the author applies a method to guide the robot to catch the moving target without colliding with any dynamic obstacles based on swarm intelligence algorithms.

The content presented in this section is my scientific research results and co-authors. They have also been originally published in [24, 25, 58, 59, 60].

4.1 Swarm Robotics: Mobile Robots

4.1.1 The Movement of Mobile Robots

The final goal of the robot is catching the moving target without colliding with any dynamic obstacles in the unknown environment. The assumptions outlined below ensure that the robot can work well under certain circumstances to ensure that the goal is achieved.

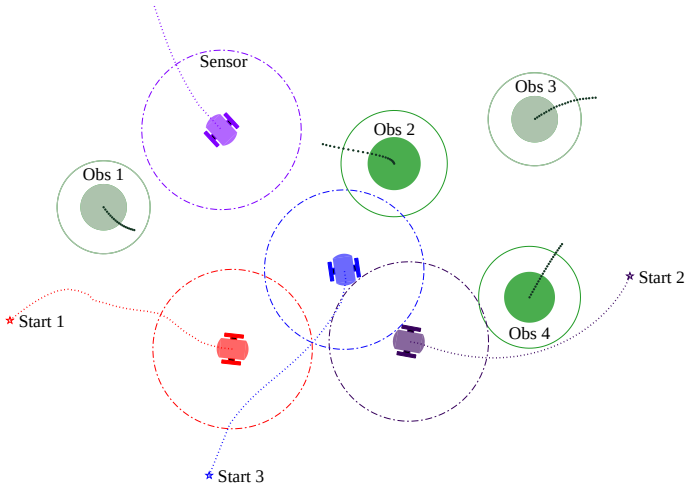


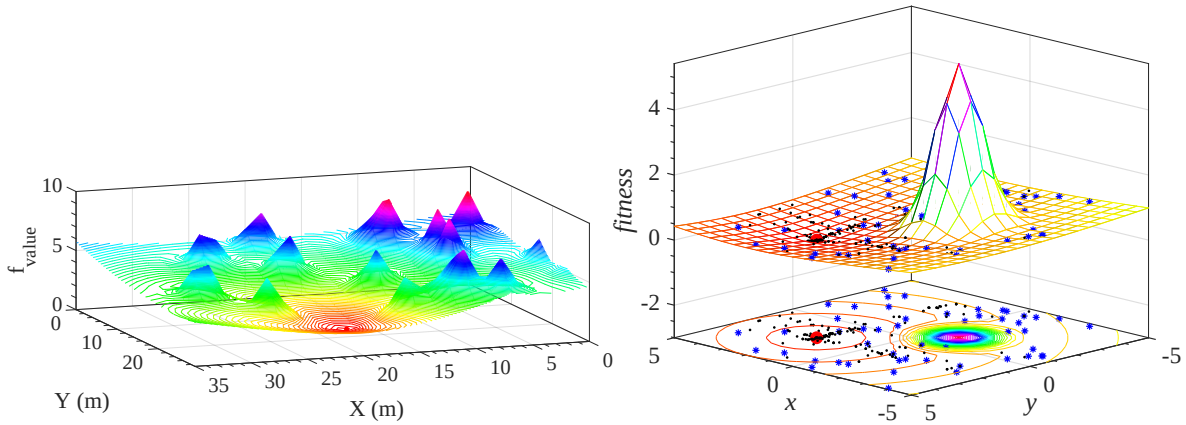
Figure 12: The robots and obstacles model.

- *Robot*: For simplicity, all the physical dimensions of the robot are enclosed by a circle of radius r_{robot}. The sensors on the robot can accurately measure the distance to the obstacles within the sensor range, see Fig. 12. The robot is offered about the position of the target and can be controlled to reach the furthest desired position called moving step, i.e., the

robot moves from the current position to a given furthest position without any difficulty. Moving step is a given parameter depending on the physical structure of the robot.

- *Obstacles*: Obstacles are considered as circles of radius r_{obs}. The obstacles can move at any speed and the robot does not know about the obstacles position until they are detected by the sensors.
- *Velocity*: The velocity of the robot must be greater than the velocity of each obstacle because if the opposite happens, the robot will not be able to avoid these dynamic obstacles. Similarly, to catch the target, the robot must move with the velocity greater than the target's velocity.

A method that can be visualized as *high mountain flowing water* is proposed to solve the issue. In this method, an imaginary map is built, in which the starting position of the robot is considered as the top of the high mountain, the target considered as the lowest lying land, the obstacles considered as small hills, and the robot considered as water, flowing from the top of the mountain to the lowest lying land, flowing around small hills without flowing back into them as a natural law, see Fig. 13. When the target and obstacles move, the small hills and the lowest lying land also move respectively.



(a) The imaginary map

(b) Migration of individuals in a single obstacle map

Figure 13: The imaginary environment for swarm robot.

To build this map, a mathematical model will be shown in the next subsection.

4.1.2 Imaginary Map: The Fitness Function

For any optimization problem, the fitness function is an important component, which is the object to solve. In some situations, the fitness function is already given. However, in some cases, we have to build the fitness function by modeling that problem. In this section, we present how to turn the robot path planning problem into a fitness function.

Starting with a simple rule, the robot is as close to the target and as far away from the obstacles as possible. Eq. 9 generally describes the elements X stated, where n is the number of target and obstacles detected by the sensors. The goal is to minimize the function $f_{(X)}$.

$$f_{(X)} = \sum_{i=1}^{n} X_i \tag{9}$$

For the principle as close to the target as possible, we see that the value of the function $f_{(X)}$ should be proportional to the distance from the robot to the target. Eq. 10 constructs the first element of the fitness function in detail, where (x_{robot}, y_{robot}) and (x_{target}, y_{target}) are the current positions of the robot and target respectively, and a_1 is the equilibrium coefficient.

$$X_1 = a_1 \sqrt{(x_{target} - x_{robot})^2 + (y_{target} - y_{robot})^2} \tag{10}$$

Similarly, with the rule that as far as possible from obstacles, the value of the $f_{(X)}$ function should be inversely proportional to the distance from the robot to detected obstacles. Eq. 11 describes this in detail.

$$X_i = a_i \sum_{0}^{n_{obstacle}} e^{-(c - r_{obstacle})\, dis_{obstacle}} \tag{11}$$

where:

$$dis_{obstacle} = \sqrt{(x_{obstacle} - x_{robot})^2 + (y_{obstacle} - y_{robot})^2}$$

- X_i: the obstacle elements of the $f_{(X)}$,
- a_i: the equilibrium coefficient,
- $n_{obstacle}$: the number of detected obstacles,
- c: the influential coefficient of obstacles,
- $r_{obstacle}$: the radius of detected obstacles,
- $dis_{obstacle}$: the distance from the robot to detected obstacles.

In the framework of this study, I do not go into details about robot kinematics and dynamics. I assume that the robot can move smoothly from point A to a nearby point without any problem, and the SOMA will generate the dynamic set of that points [58].

Due to the robot's physical limitations, the maximum distance between the two points mentioned is limited, named d_{limit}. However, no matter how big the d_{limit} is, the algorithm quality is completely independent of this distance.

4.2 Unmanned Aerial Vehicles Control: Drones

The control of unmanned aerial vehicles (UAVs) is often more complicated, requiring not only the speed of the algorithm, but also real-time accuracy. Many solutions have been proposed and successfully applied to path planning for UAVs such as sampling-based path planning for UAV collision avoidance [61], cooperative path planning with applications to target tracking and obstacle avoidance for multi-UAVs [62], and grid-based coverage path planning with minimum

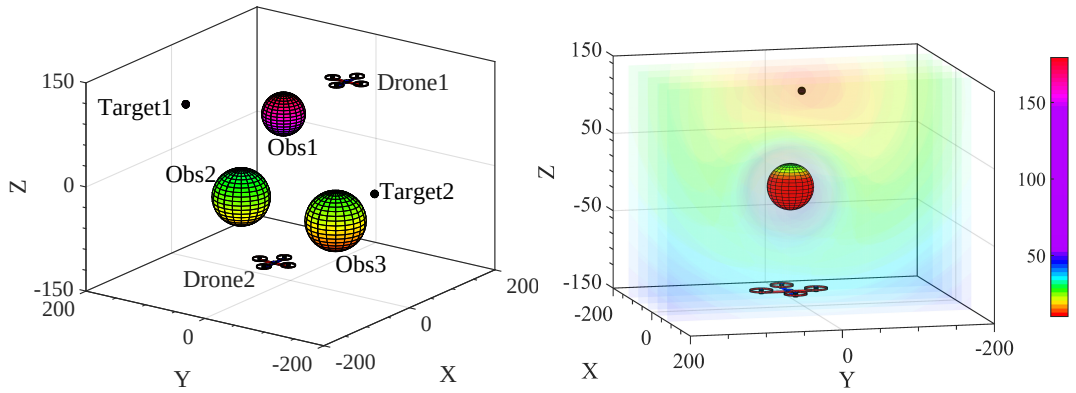
energy over irregular-shaped areas with UAVs [63]. However, the application of SOMA to plan the trajectory of the drone is unprecedented.

In this subsection, I propose the application of the SOMA to generate the trajectory for the drone, avoid detected obstacles and catch the given target.

The primary goal of the drone is to move toward and hit the given target without any collision with obstacles along the way. To accomplish this, the following assumptions are needed.

- *Obstacles*: Obstacles come in many different shapes and sizes in nature. However, within the framework of this study, we assume that the entire physical size of the obstacle is surrounded by a sphere of radius $r_{obstacle}$. These obstacles do not move in space and they will be detected and located by the sensors fitted on the drone.
- *Drone*: In this study, we assume that the drone is capable of freely moving through space, being able to fly from point A to point B near A without any problem, called the moving step $r_{movingstep}$. Depending on the size, structure, and controller, each drone has a different moving step. Besides, the drone is equipped with a sensor system to detect and identify obstacles within its operating range with radius r_{detect}. The target position is given before and provided to the drone.

The operating model of drones and obstacles is shown in Fig. 14.



(a) The drones and obstacles model (b) The drone model in 4-dimensional space

Figure 14: The drone model in 3D and 4D.

Now, it is conceivable that the task of the algorithm is to find all B positions consecutively from start position A, forming a set of points that the drone will have to move through. The next B position must both shorten the distance of the drone to the target while avoiding collisions with obstacles. To formulate this idea, I propose an equation containing two components: the first component acts as a magnet to pull the drone towards the target and the second component pushes the drone away from the obstacles. This equation is proposed in Eq. 12.

$$f_{value} = a_1 * e^{a_2 * dis_{tar}^{a_3}} + \sum_{n=0}^{n_{obs}} b_1 * e^{b_2 * dis_{obs}^{b_3}} \tag{12}$$

where:

- f_{value} : the fitness value,

- n_{obs} : the number of detected obstacles,
- dis_{tar} : the distance from the drone to the given target, in Eq. 13,
- dis_{obs} : the distance from the drone to each detected obstacle, in Eq. 14,
- a_x, b_x : the equilibrium coefficients ($x = 1, 2, 3.$).

$$dis_{tar} = \sqrt{(x_{target} - x_{drone})^2 + (y_{target} - y_{drone})^2 + (z_{target} - z_{drone})^2} \tag{13}$$

$$dis_{obs} = \sqrt{(x_{obs} - x_{drone})^2 + (y_{obs} - y_{drone})^2 + (z_{obs} - z_{drone})^2} \tag{14}$$

The problem of catching the target and avoiding obstacles for the drone has now become an optimization problem. Accordingly, generating the trajectory of the drone is to find the optimal solution for Eq. 12.

Fig. 14b depicts drone activity in 4-dimensional space. The color represents the fitness space. This space will change as the drone changes its position. Determining a set of points as mentioned will be executed by SOMA.

5 EXPERIMENT DESIGN

5.1 Computational Setup

5.1.1 Test Functions

To thoroughly evaluate the performance of the proposed algorithms, three common test suites of the IEEE Congress on Evolutionary Computation (IEEE CEC) were used, including a total of 73 functions as listed below:

- The first suite consists of 28 functions from the IEEE CEC 2013 Special Session on Real Parameter Single Objective Optimization, consisting of 28 functions (CEC13, detail: [64]);
- The second suite consists of 15 functions from the IEEE CEC 2015 Competition on Learning-based Real Parameter Single Objective Optimization (CEC15, see detail at [65]);
- And the last suite consists of 30 functions from the IEEE CEC 2017 Special Session and Competition on Single Objective Real Parameter Numerical Optimization (CEC17, [66]).

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 [64, 65, 66].

Table 1: The 100-Digit Challenge Basic Test Functions.

No.	Functions	D	Search range
1	Storn's Chebyshev Polynomial Fitting Problem	9	[-8192,8192]
2	Inverse Hilbert Matrix Problem	16	[-16384,16384]
3	Lennard-Jones Minimum Energy Cluster	18	[-4,4]
4	Rastrigin's Function	10	[-100,100]
5	Griewangk's Function	10	[-100,100]
6	Weierstrass Function	10	[-100,100]
7	Modified Schwefel's Function	10	[-100,100]
8	Expanded Schaffer's F6 Function	10	[-100,100]
9	Happy Cat Function	10	[-100,100]
10	Ackley Function	10	[-100,100]

In addition to those benchmark suites, the proposed algorithms were used to attend the 100-Digit Challenge Competition [67], known as the CEC 2019. The competition provides 10 hard functions, each participant is allowed to use an algorithm to solve (optimize) those functions with an accuracy of 10 digits each, and 10 points are awarded for each completely solved function. Tab. 1 lists the basic functions used in the 100-Digit Challenge. All test functions are scalable and they were designed to have the same global minimum value of 1.0 within the search range. The dimensions and search range are also in the two last columns in this table.

Different from previous CEC competitions, the year's competition highlights the accuracy rather than the time to solve. Therefore, the algorithm is not limited to function evaluations and is allowed to adjust 2 control parameters in the same way for all 10 functions. This competition is a great challenge for cutting-edge algorithms, see [18, 20, 68] for more detailed.

5.1.2 Comparison Algorithms and Control Parameters

To demonstrate improvement over previous versions, the results were compared to the original and latest versions of SOMA ATO and ATA [6, 7].

In order to investigate the proposed algorithms 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 the proposed SOMAs 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 proposed SOMA on the optimization algorithm map.

For SOMA T3A:

- Artificial bee colony (ABC) [69, 70, 71],
- Hybrid firefly and particle swarm optimization (HFPSO) [72],
- Salp swarm algorithm (SSA) [73],
- Self-adaptive multi population elitist Jaya (SAMPE Jaya) [74].

The control parameter values of these algorithms have been used from the original papers in the citations without any changes except the fitness functions and termination.

The dimension of all test problems was set at $30D$ with the same search range $[-100, 100]^D$ and $MaxFEs = 300000$. For each function, each algorithm was independently repeated 51 times. The error value will be taken as 0 if it is smaller than 10^{-8}. The Wilcoxon rank-sum test was applied at the 5% significance level to evaluate whether the differences between the results are significant [75, 76].

The settings of SOMA T3A for all benchmark problems: $PopSize = 30$, $N_{jump} = 45$, $n = 4$, $m = 10$, $k = 10$.

For SOMA Pareto:

- On the CEC'13:
  - Super-fit Multicriteria Adaptive Differential Evolution [77] (SMASE);
  - A CMA-ES Super-fit Scheme for the Re-sampled Inheritance Search [78] (CMAES-RIS);
  - Differential Evolution with Automatic Parameter Configuration [79] (DE-APC);
  - A Self-adaptive Heterogeneous Particle Swarm Optimization [80] (fk-PSO).
- On the CEC'15:
  - Tuning Maturity Model of Ecogeography-Based Optimization [81] (TEBO);
  - An Improved Covariance Matrix Leaning and Searching Preference Algorithm [82] (ICMLSP);

  - A Self-adaptive Dynamic Particle Swarm Optimizer [83] (SaDPSO);
  - Dynamic Search Fireworks Algorithm with Covariance Mutation [84] (dynFWACM).
- On the CEC'17:
  - Self-Organizing Migrating Algorithm Original [6] (SOMA AllToOne and AllToAll);
  - Self-Organizing Migrating Algorithm Team to Team Adaptive [17] (SOMA T3A);
  - A Version of IPOP-CMA-ES Algorithm with Midpoint [85] (RB-IPOP-CMA-ES);
  - Proactive Particles in Swarm Optimization: a Settings-Free Algorithm [86] (PPSO);
  - Teaching Learning Based Optimization with Focused Learning and its Performance [87] (TLBO-FL);
  - Self-Adaptive Multi Population Elitist Jaya [88] (SAMPE-Jaya).

In the above algorithms, there are 11 algorithms that have participated in the CEC Competition in the corresponding years, except for SAMPE-Jaya and SOMA. Control parameter settings of these algorithms have been used the same as the original papers in the citations without any changes. For SOMA ATO and ATA, $PopSize = 30$ and $PopSize = 20$ respectively; $Step = 0.11$; $PRT = 0.1$; $PathLength = 3$.

The dimension of all 73 test functions was set at $30D$ with the same search range $[-100, 100]^D$ and $MaxFEs = 10000 * D$. For each function, each algorithm was independently repeated 51 times. Error values will be taken as 0 if it is smaller than 10^{-8}, as experimental settings requested in [64], [65], and [66]. The Wilcoxon rank-sum test was applied at the 5% significance level to evaluate whether the differences between the results are significant [75, 76].

SOMA Pareto setting: $PopSize = 100$, $N_{jump} = 10$, PRT and $Step$ as in Eq. 6 and Eq. 7 with $T_1 = T_2 = 1$ for all problems.

For iSOMA:

- IEEE CEC 2013 (CEC13):
  - Evaluating the performance of SHADE on CEC 2013 benchmark problems [89] (SHADE);
  - Super-fit multicriteria adaptive differential evolution [77] (SMADE);
  - A CMA-ES super-fit scheme for the re-sampled inheritance search [90] (CMAES-RIS);
  - Testing a particle swarm optimization and artificial bee colony hybrid algorithm on the CEC13 benchmarks [91] (SPSOABC);
  - A genetic algorithm for solving the CEC'2013 competition problems on real-parameter optimization [92] (TPC-GA).
- IEEE CEC 2015 (CEC15):
  - A differential evolution algorithm with success-based parameter adaptation for CEC2015 learning-based optimization [93] (DEsPA);
  - Tuning maturity model of ecogeography-based optimization on CEC 2015 single-objective optimization test problems [81] (TEBO);
  - A Self-adaptive Dynamic Particle Swarm Optimizer [83] (SaDPSO);
  - An improved covariance matrix leaning and searching preference algorithm for solving CEC 2015 benchmark problems [82] (ICMLSP);

  - Dynamic search fireworks algorithm with covariance mutation for solving the CEC 2015 learning based competition problems [84] (dynFWACM).
- IEEE CEC 2017 (CEC17):
  - A differential evolution strategy [94] (DES);
  - A version of IPOP-CMA-ES algorithm with midpoint for CEC 2017 single objective bound constrained problems [95] (RB-IPOP-CMA-ES);
  - Proactive particles in swarm optimization: A settings-free algorithm for real-parameter single objective optimization problems [96] (PPSO);
  - Dynamic yin-yang pair optimization and its performance on single objective real parameter problems of cec 2017 [97] (DYYPO);
  - Teaching learning based optimization with focused learning and its performance on CEC2017 functions [98] (TLBO-FL).

Tests on $10D$ and $30D$ are carried out, with a search range of $[-100, 100]^D$ for all problems. The maximum number of function evaluations was set at $10000 * D$ ($MaxFEs$ for $10D = 100000$; for $30D = 300000$). Error value smaller than 10^{-8} will be taken as zero. For each function, each algorithm was independently repeated 51 times, as experimental settings requested in [64, 65, 66]. The Wilcoxon rank-sum test was applied at the 5% significance level to evaluate whether the differences between the results are significant [75, 76].

The control parameter settings of iSOMA for all problems: $PopSize = 100$, $N_{jump} = 10$, $n = 5$, $m = 10$, $k = 15$, $Step = 0.3$, and PRT as in Eq. 4. Other detailed SOMA parameters are found in cited publications.

The control parameter settings of the rest algorithms have been used the same as the original papers in the citations without any changes.

5.2 Simulation Setup

5.2.1 Mobile Robot Maps

5.2.1.1 Selective scenarios To rigorously evaluate the feasibility of the proposed solution, we built 4 selective scenarios, covering most of basic situations that can occur in the real-world.

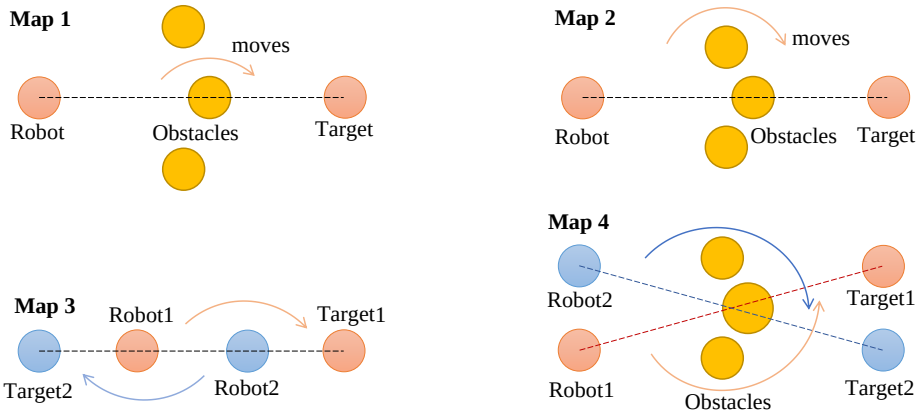


Figure 15: Selective scenarios to test the operability of robots.

The first scenario is the simplest one, having a robot (with a respective target) and three static obstacles. The location of the obstacles is intentionally arranged so that they are symmetrical and centered on the line connecting the robot to the target. The gap between the three obstacles is calculated wide enough for the robot to move through them. This scenario is set up to test the ability of the robot to pass through sufficient gaps between obstacles (see Map 1 of Fig. 15).

The second scenario is similar to the first but the distance between the obstacles has been changed so that they are smaller than the physical size of the robot (it cannot move through those gaps). The aim is to trap the robot into the local minima and observe how to escape from the trapped area of the robot (Map 2 of Fig. 15).

In the third scenario, two robots with two respective targets were set. There is no obstacle in this map, but robots will be obstacles to each other. All of them were intentionally put in a straight line so that the robots will move in opposite directions. This situation tests the possibility of mutual avoidance between robots (Map 3 of Fig. 15).

The last one is the most complex scenario. Two robots, two respective targets, and three obstacles were used. The robots are on the same side of the obstacles, and the targets are on the opposite side but diagonally. The obstacles located in the middle are not only to prevent the movement of the robots, but also trap the robot to the local minimum associated with the other robot. This scenario tests the generality of the proposed algorithm (Map 4 of Fig. 15).

Table 2: Locations of obstacles and robots in Cartesian coordinates - Map 1 and 2 (in *meter*)

The object	Obs1	Obs2	Obs3	Ro1	Tar1
x_{map_1}	0.4	1.1	1.3	0.1	2.0
y_{map_1}	1.1	0.4	1.3	0.1	2.0
r_{map_1}	0.3	0.3	0.3	-	-
x_{map_2}	0.6	1.3	1.1	0.1	2.0
y_{map_2}	1.3	0.6	1.1	0.1	2.0
r_{map_2}	0.2	0.2	0.2	-	-

Table 3: Locations of robots in Cartesian coordinates - Map 3 (in *meter*)

The object	Ro1	Ro2	Tar1	Tar2
x_{map_3}	0.6	1.7	2.0	0.3
y_{map_3}	0.6	1.7	2.0	0.3

Table 4: Locations of obstacles and robots in Cartesian coordinates - Map 4 (in *meter*)

The object	Obs1	Obs2	Obs3	Ro1	Ro2	Tar1	Tar2
x_{map_4}	0.7	1.4	1.1	0.5	0.2	1.7	2.2
y_{map_4}	1.4	0.7	1.1	0.2	0.5	2.2	1.7
r_{map_4}	0.2	0.2	0.3	-	-	-	-

The detailed locations of robots, obstacles, and targets are shown in Tabs. 2, 3, and 4.

5.2.1.2 Setup for unknown complex environment The starting positions of all robots and targets in this map are given in Tab. 5. The sizes and positions of the obstacles are given in Tab. 6. The robots used in all simulations have a radius $r_{robot} = 0.08m$. The robot sensors have an active range with a radius $r_{sensor} = 0.36m$. The moving step of the robot is $r_{movingstep} = 0.05m$.

Table 5: The starting positions and the target positions of the robots.

Map		5^{th}					6^{th}			
Position		Ro1	Ro2	Ro3	Ro4	Ro5	Ro1	Ro2	Ro3	Ro4
Start	$x\ (m)$	1.6	0.5	0.2	3.2	0.7	0.1	2.9	1.0	0.5
	$y\ (m)$	2.3	0.3	1.4	0.5	2.3	0.7	0.9	0.0	2.5
Target	$x\ (m)$	1.7	2.5	3.0	0.5	2.7	3.0	0.2	2.0	2.2
	$y\ (m)$	0.4	2.5	1.5	2.5	1.0	1.5	1.6	2.5	0.2

Table 6: The starting position of the obstacles.

Map	$Obstacle_i$	Obs1	Obs2	Obs3	Obs4	Obs5	Obs6	Obs7
5^{th}	$x_i\ (m)$	2.6	0.7	2.3	2.0	1.0	-	-
	$y_i\ (m)$	1.9	0.9	0.4	1.4	1.6	-	-
	$r_i\ (m)$	0.19	0.20	0.21	0.22	0.23	-	-
6^{th}	$x_i\ (m)$	1.0	1.4	2.5	0.3	0.7	2.6	1.4
	$y_i\ (m)$	0.7	1.5	2.0	0.3	1.1	0.7	2.3
	$r_i\ (m)$	0.19	0.24	0.21	0.20	0.21	0.23	0.22

5.2.1.3 Control parameters The objects were drawn using Matlab software R2020b version in Windows 10 Pro Edition 20H2 Version. The SOMA for each robot is also programmed using Matlab. The main control parameters of the algorithm are given in Tab. 7.

Table 7: The control parameter values of SOMA.

Migration	*PopSize*	*Step*	*PRT*	*PathLength*
20	40	0.11	0.1	3.0

$PRT = 0.1$, $Step = 0.11$, $Pathlength = 3$: These options are common to the SOMA algorithm, and it is selected based on the recommendation from the original paper [6, 7, 8].

All robots used in simulations have a radius $r_{robot} = 0.08m$. The sensors have a radius of active range $r_{sensor} = r_{robot} + 0.28m$. The maximum step of the robots is $d_{limit} = 0.04m$.

5.2.2 Experimental Setup for Drone Simulation

We implemented the operational experiment of drones using the self-organizing migrating algorithm in the Matlab environment, under the Windows 10 64-bit operating system.

Configuration parameters of the SOMA are listed below:

- The number of individuals in the population: $PopSize = 100$
- The number of jumps: $N_{jump} = 30$
- The PRT threshold: $PRT = 0.1$
- The granularity of each jump: $Step = 0.11$
- The maximum number of the migration loop: $MaxMig = 50$

Tab. 8 shows the positions of obstacles in the Cartesian coordinate system. The entire physical size of the obstacle is considered a sphere with radius $r_{obstacle}$. These obstacles do not move and are placed in positions that prevent the direct movement of the drones to the targets. Tab. 9 presents the starting position of the drones as well as the position of the respective targets.

Table 8: The positions and radius of obstacles (in *meters*)

$Obstacle_i$	Obs1	Obs2	Obs3	Obs4
x_i	-3	1	5	-2
y_i	-1	9	-4	-14
z_i	10	-1	-7	0
r_i	4	3	5	2

Table 9: The initial position of drones and targets (in *meters*)

The object	Dro1	Dro2	Tar1	Tar2
x_i	11	6	-5	-10
y_i	-14	-5	14	-12
z_i	-13	13	9	-10

In this simulation, we have assumed that the drones can move within a radius $r_{step} = 1.5m$ without any difficulty. Sensors on drones are capable of detecting obstacles within a radius of $15m$. The drones are provided with the location of the target as indicated above.

6 RESULTS AND DISCUSSION

The results are divided into two parts. The first part is the computational results of proposed algorithms tested on the CEC 2013, 2015, and 2017 and the 100-Digit Challenge Competition. Their performance is compared with well-known algorithms to investigate the SOMA position on the SI map. The second part presents their application in the field of swarm robotics.

The content presented in this section is my scientific research results and co-authors. They have also been originally published in [17, 18, 19, 20, 23, 24, 25, 50, 58, 59, 60, 68].

6.1 Computational Results of SOMA T3A

The error values of 51 continuous runs are used as a basis for comparing the performance of algorithms on dimension $D = 30$. It is obtained by the difference between the best value found by the algorithm and the global optimum value within the given search ranges of those functions $(f(x) - f(x^*))$. Note that the error value is considered to be zero when smaller than 10^{-8} as mentioned in the contest rules of [64, 65, 66].

6.1.1 Compared to Other Versions of SOMA

Table 10: Comparison of SOMA T3A with SOMA ATO and ATA on the CEC13 benchmark functions (30 dimensions, 51 runs).

f	**SOMA T3A** Mean (Std Dev)	**SOMA AllToOne** Mean (Std Dev)	**SOMA AllToAll** Mean (Std Dev)
f_1	**0.00e+00 (0.00e+00)**	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈
f_2	**3.48e+05 (2.01e+05)**	1.70e+07 (4.03e+06)−	1.52e+07 (4.49e+06)−
f_3	**2.06e+07 (3.02e+07)**	9.75e+07 (1.20e+08)−	2.32e+08 (1.93e+08)−
f_4	**4.79e+02 (3.26e+02)**	2.49e+04 (5.35e+03)−	2.03e+04 (5.02e+03)−
f_5	**0.00e+00 (0.00e+00)**	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈
f_6	3.13e+01 (2.46e+01)	3.24e+01 (2.34e+01)−	**2.86e+01 (1.88e+01)**+
f_7	**3.48e+01 (9.38e+00)**	8.21e+01 (1.24e+01)−	9.12e+01 (1.57e+01)−
f_8	**2.09e+01 (4.62e−02)**	2.09e+01 (5.24e−02)≈	2.10e+01 (5.80e−02)≈
f_9	2.85e+01 (2.33e+00)	3.11e+01 (1.30e+00)−	**2.82e+01 (1.71e+00)**≈
f_{10}	**1.87e−01 (9.30e−02)**	3.88e−01 (2.44e−01)−	3.03e−01 (1.23e−01)−
f_{11}	2.58e+00 (1.40e+00)	7.02e−01 (8.74e−01)+	**1.95e−01 (4.46e−01)**+
f_{12}	**3.91e+01 (1.16e+01)**	1.65e+02 (1.87e+01)−	1.22e+02 (2.09e+01)−
f_{13}	**8.03e+01 (2.67e+01)**	1.80e+02 (1.41e+01)−	1.59e+02 (2.02e+01)−
f_{14}	1.56e+01 (8.22e+00)	1.19e+01 (6.64e+00)+	**4.17e+00 (3.80e+00)**+
f_{15}	**3.87e+03 (6.38e+02)**	5.51e+03 (3.16e+02)−	4.59e+03 (3.72e+02)−
f_{16}	2.08e+00 (5.02e−01)	2.13e+00 (2.29e−01)≈	**1.89e+00 (1.82e−01)**+
f_{17}	3.36e+01 (1.19e+00)	3.14e+01 (6.21e−01)+	**3.07e+01 (2.15e−01)**+
f_{18}	**6.37e+01 (1.27e+01)**	2.12e+02 (1.43e+01)−	1.84e+02 (1.59e+01)−
f_{19}	1.97e+00 (3.23e−01)	1.88e+00 (3.01e−01)≈	**1.47e+00 (1.97e−01)**+
f_{20}	**1.05e+01 (7.97e−01)**	1.33e+01 (5.48e−01)−	1.35e+01 (5.97e−01)−
f_{21}	3.16e+02 (8.90e+01)	3.21e+02 (8.91e+01)≈	**2.59e+02 (5.32e+01)**+
f_{22}	1.29e+02 (4.65e+01)	1.42e+02 (5.39e+01)≈	**6.02e+01 (4.18e+01)**+
f_{23}	**4.61e+03 (8.02e+02)**	6.27e+03 (3.41e+02)−	5.38e+03 (5.05e+02)−
f_{24}	**2.50e+02 (1.06e+01)**	2.76e+02 (9.04e+00)−	2.73e+02 (8.09e+00)−
f_{25}	**2.95e+02 (6.61e+00)**	3.05e+02 (3.74e+00)−	2.98e+02 (6.21e+00)−
f_{26}	**2.00e+02 (7.75e−03)**	2.01e+02 (3.01e−01)−	2.01e+02 (3.55e−01)−
f_{27}	1.01e+03 (9.22e+01)	1.04e+03 (2.08e+02)−	**9.33e+02 (2.66e+02)**≈
f_{28}	**3.00e+02 (0.00e+00)**	**3.00e+02 (0.00e+00)**≈	**3.00e+02 (0.00e+00)**≈
	+	3	8
	−	17	14
	≈	8	6

The comparison results between SOMA T3A and the original version tested on the CEC13 benchmark suite are shown in Tab. 10. On each row corresponding to each function, the best results are marked in bold. The signs (+), (−), and (≈) respectively indicate that the algorithm has significantly better results, significantly worse results, and not significantly better or worse results compared to SOMA T3A using the Wilcoxon rank-sum test at the significance level 5%.

SOMA T3A demonstrates its performance in the unimodal functions $f_1 - f_5$ compared to the other versions when reaching significantly better results than the rest. On the basic multimodal functions and composition functions, SOMA T3A continues to prove good results when significantly better than SOMA ATO (11 out of 19 problems) and SOMA ATA (10 out of 19 problems), only worse than 3 and 7 out of 19 problems respectively.

Tab. 11 presents the comparison results of 30 functions on the CEC17 benchmark suite. In this experiment, the proposed algorithm shows the superior performance compared to SOMA ATO with 26 of 30 cases having significantly better results, only losing 4 cases.

Table 11: Comparison of SOMA T3A with SOMA ATO and ATA on the CEC17 benchmark functions (30 dimensions, 51 runs).

F	**SOMA T3A** Mean (Std Dev)	**SOMA AllToOne** Mean (Std Dev)	**SOMA AllToAll** Mean (Std Dev)
F_1	**0.00e+00 (0.00e+00)**	1.48e+03 (2.41e+03)−	5.52e+02 (1.14e+03)−
F_2	2.97e+09 (1.46e+10)	3.61e+08 (2.47e+09)+	**9.10e+04 (5.96e+05)**+
F_3	**1.90e−02 (6.43e−02)**	1.54e+04 (4.40e+03)−	9.89e+03 (3.34e+03)−
F_4	**5.12e+01 (3.12e+01)**	8.46e+01 (2.78e+01)−	8.54e+01 (2.19e+01)−
F_5	5.20e+01 (1.70e+01)	6.81e+01 (6.19e+00)−	**4.99e+01 (9.35e+00)**≈
F_6	4.22e−04 (5.76e−04)	**0.00e+00 (0.00e+00)**+	**0.00e+00 (0.00e+00)**+
F_7	**7.77e+01 (1.48e+01)**	1.06e+02 (7.50e+00)−	8.27e+01 (8.37e+00)≈
F_8	5.65e+01 (1.48e+01)	7.07e+01 (6.25e+00)−	**5.46e+01 (8.48e+00)**≈
F_9	4.14e+00 (4.32e+00)	**7.17e−01 (1.25e+00)**+	3.05e+00 (4.67e+00)+
F_{10}	2.51e+03 (4.67e+02)	3.08e+03 (2.21e+02)−	**2.35e+03 (2.98e+02)**≈
F_{11}	2.35e+01 (2.15e+01)	6.00e+01 (2.77e+01)−	**1.75e+01 (1.32e+01)**≈
F_{12}	**1.04e+04 (5.79e+03)**	3.96e+05 (2.70e+05)−	5.09e+05 (3.26e+05)−
F_{13}	**1.63e+02 (2.24e+02)**	1.31e+04 (1.39e+04)−	8.30e+03 (7.59e+03)−
F_{14}	**6.86e+01 (7.42e+01)**	4.28e+04 (3.36e+04)−	8.75e+04 (1.14e+05)−
F_{15}	**2.52e+01 (1.77e+01)**	7.45e+03 (7.70e+03)−	2.12e+03 (2.42e+03)−
F_{16}	**5.61e+02 (1.66e+02)**	7.83e+02 (1.27e+02)−	5.89e+02 (1.72e+02)≈
F_{17}	**9.63e+01 (7.21e+01)**	2.34e+02 (7.51e+01)−	1.45e+02 (8.89e+01)−
F_{18}	**1.24e+04 (1.34e+04)**	2.09e+05 (1.09e+05)−	2.04e+05 (1.17e+05)−
F_{19}	**1.91e+01 (9.23e+00)**	7.99e+03 (8.78e+03)−	2.92e+03 (3.60e+03)−
F_{20}	**1.57e+02 (8.33e+01)**	2.91e+02 (9.03e+01)−	1.85e+02 (8.71e+01)−
F_{21}	**2.45e+02 (4.41e+01)**	2.79e+02 (8.92e+00)−	2.50e+02 (2.86e+01)≈
F_{22}	**3.85e+02 (8.73e+02)**	5.43e+02 (1.02e+03)−	6.45e+02 (1.08e+03)−
F_{23}	**4.01e+02 (1.70e+01)**	4.26e+02 (9.35e+00)−	4.04e+02 (1.05e+01)≈
F_{24}	**4.74e+02 (1.79e+01)**	5.49e+02 (1.40e+01)−	5.12e+02 (4.29e+01)−
F_{25}	3.88e+02 (1.11e+00)	3.87e+02 (1.09e+00)+	**3.87e+02 (9.42e−01)**+
F_{26}	**6.61e+02 (5.68e+02)**	1.18e+03 (6.92e+02)−	1.00e+03 (5.68e+02)−
F_{27}	**5.12e+02 (6.49e+00)**	5.20e+02 (6.22e+00)−	**5.12e+02 (6.48e+00)**≈
F_{28}	**3.23e+02 (4.25e+01)**	4.04e+02 (1.14e+01)−	4.02e+02 (4.98e+00)−
F_{29}	5.63e+02 (9.71e+01)	6.67e+02 (7.73e+01)−	**5.29e+02 (7.50e+01)**≈
F_{30}	**4.34e+03 (2.00e+03)**	7.12e+03 (2.83e+03)−	4.60e+03 (9.44e+02)−
	+	4	4
	−	26	16
	≈	0	10

Compared to SOMA ATA, SOMA T3A gives better results in hybrid functions $F_{11} - F_{20}$ and composition functions $F_{21} - F_{30}$. For unimodal functions $F_1 - F_3$ and simple multimodal functions $F_4 - F_{10}$, the results of the two algorithms are similar. Totally, SOMA T3A is better than Soma ATA with 16(+), 10(≈) and 4(−).

6.1.2 Compared to Other Algorithms

In the previous subsection, we compared SOMA T3A with the SOMA family, and it showed better performance. To further validate its effectiveness outside the family, a comparison with other algorithms in the same class needs to be implemented.

Tab. 12 and Tab. 13 report the comparison result between SOMA T3A and the other algorithms: ABC, HFPSO, SSA, and SAMPE Jaya implemented on the CEC13 and CEC17 benchmark suites respectively.

Table 12: Comparison of SOMA T3A with other algorithms on the CEC13 benchmark functions (30 dimensions). All results are the means of 51 runs.

f	**SOMA T3A** Mean (Std Dev)	**ABC** Mean (Std Dev)	**HFPSO** Mean (Std Dev)	**SSA** Mean (Std Dev)	**SAMPE JAYA** 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)**≈	8.65e+03 (2.63e+03)−
f_2	**3.48e+05 (2.01e+05)**	8.60e+06 (2.84e+06)−	4.00e+05 (1.78e+05)≈	1.21e+06 (5.21e+05)−	4.37e+07 (6.30e+06)−
f_3	**2.06e+07 (3.02e+07)**	4.68e+08 (4.83e+08)−	4.30e+08 (7.11e+08)−	2.46e+08 (3.11e+08)−	4.94e+07 (3.01e−08)−
f_4	4.79e+02 (3.26e+02)	9.76e+04 (1.37e+04)−	1.53e+02 (1.13e+02)+	**8.30e+01 (7.26e+01)**+	3.94e+04 (8.47e+03)−
f_5	**0.00e+00 (0.00e+00)**	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	2.09e−03 (1.77e−04)−	2.67e+03 (1.71e+03)−
f_6	3.13e+01 (2.46e+01)	**1.47e+01 (4.38e+00)**+	2.62e+01 (2.59e+01)+	3.71e+01 (2.83e+01)≈	5.47e+02 (2.94e+02)−
f_7	**3.48e+01 (9.38e+00)**	1.25e+02 (1.68e+01)−	8.40e+01 (3.12e+01)−	7.14e+01 (1.85e+01)−	1.18e+02 (2.29e+01)−
f_8	2.09e+01 (4.62e−02)	2.10e+01 (5.15e−02)≈	2.09e+01 (6.05e−02)+	2.09e+01 (5.04e−02)≈	**1.25e+01 (0.00e+00)**+
f_9	2.85e+01 (2.33e+00)	3.14e+01 (1.73e+00)−	**1.93e+01 (3.49e+00)**+	2.40e+01 (3.21e+00)+	8.75e+01 (0.00e+00)−
f_{10}	1.87e−01 (9.30e−02)	1.49e+00 (2.42e−01)−	1.21e−01 (5.43e−02)+	**9.11e−02 (6.10e−02)**+	9.14e+02 (2.41e+02)−
f_{11}	2.58e+00 (1.40e+00)	**0.00e+00 (0.00e+00)**+	5.02e+01 (1.62e+01)−	1.33e+02 (4.79e+01)−	3.29e+02 (4.36e+01)−
f_{12}	**3.91e+01 (1.16e+01)**	2.75e+02 (5.16e+01)−	9.29e+01 (2.70e+01)−	1.19e+02 (3.51e+01)−	3.23e+02 (1.97e+01)−
f_{13}	**8.03e+01 (2.67e+01)**	3.26e+02 (3.83e+01)−	1.64e+02 (3.11e+01)−	2.21e+02 (5.24e+01)−	2.89e+02 (3.77e+01)−
f_{14}	1.56e+01 (8.22e+00)	**2.21e−01 (2.80e−01)**+	1.46e+03 (3.83e+02)−	3.75e+03 (6.34e+02)−	2.90e+03 (6.23e+02)−
f_{15}	3.87e+03 (6.38e+02)	3.79e+03 (3.81e+02)≈	3.89e+03 (5.65e+02)≈	3.90e+03 (5.70e+02)≈	**1.94e+03 (0.00e+00)**+
f_{16}	2.08e+00 (5.02e−01)	1.03e+00 (1.74e−01)+	1.09e+00 (4.74e−01)+	**5.14e−01 (1.90e−01)**+	2.46e+00 (2.60e−01)−
f_{17}	3.36e+01 (1.19e+00)	**3.05e+01 (2.78e−02)**+	7.24e+01 (1.17e+01)−	1.71e+02 (4.21e+01)−	4.76e+02 (1.37e+02)−
f_{18}	**6.37e+01 (1.27e+01)**	2.93e+02 (3.57e+01)−	1.05e+02 (1.82e+01)−	1.68e+02 (3.90e+01)−	4.22e+02 (4.81e+01)−
f_{19}	1.97e+00 (3.23e−01)	**3.64e−01 (8.98e−02)**+	8.78e+00 (1.74e+01)−	7.19e+00 (1.97e+00)−	2.06e+03 (4.89e+02)−
f_{20}	**1.05e+01 (7.97e−01)**	1.46e+01 (1.99e−01)−	1.47e+01 (9.58e−01)−	1.23e+01 (1.34e+00)−	1.42e+01 (7.68e−01)−
f_{21}	3.16e+02 (8.90e+01)	1.48e+02 (3.30e+01)+	3.25e+02 (8.20e+01)≈	3.19e+02 (8.24e+01)−	**8.50e+01 (0.00e+00)**+
f_{22}	1.29e+02 (4.65e+01)	**2.90e+01 (1.02e+01)**+	1.82e+03 (5.19e+02)−	4.25e+03 (7.86e+02)−	1.85e+02 (0.00e+00)−
f_{23}	4.61e+03 (8.02e+02)	4.88e+03 (4.85e+02)−	4.84e+03 (1.16e+03)≈	4.25e+03 (7.46e+02)+	**2.85e+02 (0.00e+00)**+
f_{24}	**2.50e+02 (1.06e+01)**	2.91e+02 (6.01e+00)−	2.72e+02 (1.00e+01)−	2.71e+02 (1.11e+01)−	3.85e+02 (0.00e+00)−
f_{25}	2.95e+02 (6.61e+00)	3.08e+02 (5.24e+00)−	3.09e+02 (1.87e+01)−	**2.85e+02 (9.87e+00)**+	4.85e+02 (0.00e+00)−
f_{26}	**2.00e+02 (7.75e−03)**	2.01e+02 (2.79e−01)−	3.27e+02 (5.99e+01)−	2.26e+02 (6.18e+01)−	5.85e+02 (0.00e+00)−
f_{27}	1.01e+03 (9.22e+01)	**5.17e+02 (2.73e+02)**+	8.91e+02 (9.65e+01)+	9.47e+02 (8.74e+01)+	6.85e+02 (0.00e+00)+
f_{28}	3.00e+02 (0.00e+00)	**2.29e+02 (9.17e+01)**≈	5.43e+02 (5.72e+02)−	3.79e+02 (3.54e+02)−	7.85e+02 (0.00e+00)−
	+	9	7	7	5
	−	14	15	17	23
	≈	5	6	4	0

In Tab. 12, the mean and standard deviation of the error values obtained by five algorithms within 51 runs are listed. The total numbers of +, −, ≈ signs are also listed in the last three rows of the table. Compare to other algorithms, SOMA T3A shows the competitive results when reaching significantly better results more than a half of 28 benchmark functions (within 14 winnings compared to ABC, 15 for HFPSO, 17 for SSA, and 23 for SAMPE Jaya).

The superiority of the proposed algorithm is shown more clearly on the CEC17 test as listed in Tab. 13. The overall performance reached by SOMA T3A on this benchmark suite is higher than two-thirds of the 30 functions (within 22 winnings compared to ABC, 23 for HFPSO, 28 for SSA, and 22 for SAMPE Jaya).

Table 13: Comparison of SOMA T3A with other algorithms on the CEC17 benchmark functions (30 dimensions). All results are the means of 51 runs.

F	SOMA T3A Mean (Std Dev)	ABC Mean (Std Dev)	HFPSO Mean (Std Dev)	SSA Mean (Std Dev)	SAMPE JAYA Mean (Std Dev)
F_1	**0.00e+00 (0.00e+00)**	1.79e+02 (2.05e+02)−	4.94e+03 (6.04e+03)−	5.82e+03 (6.27e+03)−	9.95e+09 (5.01e+09)−
F_2	2.97e+09 (1.46e+10)	9.15e+08 (5.64e+09)≈	4.81e+06 (3.44e+07)+	**1.38e+06 (5.73e+06)**+	6.87e+35 (3.43e+36)−
F_3	1.90e−02 (6.43e−02)	1.16e+05 (2.00e+04)−	**0.00e+00 (0.00e+00)**+	3.05e−08 (8.51e−09)+	5.09e+04 (1.35e+04)−
F_4	5.12e+01 (3.12e+01)	**2.82e+01 (2.90e+01)**+	4.64e+01 (2.59e+01)+	9.10e+01 (2.01e+01)−	1.01e+03 (5.95e+02)−
F_5	**5.20e+01 (1.70e+01)**	8.53e+01 (1.22e+01)−	8.61e+01 (2.72e+01)−	1.18e+02 (3.51e+01)−	2.40e+02 (3.10e+01)−
F_6	4.22e−04 (5.76e−04)	**0.00e+00 (0.00e+00)**+	2.40e+00 (3.76e+00)−	2.83e+01 (9.76e+00)−	3.59e+01 (6.23e+00)−
F_7	**7.77e+01 (1.48e+01)**	9.71e+01 (9.07e+00)−	1.10e+02 (2.62e+01)−	1.63e+02 (4.29e+01)−	3.44e+02 (1.33e+02)−
F_8	**5.65e+01 (1.48e+01)**	9.22e+01 (1.34e+01)−	8.55e+01 (2.49e+01)−	1.17e+02 (2.78e+01)−	2.17e+02 (2.59e+01)−
F_9	**4.14e+00 (4.32e+00)**	8.42e+02 (4.39e+02)−	3.19e+02 (9.03e+02)−	2.11e+03 (1.39e+03)−	1.01e+02 (7.18e−14)−
F_{10}	2.51e+03 (4.67e+02)	2.25e+03 (2.70e+02)+	3.38e+03 (6.50e+02)−	3.82e+03 (7.13e+02)−	**6.42e−01 (0.00e+00)**+
F_{11}	**2.35e+01 (2.15e+01)**	4.79e+02 (3.47e+02)−	1.30e+02 (5.52e+01)−	1.68e+02 (4.72e+01)−	1.43e+02 (1.51e+02)−
F_{12}	1.04e+04 (5.79e+03)	3.79e+05 (2.20e+05)−	2.01e+05 (2.94e+05)−	1.59e+06 (1.20e+06)−	**5.56e+02 (6.89e−13)**+
F_{13}	**1.63e+02 (2.24e+02)**	6.12e+03 (5.23e+03)−	3.21e+04 (3.12e+04)−	1.06e+05 (7.45e+04)−	4.97e+07 (4.44e+07)−
F_{14}	**6.86e+01 (7.42e+01)**	6.77e+04 (4.57e+04)−	3.38e+03 (3.13e+03)−	3.59e+03 (2.91e+03)−	5.57e+04 (5.92e+04)−
F_{15}	**2.52e+01 (1.77e+01)**	7.07e+02 (8.23e+02)−	1.44e+04 (1.86e+04)−	6.01e+04 (3.88e+04)−	3.83e+05 (9.94e+04)−
F_{16}	**5.61e+02 (1.66e+02)**	6.33e+02 (1.65e+02)−	7.76e+02 (2.64e+02)−	8.93e+02 (3.16e+02)−	1.23e+03 (2.15e+02)−
F_{17}	**9.63e+01 (7.21e+01)**	2.20e+02 (1.01e+02)−	3.12e+02 (1.81e+02)−	3.43e+02 (1.72e+02)−	2.66e+02 (6.98e+01)−
F_{18}	**1.24e+04 (1.34e+04)**	1.55e+05 (8.52e+04)−	1.55e+05 (1.80e+05)−	1.85e+05 (1.66e+05)−	8.26e+05 (9.54e+05)−
F_{19}	**1.91e+01 (9.23e+00)**	1.14e+03 (1.27e+03)−	7.82e+03 (8.57e+03)−	2.93e+05 (1.35e+05)−	2.22e+07 (6.23e+07)−
F_{20}	**1.57e+02 (8.33e+01)**	2.63e+02 (9.04e+01)−	2.95e+02 (1.17e+02)−	3.63e+02 (1.34e+02)−	3.40e+02 (1.38e+02)−
F_{21}	2.45e+02 (4.41e+01)	2.49e+02 (7.62e+01)−	2.92e+02 (2.02e+01)−	3.01e+02 (2.39e+01)−	**1.70e+02 (5.74e−14)**+
F_{22}	**3.85e+02 (8.73e+02)**	5.47e+02 (1.04e+03)−	1.64e+03 (1.80e+03)≈	2.32e+03 (1.89e+03)−	8.78e+02 (6.51e+02)−
F_{23}	**4.01e+02 (1.70e+01)**	4.19e+02 (2.71e+01)−	5.29e+02 (6.78e+01)−	4.52e+02 (2.78e+01)−	6.19e+02 (3.88e+01)−
F_{24}	4.74e+02 (1.79e+01)	**4.61e+02 (2.05e+02)**+	6.04e+02 (8.44e+01)−	5.11e+02 (2.79e+01)−	6.95e+02 (3.88e+01)−
F_{25}	3.88e+02 (1.11e+00)	**3.84e+02 (7.86e−01)**+	3.90e+02 (6.99e+00)≈	3.94e+02 (1.64e+01)−	6.17e+02 (1.76e+02)−
F_{26}	6.61e+02 (5.68e+02)	**4.01e+02 (5.21e+02)**+	1.39e+03 (1.16e+03)≈	2.02e+03 (7.79e+02)−	4.87e+02 (6.31e−13)+
F_{27}	5.12e+02 (6.49e+00)	5.14e+02 (6.08e+00)≈	5.56e+02 (4.65e+01)−	5.38e+02 (1.69e+01)−	**3.87e+02 (4.59e−13)**+
F_{28}	3.23e+02 (4.25e+01)	3.99e+02 (1.38e+01)−	3.56e+02 (6.72e+01)≈	4.09e+02 (4.53e+01)−	**2.87e+02 (4.02e−13)**+
F_{29}	5.63e+02 (9.71e+01)	6.29e+02 (7.91e+01)−	7.54e+02 (1.50e+02)−	9.25e+02 (2.28e+02)−	**1.87e+02 (1.15e−13)**+
F_{30}	4.34e+03 (2.00e+03)	4.99e+03 (1.40e+03)−	1.09e+04 (1.36e+04)−	1.17e+06 (7.48e+05)−	**8.72e+01 (0.00e+00)**+
	+	6	3	2	8
	−	22	23	28	22
	≈	2	4	0	0

6.2 Comparative Results of SOMA Pareto

The comparison results between SOMA Pareto and other algorithms are presented in form tables, in which each row is the mean and standard deviation errors of running 51 consecutive trials corresponding to each function. The signs (+), (−), and (≈) respectively indicate that the comparing algorithm has significantly better results, significantly worse results, and not significantly better or worse results compared to SOMA Pareto using the Wilcoxon rank-sum test (WRT) at 5% significance level. The best result without statistical tests on each row was marked in bold.

6.2.1 Out-Performance of SOMA Pareto Compares to Other SOMA

Tab. 14 shows the performance of SOMA Pareto compared to other versions of SOMA. For SOMA ATO, SOMA Pareto had 25 out of 30 cases having significantly better results (win), 4 cases having significantly worse results (lose), and one case unconfirm significantly better or worse results (draw) using WRT. Compared to SOMA ATA, SOMA Pareto wins 21, loses 8, and draws 1. For SOMA T3A, SOMA Pareto wins 14, loses 11, and draws 5. These results demonstrate that the proposed algorithm is completely better than the original versions, as well as the latest version of SOMA.

Table 14: Comparison of SOMA Pareto with SOMA ATO, ATA, and SOMA T3A on the CEC17 benchmark functions (30 dimensions, 51 runs).

F	**SOMA Pareto** Mean (Std Dev)	**SOMA AllToOne** Mean (Std Dev)	**SOMA AllToAll** Mean (Std Dev)	**SOMA T3A** Mean (Std Dev)
F_1	2.49e−08 (1.48e−07)	1.48e+03 (2.41e+03)−	5.52e+02 (1.14e+03)−	**0.00e+00 (0.00e+00)**+
F_2	**1.67e+03 (9.51e+03)**	3.61e+08 (2.47e+09)−	9.10e+04 (5.96e+05)−	2.97e+09 (1.46e+10)−
F_3	**3.81e−05 (1.45e−04)**	1.54e+04 (4.40e+03)−	9.89e+03 (3.34e+03)−	1.90e−02 (6.43e−02)−
F_4	**3.77e+00 (9.58e+00)**	8.46e+01 (2.78e+01)−	8.54e+01 (2.19e+01)−	5.12e+01 (3.12e+01)−
F_5	**3.00e+01 (7.20e+00)**	6.81e+01 (6.19e+00)−	4.99e+01 (9.35e+00)−	5.20e+01 (1.70e+01)−
F_6	1.41e−03 (1.83e−03)	**0.00e+00 (0.00e+00)**+	**0.00e+00 (0.00e+00)**+	4.22e−04 (5.76e−04)+
F_7	**5.13e+01 (7.60e+00)**	1.06e+02 (7.50e+00)−	8.27e+01 (8.37e+00)−	7.77e+01 (1.48e+01)−
F_8	**2.85e+01 (7.89e+00)**	7.07e+01 (6.25e+00)−	5.46e+01 (8.48e+00)−	5.65e+01 (1.48e+01)−
F_9	6.37e+00 (5.03e+00)	**7.17e−01 (1.25e+00)**+	3.05e+00 (4.67e+00)+	4.14e+00 (4.32e+00)+
F_{10}	2.84e+03 (5.85e+02)	3.08e+03 (2.21e+02)−	**2.35e+03 (2.98e+02)**+	2.51e+03 (4.67e+02)+
F_{11}	2.80e+01 (1.94e+01)	6.00e+01 (2.77e+01)−	**1.75e+01 (1.32e+01)**+	2.35e+01 (2.15e+01)+
F_{12}	1.13e+04 (5.68e+03)	3.96e+05 (2.70e+05)−	5.09e+05 (3.26e+05)−	**1.04e+04 (5.79e+03)**≈
F_{13}	**7.26e+01 (5.11e+01)**	1.31e+04 (1.39e+04)−	8.30e+03 (7.59e+03)−	1.63e+02 (2.24e+02)−
F_{14}	1.21e+02 (3.14e+02)	4.28e+04 (3.36e+04)−	8.75e+04 (1.14e+05)−	**6.86e+01 (7.42e+01)**+
F_{15}	1.49e+02 (3.22e+02)	7.45e+03 (7.70e+03)−	2.12e+03 (2.42e+03)−	**2.52e+01 (1.77e+01)**≈
F_{16}	6.93e+02 (2.43e+02)	7.83e+02 (1.27e+02)−	5.89e+02 (1.72e+02)+	**5.61e+02 (1.66e+02)**+
F_{17}	**8.78e+01 (8.76e+01)**	2.34e+02 (7.51e+01)−	1.45e+02 (8.89e+01)−	9.63e+01 (7.21e+01)≈
F_{18}	**1.15e+04 (6.72e+03)**	2.09e+05 (1.09e+05)−	2.04e+05 (1.17e+05)−	1.24e+04 (1.34e+04)≈
F_{19}	4.29e+01 (4.28e+01)	7.99e+03 (8.78e+03)−	2.92e+03 (3.60e+03)−	**1.91e+01 (9.23e+00)**+
F_{20}	1.75e+02 (8.22e+01)	2.91e+02 (9.03e+01)−	1.85e+02 (8.71e+01)≈	**1.57e+02 (8.33e+01)**≈
F_{21}	**2.28e+02 (8.31e+00)**	2.79e+02 (8.92e+00)−	2.50e+02 (2.86e+01)−	2.45e+02 (4.41e+01)−
F_{22}	**1.45e+02 (3.21e+02)**	5.43e+02 (1.02e+03)−	6.45e+02 (1.08e+03)−	3.85e+02 (8.73e+02)−
F_{23}	**3.82e+02 (8.72e+00)**	4.26e+02 (9.35e+00)−	4.04e+02 (1.05e+01)−	4.01e+02 (1.70e+01)−
F_{24}	**4.52e+02 (7.26e+00)**	5.49e+02 (1.40e+01)−	5.12e+02 (4.29e+01)−	4.74e+02 (1.79e+01)−
F_{25}	3.88e+02 (3.32e+00)	3.87e+02 (1.09e+00)+	**3.87e+02 (9.42e−01)**+	3.88e+02 (1.11e+00)+
F_{26}	1.41e+03 (2.01e+02)	1.18e+03 (6.92e+02)≈	1.00e+03 (5.68e+02)+	**6.61e+02 (5.68e+02)**+
F_{27}	5.34e+02 (6.75e+00)	5.20e+02 (6.22e+00)+	**5.12e+02 (6.48e+00)**+	5.12e+02 (6.49e+00)+
F_{28}	**3.04e+02 (2.05e+01)**	4.04e+02 (1.14e+01)−	4.02e+02 (4.98e+00)−	3.23e+02 (4.25e+01)−
F_{29}	**5.20e+02 (9.75e+01)**	6.67e+02 (7.73e+01)−	5.29e+02 (7.50e+01)−	5.63e+02 (9.71e+01)−
F_{30}	**3.22e+03 (2.87e+02)**	7.12e+03 (2.83e+03)−	4.60e+03 (9.44e+02)−	4.34e+03 (2.00e+03)−
	+	4	8	11
	−	25	21	14
	≈	1	1	5

6.2.2 Promising Results Compare to Well-Known Algorithms

Mean and standard deviation of SOMA Pareto, SMADE, CMAES-RIS, DE-APC, and fk-PSO tested on 28 functions of CEC13 are presented in Tab. 15. Three last rows show the comparison results from SOMA Pareto to the rest. In this experiment, SOMA Pareto achieved competitive results compared to the SMADE and CMAES-RIS when SOMA Pareto won SMADE 9, lost 11, and won CMAES-RIS 10, lost 9. Compare to DE-APC and fk-PSO, the proposed algorithm shows much better performance when reaching 14 wins, only losing 10 and 7, respectively.

Compared to ICMLSP and SaDPSO tested on CEC15, SOMA Pareto continues to show competitive performance when winning 8 and 7, respectively, losing 7 and 6, respectively. Despite showing the superiority performance to dynFWACM, SOMA Pareto has not yet gained well when compared with TEBO. These results are shown in Tab. 16.

Tab. 17 reports the comparison result between SOMA Pareto and the other algorithms: RB-IPOP-CMA-ES, PPSO, TLBO-FL, and SAMPE-Jaya that implemented on the CEC17 benchmark suites. In this test suite, SOMA Pareto exhibits completely superior performance compared to other algorithms, when winning 25, 26, and 22 out of 30 cases respectively, except RB-IPOP-CMA-ES algorithm.

Table 15: Comparison of SOMA Pareto with well-known algorithms on the CEC13 benchmark functions (30 dimensions, 51 runs).

F	SOMA Pareto Mean (Std Dev)	SMADE Mean (Std Dev)	CMAES-RIS Mean (Std Dev)	DE-APC Mean (Std Dev)	fk-PSO 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	8.79e+04 (3.68e+04)	**0.00e+00 (0.00e+00)**+	**0.00e+00 (0.00e+00)**+	1.75e+05 (1.33e+05)−	1.59e+06 (8.11e+05)−
F_3	3.60e+07 (5.49e+07)	9.82e+03 (4.99e+04)+	**2.24e+03 (1.10e+04)**+	3.21e+06 (1.19e+07)+	2.40e+08 (3.75e+08)−
F_4	7.43e+02 (1.12e+03)	**0.00e+00 (0.00e+00)**+	**0.00e+00 (0.00e+00)**+	2.20e−01 (6.03e−01)+	4.78e+02 (1.98e+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	1.71e+01 (1.95e+01)	2.67e+00 (7.92e+00)+	**6.94e−04 (2.01e−03)**+	9.35e+00 (2.06e+00)+	2.99e+01 (1.78e+01)−
F_7	3.74e+01 (7.86e+00)	3.25e+01 (1.63e+01)≈	4.48e+01 (2.96e+01)≈	**2.18e+01 (1.89e+01)**+	6.39e+01 (3.12e+01)−
F_8	2.09e+01 (4.89e−02)	2.10e+01 (4.85e−02)≈	2.09e+01 (8.19e−02)+	**2.09e+01 (5.24e−02)**≈	2.09e+01 (6.34e−02)≈
F_9	2.35e+01 (4.33e+00)	2.23e+01 (3.61e+00)≈	2.37e+01 (1.95e+00)≈	3.07e+01 (9.41e+00)−	**1.85e+01 (2.72e+00)**+
F_{10}	3.00e−01 (1.48e−01)	1.84e−02 (1.35e−02)+	**8.31e−03 (5.46e−03)**+	6.42e−02 (4.82e−02)+	2.29e−01 (1.33e−01)+
F_{11}	1.55e+01 (4.86e+00)	1.09e+01 (4.23e+00)+	2.54e+01 (6.36e+00)−	**3.08e+00 (4.50e+00)**+	2.36e+01 (8.84e+00)−
F_{12}	3.57e+01 (8.45e+00)	5.72e+01 (1.72e+01)−	7.94e+01 (4.39e+01)−	**3.17e+01 (8.51e+00)**+	5.64e+01 (1.52e+01)−
F_{13}	8.06e+01 (2.32e+01)	1.28e+02 (3.53e+01)−	1.56e+02 (5.42e+01)−	**7.55e+01 (2.65e+01)**≈	1.23e+02 (2.21e+01)−
F_{14}	1.26e+03 (4.01e+02)	**1.33e+02 (1.28e+02)**+	7.92e+02 (2.21e+02)+	3.84e+03 (3.85e+02)−	7.04e+02 (2.40e+02)+
F_{15}	3.34e+03 (6.66e+02)	4.10e+03 (8.55e+02)−	**3.13e+03 (4.57e+02)**≈	4.14e+03 (1.08e+03)−	3.42e+03 (5.21e+02)≈
F_{16}	1.87e+00 (3.60e−01)	1.31e−01 (7.65e−02)+	**1.07e−01 (6.78e−02)**+	2.46e+00 (4.45e−01)−	8.48e−01 (2.23e−01)+
F_{17}	5.22e+01 (5.37e+00)	**3.48e+01 (1.54e+00)**+	5.50e+01 (5.24e+00)−	5.92e+01 (5.56e+00)−	5.26e+01 (7.18e+00)≈
F_{18}	**5.25e+01 (8.29e+00)**	8.33e+01 (2.08e+01)−	1.89e+02 (2.73e+01)−	6.04e+01 (9.80e+00)−	6.81e+01 (9.78e+00)−
F_{19}	2.78e+00 (7.47e−01)	2.55e+00 (5.23e−01)≈	2.80e+00 (6.42e−01)≈	**2.30e+00 (6.24e−01)**+	3.12e+00 (9.93e−01)−
F_{20}	**9.86e+00 (6.76e−01)**	1.05e+01 (8.15e−01)−	1.43e+01 (5.75e−01)−	1.26e+01 (7.40e−01)−	1.20e+01 (9.36e−01)−
F_{21}	3.28e+02 (7.87e+01)	3.27e+02 (8.73e+01)+	**1.86e+02 (4.01e+01)**+	2.67e+02 (6.58e+01)+	3.11e+02 (8.00e+01)+
F_{22}	1.30e+03 (4.06e+02)	**1.79e+02 (4.54e+01)**+	1.17e+03 (2.93e+02)≈	4.56e+03 (6.08e+02)−	8.59e+02 (3.13e+02)+
F_{23}	**3.48e+03 (6.40e+02)**	4.22e+03 (8.83e+02)−	4.03e+03 (5.43e+02)−	4.18e+03 (9.21e+02)−	3.57e+03 (5.96e+02)≈
F_{24}	**2.27e+02 (4.44e+00)**	2.32e+02 (2.60e+01)−	2.59e+02 (1.76e+01)−	2.92e+02 (1.90e+01)−	2.48e+02 (8.20e+00)−
F_{25}	2.78e+02 (8.97e+00)	2.78e+02 (1.00e+01)≈	2.82e+02 (8.50e+00)≈	2.99e+02 (6.86e+00)−	**2.49e+02 (7.89e+00)**+
F_{26}	2.00e+02 (2.11e−03)	2.15e+02 (5.30e+01)−	**1.97e+02 (1.21e+01)**≈	3.28e+02 (5.47e+01)−	2.95e+02 (7.13e+01)−
F_{27}	**6.38e+02 (8.02e+01)**	6.47e+02 (1.39e+02)≈	7.49e+02 (1.87e+02)−	1.19e+03 (1.86e+02)−	7.76e+02 (7.18e+01)−
F_{28}	3.00e+02 (3.27e−13)	3.88e+02 (3.27e+02)−	5.39e+02 (1.33e+03)−	**3.00e+02 (0.00e+00)**+	4.01e+02 (3.51e+02)−
	+	11	9	10	7
	−	9	10	14	14
	≈	8	9	4	7

Table 16: Comparison of SOMA Pareto with well-known algorithms on the CEC15 benchmark functions (30 dimensions, 51 runs).

F	SOMA Pareto Mean (Std Dev)	TEBO Mean (Std Dev)	ICMLSP Mean (Std Dev)	SaDPSO Mean (Std Dev)	dynFWACM Mean (Std Dev)
F_1	1.11e+04 (8.30e+03)	3.69e+02 (7.71e+02)+	**0.00e+00 (0.00e+00)**+	1.93e−02 (8.42e−02)+	6.17e+05 (2.49e+05)−
F_2	3.14e−01 (1.10e+00)	**4.54e−07 (3.02e−06)**+	4.05e−05 (1.44e−04)+	2.88e+02 (1.09e+03)−	3.31e+03 (3.59e+03)−
F_3	2.08e+01 (9.95e−02)	2.00e+01 (6.32e−02)+	2.00e+01 (7.57e−03)+	2.00e+01 (4.15e−05)+	**2.00e+01 (5.75e−06)**+
F_4	**2.95e+01 (6.53e+00)**	4.41e+01 (1.06e+01)−	2.31e+02 (5.66e+01)−	4.25e+01 (9.31e+00)−	1.30e+02 (3.80e+01)−
F_5	2.59e+03 (5.71e+02)	**1.96e+03 (6.32e+02)**+	4.03e+03 (6.56e+02)−	2.52e+03 (3.58e+02)≈	3.38e+03 (6.98e+02)−
F_6	5.47e+03 (5.55e+03)	**6.98e+02 (6.52e+02)**+	1.47e+03 (4.08e+02)+	1.38e+03 (6.04e+02)+	2.69e+04 (1.90e+04)−
F_7	**4.00e+00 (9.00e−01)**	4.42e+00 (1.41e+00)≈	2.07e+01 (1.45e+01)−	9.52e+00 (1.93e+00)−	1.46e+01 (2.57e+00)−
F_8	5.71e+03 (5.02e+03)	**1.21e+02 (1.48e+02)**+	9.42e+02 (2.50e+02)+	1.62e+03 (1.35e+03)+	2.40e+04 (1.32e+04)−
F_9	**1.03e+02 (1.61e−01)**	1.08e+02 (1.22e+00)−	1.63e+02 (1.32e+02)−	1.03e+02 (1.86e−01)−	1.08e+02 (9.01e−01)−
F_{10}	4.55e+03 (4.50e+03)	**6.21e+02 (9.31e+01)**+	1.43e+03 (3.36e+02)+	6.52e+03 (4.66e+03)−	3.15e+04 (2.01e+04)−
F_{11}	**3.14e+02 (5.70e+01)**	4.81e+02 (1.95e+02)−	1.12e+03 (2.72e+02)−	3.20e+02 (8.88e+00)−	6.72e+02 (1.54e+02)−
F_{12}	**1.04e+02 (4.13e−01)**	1.06e+02 (1.04e+00)−	1.61e+02 (4.11e+01)−	1.05e+02 (4.90e−01)−	1.17e+02 (1.23e+01)−
F_{13}	1.03e+02 (5.72e+00)	9.87e+01 (5.46e+00)+	8.53e−02 (1.26e−01)+	1.01e+02 (4.06e+00)≈	**2.62e−02 (7.46e−03)**+
F_{14}	3.27e+04 (4.54e+02)	3.45e+04 (4.04e+03)−	4.21e+04 (4.67e+03)−	**1.87e+04 (5.27e+03)**+	4.49e+04 (1.02e+03)−
F_{15}	1.00e+02 (2.59e−13)	**1.00e+02 (0.00e+00)**+	1.27e+02 (1.62e+01)−	1.00e+02 (1.19e−13)+	**1.00e+02 (0.00e+00)**+
	+	9	7	6	3
	−	5	8	7	12
	≈	1	0	2	0

Table 17: Comparison of SOMA Pareto with well-known algorithms on the CEC17 benchmark functions (30 dimensions, 51 runs).

F	SOMA Pareto Mean (Std Dev)	RB-IPOP-CMA-ES Mean (Std Dev)	PPSO Mean (Std Dev)	TLBO-FL Mean (Std Dev)	SAMPE-Jaya Mean (Std Dev)
F_1	**2.49e−08 (1.48e−07)**	3.15e−08 (2.75e−08)−	7.48e+02 (6.06e+02)−	3.51e+03 (3.61e+03)−	9.95e+09 (5.01e+09)−
F_2	1.67e+03 (9.51e+03)	**0.00e+00 (0.00e+00)**+	5.32e+01 (6.91e+01)+	8.52e+16 (5.81e+17)−	6.87e+35 (3.43e+36)−
F_3	3.81e−05 (1.45e−04)	**0.00e+00 (0.00e+00)**+	1.13e+00 (4.83e−01)−	2.99e+03 (1.08e+03)−	5.09e+04 (1.35e+04)−
F_4	**3.77e+00 (9.58e+00)**	5.53e+01 (1.65e+01)−	4.39e+01 (3.19e+01)−	9.01e+01 (2.37e+01)−	1.01e+03 (5.95e+02)−
F_5	**3.77e+00 (9.58e+00)**	5.53e+01 (1.65e+01)−	4.39e+01 (3.19e+01)−	9.01e+01 (2.37e+01)−	1.01e+03 (5.95e+02)−
F_6	1.41e−03 (1.83e−03)	**1.21e−07 (3.97e−08)**+	2.03e+01 (4.15e+00)−	4.87e−01 (4.24e−01)−	3.59e+01 (6.23e+00)−
F_7	5.13e+01 (7.60e+00)	**3.43e+01 (1.28e+00)**+	1.35e+02 (1.63e+01)−	1.39e+02 (4.75e+01)−	3.44e+02 (1.33e+02)−
F_8	2.85e+01 (7.89e+00)	**1.76e+00 (1.65e+00)**+	8.10e+01 (1.04e+01)−	3.67e+01 (1.84e+01)−	2.17e+02 (2.59e+01)−
F_9	6.37e+00 (5.03e+00)	**0.00e+00 (0.00e+00)**+	1.36e+03 (2.82e+02)−	3.45e+01 (2.71e+01)−	1.01e+02 (7.18e−14)−
F_{10}	2.84e+03 (5.85e+02)	1.44e+03 (5.83e+02)+	3.13e+03 (3.46e+02)−	6.69e+03 (2.77e+02)−	**6.42e−01 (0.00e+00)**+
F_{11}	**2.80e+01 (1.94e+01)**	4.11e+01 (4.76e+01)≈	8.43e+01 (1.84e+01)−	8.16e+01 (4.14e+01)−	**1.43e+02 (1.51e+02)**−
F_{12}	1.13e+04 (5.68e+03)	1.09e+03 (2.81e+02)+	2.77e+04 (8.55e+03)−	5.75e+04 (8.99e+04)−	**5.56e+02 (6.89e−13)**+
F_{13}	**7.26e+01 (5.11e+01)**	1.19e+02 (4.00e+02)−	3.21e+03 (2.88e+03)−	2.02e+04 (1.79e+04)−	4.97e+07 (4.44e+07)−
F_{14}	1.21e+02 (3.14e+02)	**9.08e+01 (5.62e+01)**≈	2.32e+03 (1.52e+03)−	7.10e+03 (5.85e+03)−	5.57e+04 (5.92e+04)−
F_{15}	**1.49e+02 (3.22e+02)**	2.18e+02 (1.84e+02)≈	2.13e+03 (1.63e+03)−	2.16e+04 (2.27e+04)−	3.83e+05 (9.94e+04)−
F_{16}	6.93e+02 (2.43e+02)	5.02e+02 (2.54e+02)+	8.46e+02 (1.53e+02)−	**4.92e+02 (3.53e+02)**+	1.23e+03 (2.15e+02)−
F_{17}	**8.78e+01 (8.76e+01)**	1.32e+02 (9.54e+01)−	3.31e+02 (1.13e+02)−	1.41e+02 (6.59e+01)−	2.66e+02 (6.98e+01)−
F_{18}	1.15e+04 (6.72e+03)	**1.60e+02 (1.14e+02)**+	6.99e+04 (3.06e+04)−	3.67e+05 (1.67e+05)−	8.26e+05 (9.54e+05)−
F_{19}	**4.29e+01 (4.28e+01)**	1.15e+02 (6.59e+01)−	1.71e+03 (1.69e+03)−	1.07e+04 (1.10e+04)−	2.22e+07 (6.23e+07)−
F_{20}	**1.75e+02 (8.22e+01)**	2.97e+02 (1.19e+02)−	3.48e+02 (9.16e+01)−	2.21e+02 (1.25e+02)−	3.40e+02 (1.38e+02)−
F_{21}	2.28e+02 (8.31e+00)	2.09e+02 (1.67e+01)+	3.05e+02 (3.30e+01)−	2.34e+02 (1.16e+01)−	**1.70e+02 (5.74e−14)**+
F_{22}	1.45e+02 (3.21e+02)	6.72e+02 (7.63e+02)−	**1.00e+02 (5.05e−07)**+	1.01e+02 (1.94e+00)+	8.78e+02 (6.51e+02)−
F_{23}	3.82e+02 (8.72e+00)	**3.39e+02 (4.93e+01)**+	6.81e+02 (3.79e+01)−	3.96e+02 (1.62e+01)−	6.19e+02 (3.88e+01)−
F_{24}	4.52e+02 (7.26e+00)	**4.19e+02 (3.06e+00)**+	7.39e+02 (4.57e+01)−	4.69e+02 (1.62e+01)−	6.95e+02 (3.88e+01)−
F_{25}	3.88e+02 (3.32e+00)	**3.87e+02 (1.47e−02)**+	3.85e+02 (1.77e+00)+	4.02e+02 (1.76e+01)−	6.17e+02 (1.76e+02)−
F_{26}	1.41e+03 (2.01e+02)	**3.94e+02 (2.17e+02)**+	2.04e+03 (1.73e+03)≈	1.42e+03 (4.69e+02)≈	4.87e+02 (6.31e−13)+
F_{27}	5.34e+02 (6.75e+00)	5.12e+02 (1.13e+01)+	7.08e+02 (5.42e+01)−	5.32e+02 (2.07e+01)≈	**3.87e+02 (4.59e−13)**+
F_{28}	3.04e+02 (2.05e+01)	3.09e+02 (2.96e+01)−	3.27e+02 (3.17e+01)−	4.30e+02 (2.67e+01)−	**2.87e+02 (4.02e−13)**+
F_{29}	5.20e+02 (9.75e+01)	4.93e+02 (1.04e+02)+	7.80e+02 (1.21e+02)−	6.15e+02 (9.09e+01)−	**1.87e+02 (1.15e−13)**+
F_{30}	3.22e+03 (2.87e+02)	2.84e+03 (1.94e+03)+	3.32e+03 (3.89e+02)≈	2.57e+04 (2.78e+04)−	**8.72e+01 (0.00e+00)**+
	+	19	3	2	8
	−	8	25	26	22
	≈	3	2	2	0

An important point to note is that 11 of the 12 algorithms used to compare (except SAMPE-Jaya) have been carefully tweaked to attend the CEC Competition in its respective years. For SOMA Pareto there is only one setting to run for 73 functions of 3 benchmark suites. It is believed that the comparison results will be more raised if SOMA Pareto is set individually for each benchmark suite. However, we have proposed the same setting for all issues to keep the generality of the proposed algorithm, giving a visual perspective to the reader.

6.3 iSOMA Performance

The comparison results between iSOMA and other algorithms are reported in form tables, in which each row is the mean and standard deviation errors of running 51 consecutive trials corresponding to each function. The signs (+), (−), and (≈) respectively indicate that the comparing algorithm has significantly better results (iSOMA loses), significantly worse results (iSOMA wins), and not significantly better or worse results (draw) compared to iSOMA using the Wilcoxon rank-sum test at 5% significance level. The best results on each row among the

participating algorithms without statistical tests were marked in bold. The last three rows in each table represent the sum total of the signs (+), (−), and (≈).

Table 18: 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

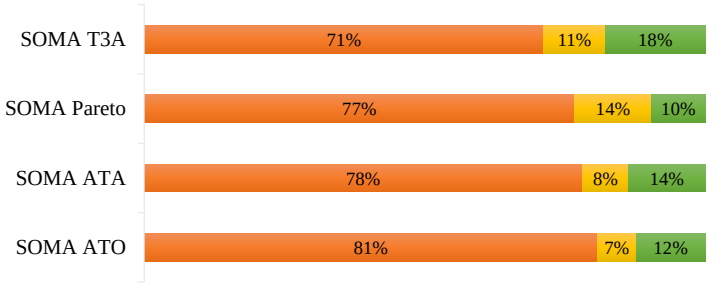
Table 19: Comparison of iSOMA with SOMA family on the CEC15 benchmark functions (30 dimensions, 51 runs).

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

6.3.1 Outperformed Other SOMAs

Tabs. 18, 19, and 20 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 case wins, while iSOMA only loses 9 and 10, draws 5 and 6, respectively, as summarized in Fig. 16. 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 T3A.



	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			

Figure 16: The summarized comparison result between the iSOMA and other SOMAs tested on 73 benchmark functions.

6.3.2 Compete Against Other Algorithms

For CEC13:

Tabs. 21 and 22 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. 17. 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 of complex functions. In contrast, iSOMA has proven its synergy on basic multimodal and composition functions.

For CEC15:

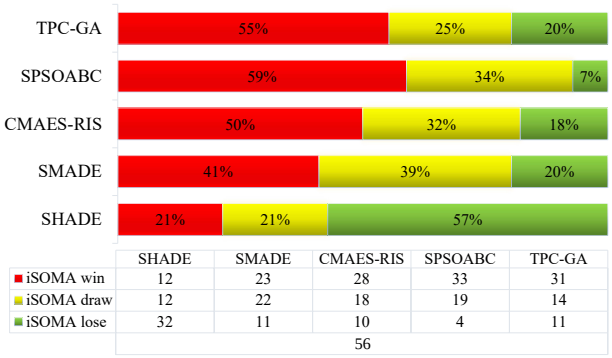
Tabs. 23 and 24, in turn, show the simulation results between iSOMA compared to DEsPA, TEBO, SaDPSO, ICMLSP and dynFWACM, on both $10D$ and $30D$ and summarized in Fig. 18.

Table 20: 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 21: 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



	SHADE	SMADE	CMAES-RIS	SPSOABC	TPC-GA
iSOMA win	12	23	28	33	31
iSOMA draw	12	22	18	19	14
iSOMA lose	32	11	10	4	11
			56		

Figure 17: The summarized comparison results between the iSOMA and other algorithms tested on the CEC13 (for both $10D$ and $30D$).

Table 22: Comparison of iSOMA with well-known algorithms on the CEC13 benchmark functions (30 dimensions, 51 runs).

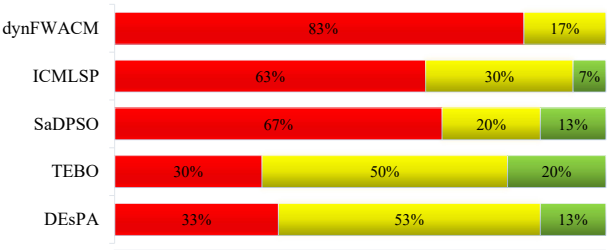
F	iSOMA Mean (Std Dev)	SHADE Mean (Std Dev)	SMADE Mean (Std Dev)	CMAES-RIS Mean (Std Dev)	SPSOABC Mean (Std Dev)	TPC-GA Mean (Std Dev)
F_1	**0.00e+00 (0.00e+00)**	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈
F_2	9.80e+04 (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

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 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 $10D$ problems, winning 17 and 13 cases over 30 functions, losing 9 and 10 cases, respectively. However, the situation was



	DEsPA	TEBO	SaDPSO	ICMLSP	dynFWACM
iSOMA win	10	9	20	19	25
iSOMA draw	16	15	6	9	5
iSOMA lose	4	6	4	2	0
			30		

Figure 18: The summarized comparison results between the iSOMA and other algorithms tested on the CEC15 (for both $10D$ and $30D$).

Table 23: 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

Table 24: Comparison of iSOMA with well-known algorithms on the CEC15 benchmark functions (30 dimensions, 51 runs).

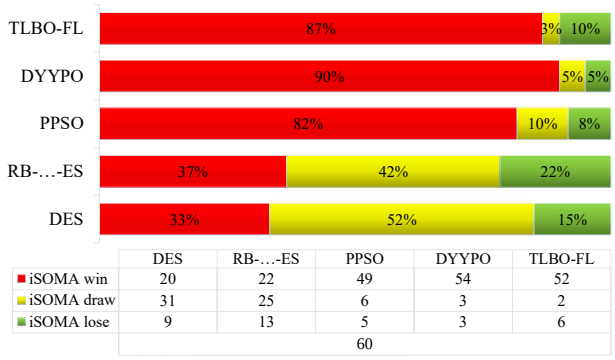
F	iSOMA Mean (Std Dev)	DEsPA Mean (Std Dev)	TEBO Mean (Std Dev)	SaDPSO Mean (Std Dev)	ICMLSP Mean (Std Dev)	dynFWACM Mean (Std Dev)
F_1	5.61e+03 (5.25e+03)	**0.00e+00 (0.00e+00)**+	3.69e+02 (7.71e+02)+	1.93e−02 (8.42e−02)+	0.00e+00 (0.00e+00)+	6.17e+05 (2.49e+05)−
F_2	1.14e−01 (4.52e−01)	**0.00e+00 (0.00e+00)**+	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 25: 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

reversed when iSOMA lost its position on $30D$. DES hence affirms the balancing power over all three groups of unimodal, basic multimodal, and composition functions.

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 Tabs. 25 and 26 and summarized in Fig. 19.



	DES	RB-…-ES	PPSO	DYYPO	TLBO-FL
iSOMA win	20	22	49	54	52
iSOMA draw	31	25	6	3	2
iSOMA lose	9	13	5	3	6
			60		

Figure 19: The summarized comparison results between the iSOMA and other algorithms tested on the CEC17 (for both $10D$ and $30D$).

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

Table 26: 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

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 an overview of the power of the proposed algorithm.

6.4 The 100-Digit Challenge Competition Result

In this section, we applied SOMA Pareto and T3A to solve 10 problems of the 100-Digit Challenge Competition and reached the highest score of 93 points. SOMA was structured into the Initialization, Organization, Migration, and Update process. How to organize individuals in the population to become the Migrants and the Leader, as well as the linear adaptive PRT and the cosine-based adaptive $Step$ parameters, are the major improvements of this method beside an increment of $MaxFEs$ to 10^9. This helps the algorithm avoid being trapped in the local minima and reaching the good score in almost all functions.

SOMA Pareto and T3A have been run 50 consecutive trials for each function, and the total number of correct digits in the 25 trials that had the best values has been counted. The average number of correct digits in the 25 best trials is the score for that function. For example, the

algorithm reaches equal or greater than 50% of the trials achieving 10 digits, then the score of that function is 10 points. The highest score in total is 100 points, refer to [67] for more details.

The algorithm will terminate when one of the two following criteria is met:

- achieving the 10-digit level accuracy,
- achieving the $MaxFEs$ ($MaxFEs = 10^9$ for 10 functions).

SOMA T3A has achieved 93 points in total in a run of 50 times for each function. Tab. 27 shows the detailed results, which the sequence number of functions are listed in the first column, the number of correct digits columns count the number of trials in a run of 50 times that the algorithm achieved from 1 to 10 correct digits respective to each function. The score for each function is the average number of correct digits of the best 25 runs that showed in the last column. The total score of all functions is presented in the last row.

In Tab. 27, SOMA T3A reached 50 over 50 runs achieving 10 correct digits on functions 1 to 6 and function 10. The score for those functions is 70 points (10 points for each function).

Table 27: SOMA T3A - Fifty runs for each function sorted by the number of correct digits

Function	Number of correct digits											Score
	0	1	2	3	4	5	6	7	8	9	10	
1	0	0	0	0	0	0	0	0	0	0	50	10
2	0	0	0	0	0	0	0	0	0	0	50	10
3	0	0	0	0	0	0	0	0	0	0	50	10
4	0	0	0	0	0	0	0	0	0	0	50	10
5	0	0	0	0	0	0	0	0	0	0	50	10
6	0	0	0	0	0	0	0	0	0	0	50	10
7	0	0	2	0	0	0	0	0	0	0	48	10
8	0	0	12	0	0	0	0	0	0	0	38	10
9	0	0	1	49	0	0	0	0	0	0	0	3
10	0	0	0	0	0	0	0	0	0	0	50	10
											Total:	93

For functions 7 and 8, SOMA T3A has 2 and 12 times achieving 2 correct digits, has 48 and 38 times achieving 10 correct digits, respectively. The score for these functions is 20 points (10 for each of them).

Function 9 is the most difficult function for SOMA T3A (and other participating algorithms) when it only has 1 time achieving 2 correct digits and 49 times achieving 3 correct digits, there are no times in 50 runs that the algorithm achieves above 3 correct digits. The score for function 9 is 3 points.

In total, SOMA T3A achieved 93 points.

Tab. 28 representative calculation results for ten functions in the 100-Digit Challenge Competition.

In this table, our algorithm accomplished 50 over 50 runs achieving ten correct digits from functions 1, 2, 3, 4, 5, 6, 7, 10. According to the competition rule, we got the highest score for these functions. In the case of functions 8 and 9, we achieved score 2 and 3.04 points, respectively. Accurately, with function 8 we reach two correct digits in 50 times, and there are no times in 50 runs that the algorithm achieves above 3 correct digits. Likewise, with function

Table 28: SOMA Pareto - Fifty runs for each function sorted by the number of correct digits

Function	Number of correct digits											Score
	0	1	2	3	4	5	6	7	8	9	10	
1	0	0	0	0	0	0	0	0	0	0	50	10
2	0	0	0	0	0	0	0	0	0	0	50	10
3	0	0	0	0	0	0	0	0	0	0	50	10
4	0	0	0	0	0	0	0	0	0	0	50	10
5	0	0	0	0	0	0	0	0	0	0	50	10
6	0	0	0	0	0	0	0	0	0	0	50	10
7	0	0	2	0	0	0	0	0	0	0	48	10
8	0	0	50	0	0	0	0	0	0	0	0	2
9	0	0	0	49	1	0	0	0	0	0	0	3.04
10	0	0	0	0	0	0	0	0	0	0	50	10
											Total:	85.04

9 we had 49 times reaching 3 correct digits, and just 1 out of 50 reaches further correct digits. In total, we achieved the score points 85.04 out of 100.

These versions, SOMA T3A and SOMA Pareto, have made great strides, holding 3^{rd} (the same ranking with HyDE-DF [21]) and 5^{th} out of 38 algorithms participating in the 100-Digit Challenge respectively, which reported in [22] 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).

6.5 Obstacle Avoidance for Mobile Robot

The simulation results are presented in the form of selected figures captured from the robot's movement in 2D and 3D.

In those 2D figures, robots are plotted on Cartesian coordinates with obstacles and targets. The circle around the robot represents the working range of the sensors. Obstacles are drawn in a dark color circle. As the robot moves, obstacles detected in the sensor's active area will be represented by bright colors. These obstacles will turn bright colors when they are detected by sensors located on the robot.

In 3D figures, the robot is represented as a big black dot, and the robot's path is represented by small black dots. Contour lines represent the surrounding environment, they can change depending on the distance from the robot to the target and the obstacles.

6.5.1 For Map 1

Figs. 20 and 21 show the robot's movement in 2D and 3D, respectively. They were captured at the step of 2^{nd}, 13^{th}, 32^{nd}, 38^{th}, 58^{th}, and 78^{th}. At the 2^{nd} step in Fig. 20, the robot has not detected the obstacles yet so they are in a dark color, and the robot tends to move straight towards the target. At this moment, the contours on the 3D map of Fig. 21 are also *"flat"* (without hills).

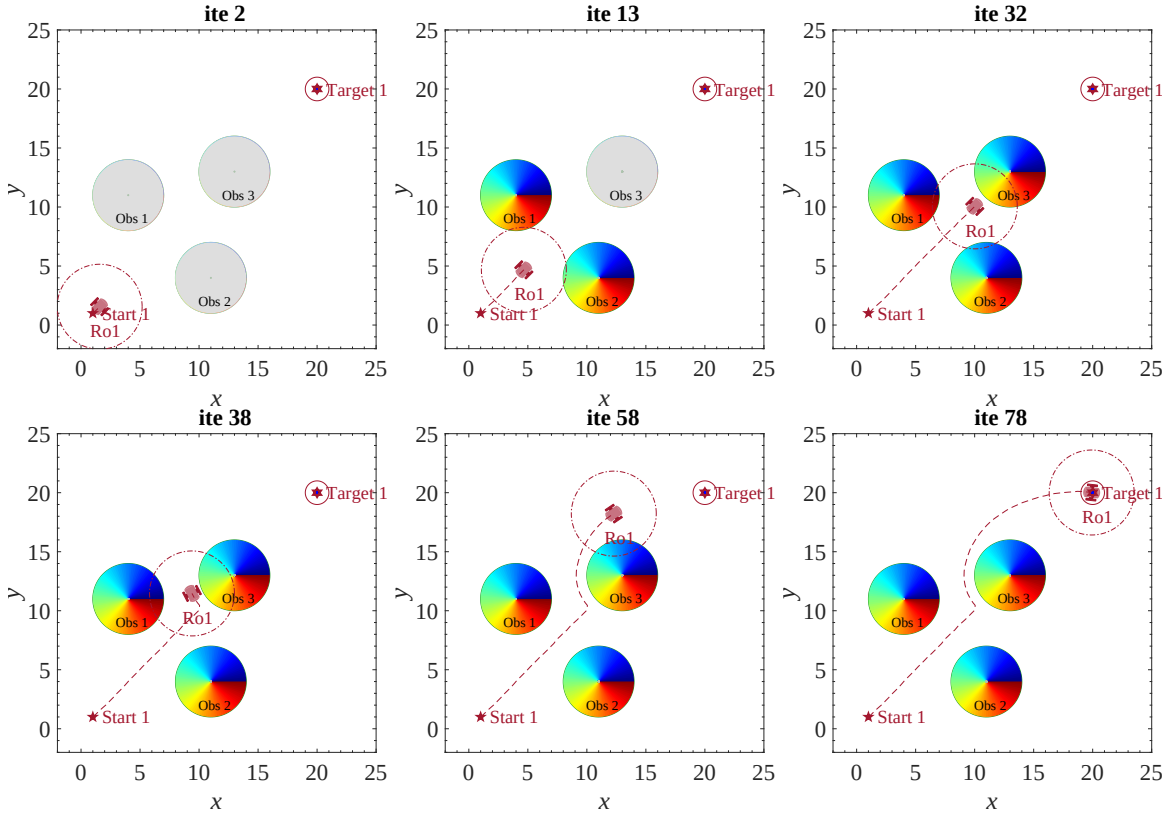


Figure 20: The movement process of the robot in Map 1: move through the gaps between obstacles to hit the target.

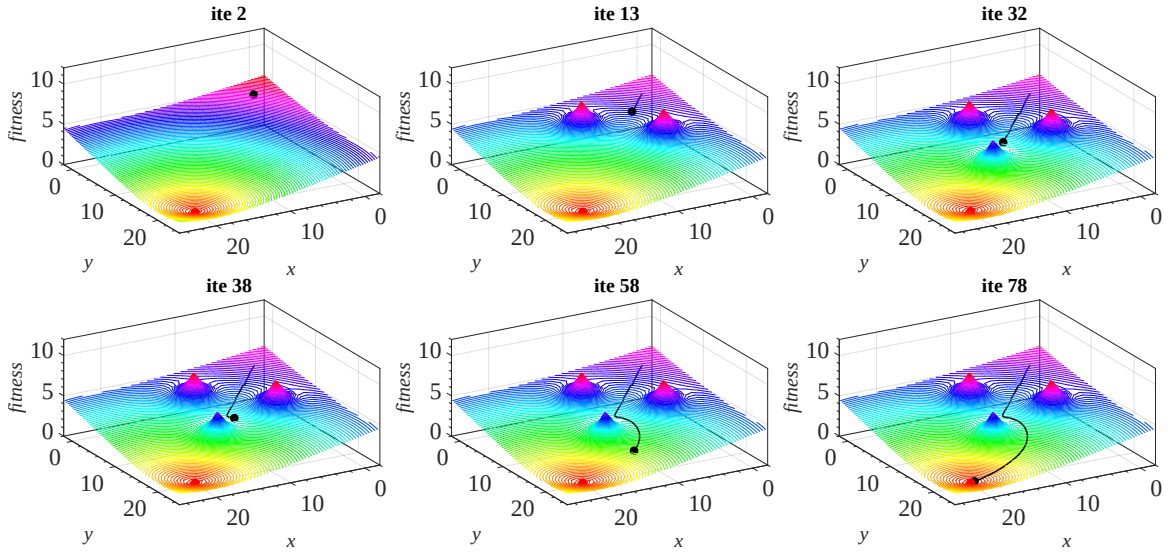


Figure 21: The movement process of the robot in Map 1 presented in 3D.

However, in steps 13^{th} and 32^{nd}, the obstacles are detected, and they have changed color, hills appear respectively in the 3D contour maps. The robot will move along these contour lines from high to low and avoid colliding on the rising hills (which are obstacles).

In the simple situation of Map 1, the distance between obstacles is large enough for the robot

to pass, so the robot has no trapped between three obstacles. The robot takes 78 steps to hit its target on this map.

6.5.2 For Map 2

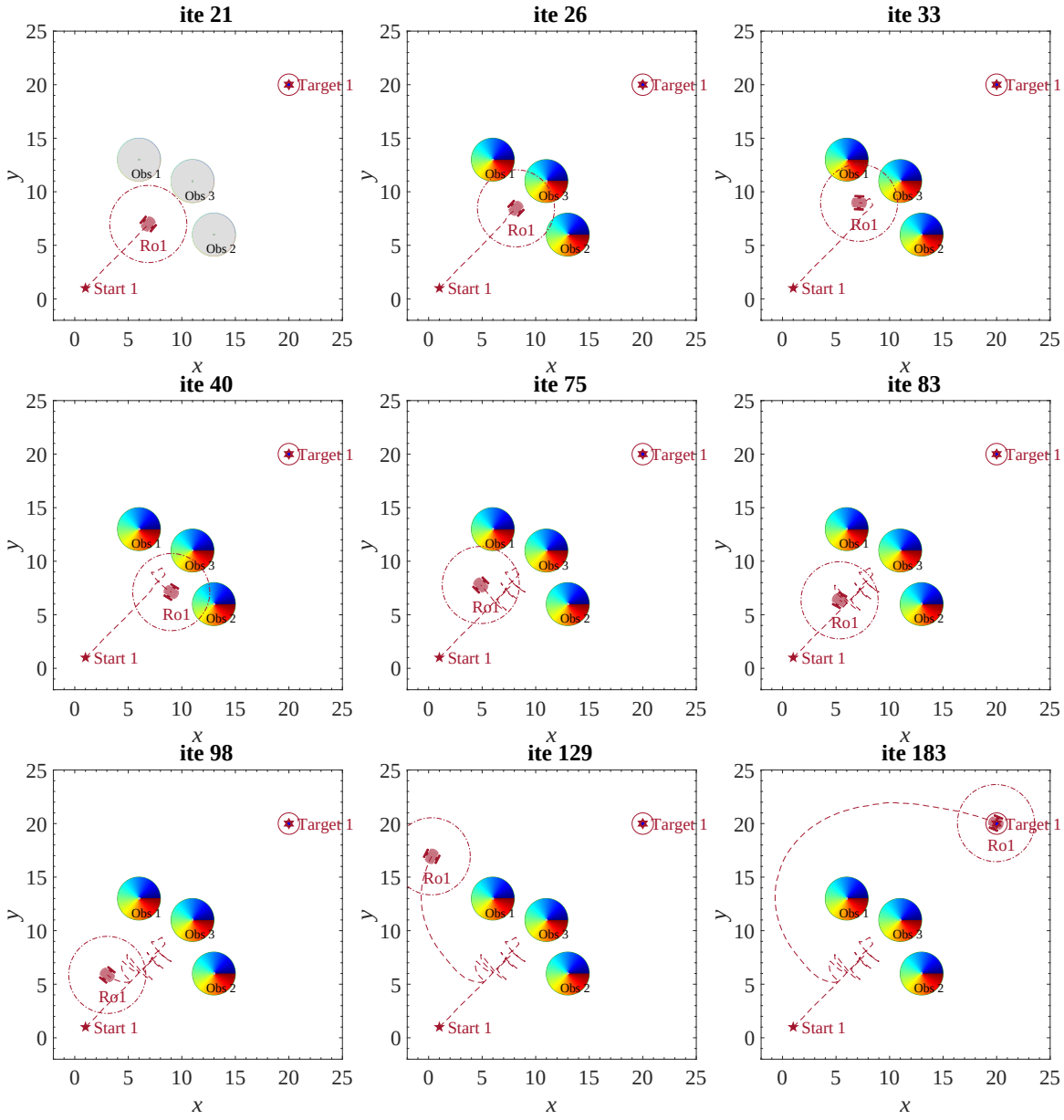


Figure 22: The movement process of the robot in Map 2: cannot move through the gaps, escape from the trap to hit the target.

Scenario 2 is intentionally arranged so that the distance between obstacles is not enough for the robot to move through. In this situation, the robot will be trapped between obstacles and will not be able to move out of the trap zone. To solve this problem, the equilibrium coefficient will change the value leading to a change the width of the hill accordingly, thereby escaping the robot from the trapped area [59].

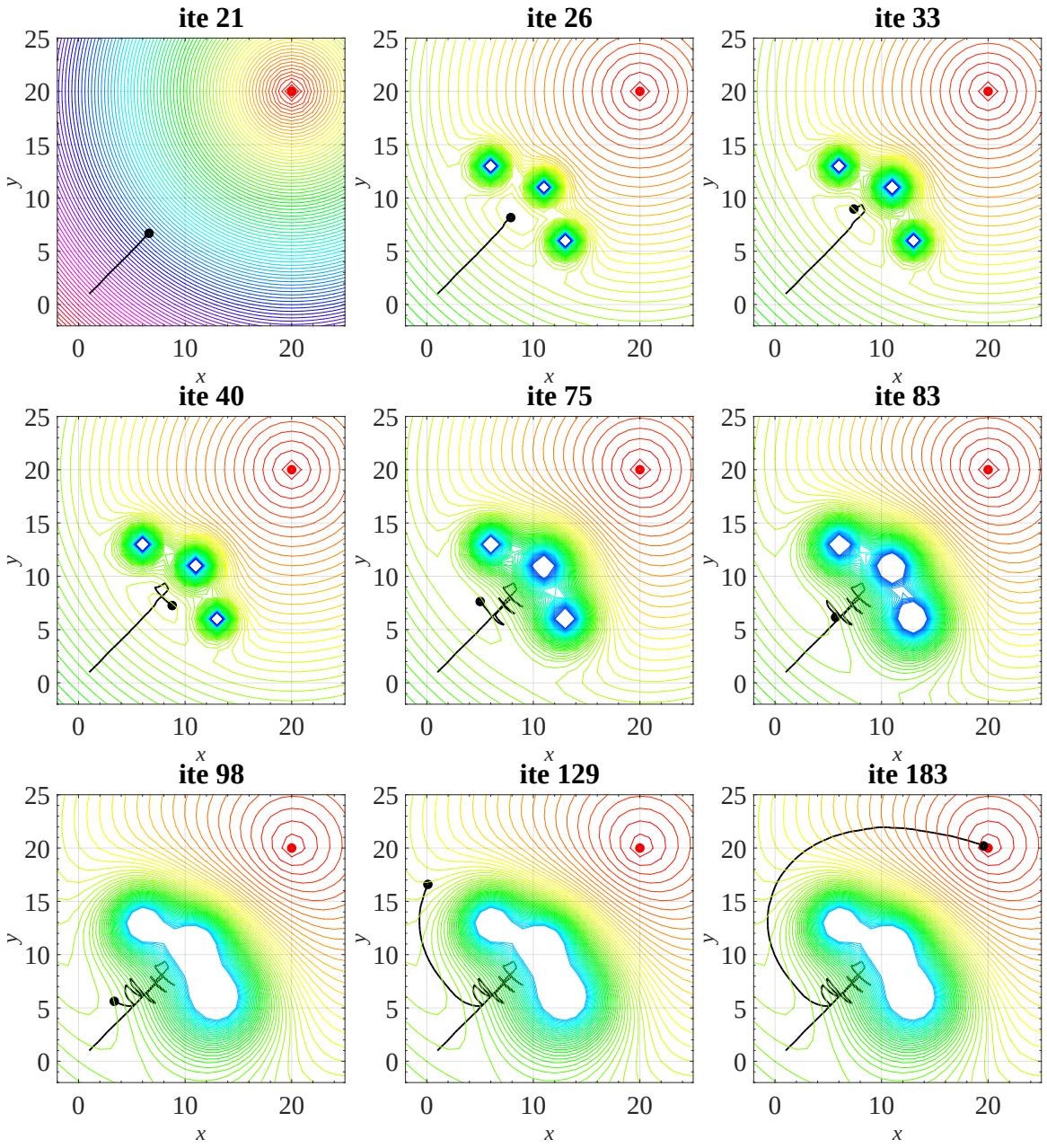


Figure 23: The movement process of the robot in Map 2 presented in contour map.

Figs. 22 and 23 show the entire operation of the robot, captured at steps 21^{st}, 26^{th}, 33^{rd}, 40^{th}, 75^{th}, 83^{rd}, 98^{th}, 129^{th}, and 183^{rd}.

In step 21^{st}, similar to map 1, the robot has not detected any obstacles so it moves straight to the target. However, in step 26^{th}, all three obstacles were detected, at which time the robot was trapped in the contour as shown in steps 26^{th} and 33^{rd}. As mentioned above, the equilibrium coefficients start to change, resulting in the size of the hills growing up, the contour changing continuously. The robot follows these contour lines, shown in steps 40^{th} to 98^{th}, and exits the trap towards the target, shown in steps 129^{th}, and 183^{rd}.

The robot took 183 steps in this scenario to escape the trap and hit the target.

6.5.3 For Map 3

Different from map 1 and map 2, map 3 has two robots and two targets, respectively. There are no obstacles on this map. Instead, the positions of the robot and the targets are intentionally arranged so that they are each other's obstacles.

Fig. 24 shows the movement of two robots, captured at steps 5^{th}, 16^{th}, 19^{th}, 25^{th}, 30^{th}, and 53^{rd}. At step 5^{th}, the robots have not detected the other robot yet so they move straight to their target. But in step 16^{th}, both robots are in the detection range of sensors, and they avoid each other as shown in steps 19^{th} to 30^{th}. Finally, they finish their work on step 53^{rd}.

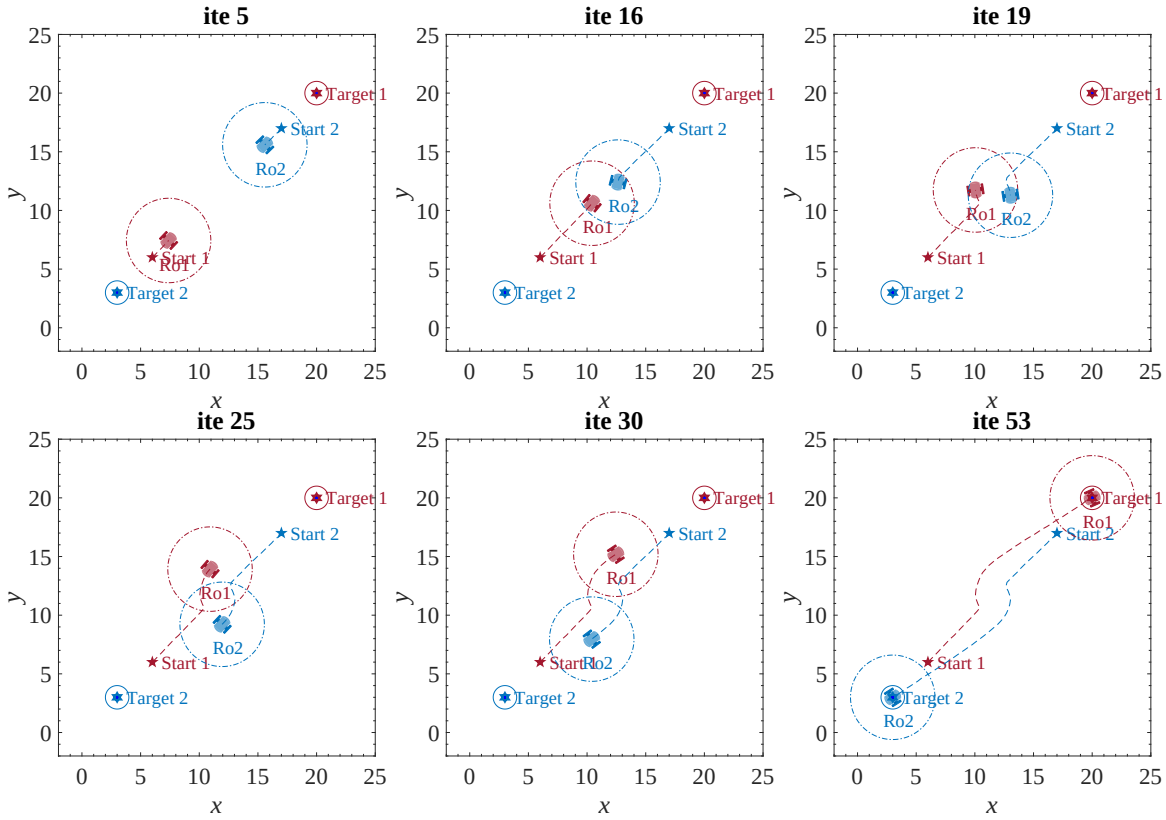


Figure 24: The movement process of the robot in Map 3: face-to-face between two robots.

6.5.4 For Map 4

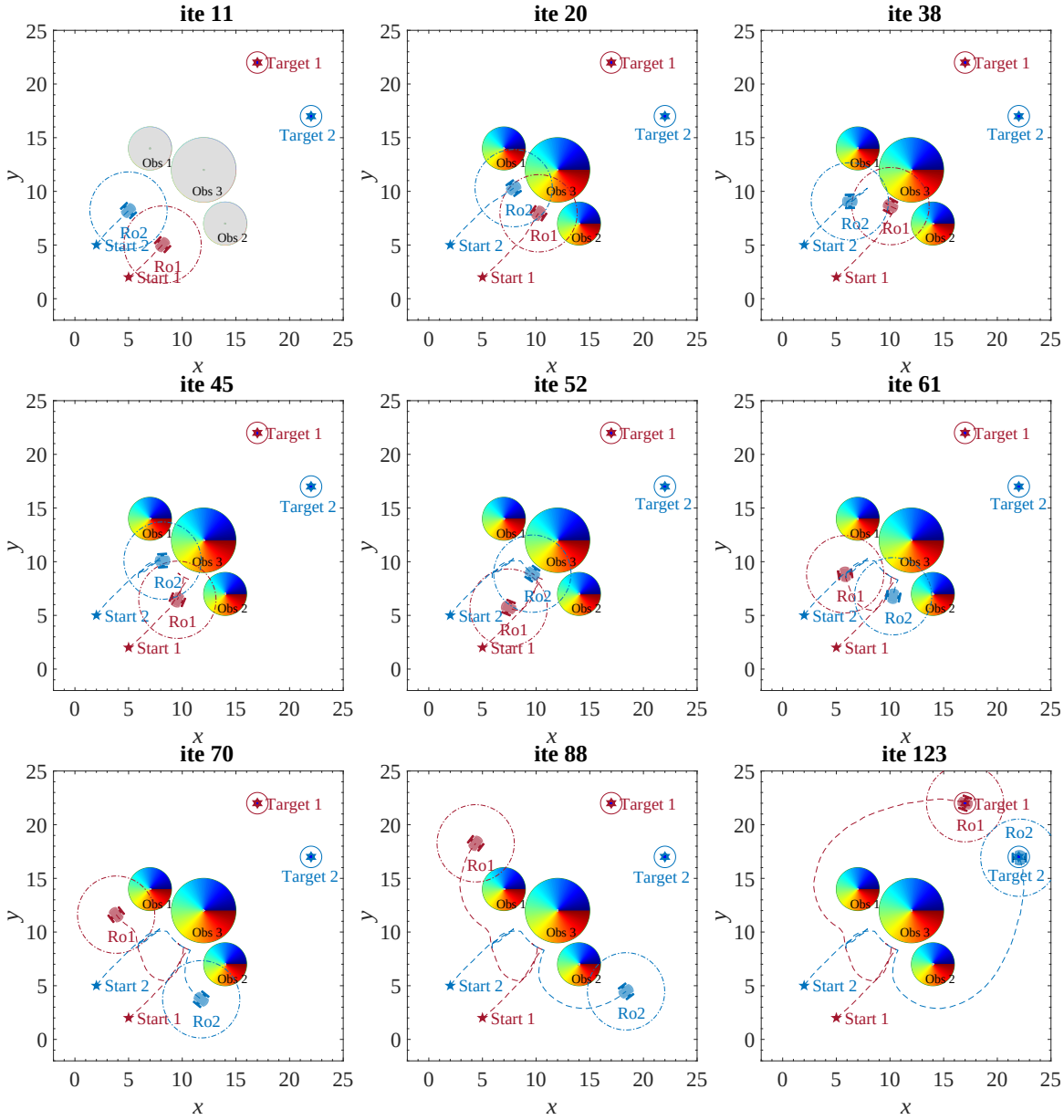


Figure 25: The movement process in Map 4: a complex combination in a single scenario.

The last scenario is the most complex one to test the general operability of robots. Two robots, two respective targets, and three obstacles are present on this map. They are arranged so that robots will be stuck between obstacles and the remaining robot will be another obstacle, moving around, preventing each other's path.

Fig. 25 reveals this operation process, captured at steps of 11^{th}, 20^{th}, 38^{th}, 45^{th}, 52^{nd}, 61^{st}, 70^{th}, 88^{th}, and 123^{rd}.

At step 11^{th}, three obstacles have not been detected and the two robots are not in each other's path so they move towards the targets. However, in step 20^{th}, three obstacles are blocking the way of both robots. Furthermore, the remaining robot now becomes the fourth obstacle

preventing the other robot's path. In step 38^{th}, Robot 2 turned its head, moved backward to find a way out of the trap zone, and Robot 1 moved along obstacles.

In steps 45^{th} to 61^{st}, the robots move around in the trap to find a way to escape, and they start out of the trap in step 70^{th}. Once out of the trap zone, there are no obstacles left, so the robots approach their target without any problem, as shown in step 88^{th}. They hit the final target at step 123^{rd}.

6.5.5 Unknown Complex Environment

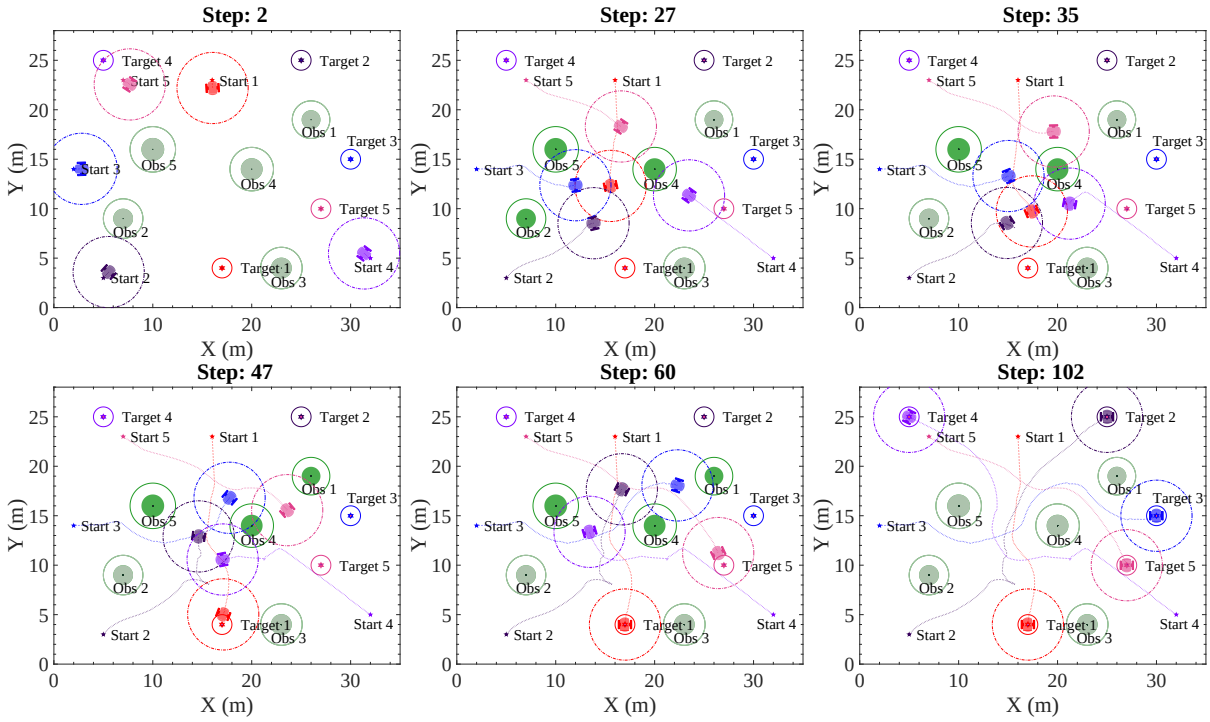


Figure 26: The movement of the multiple robot in an unknown static environment.

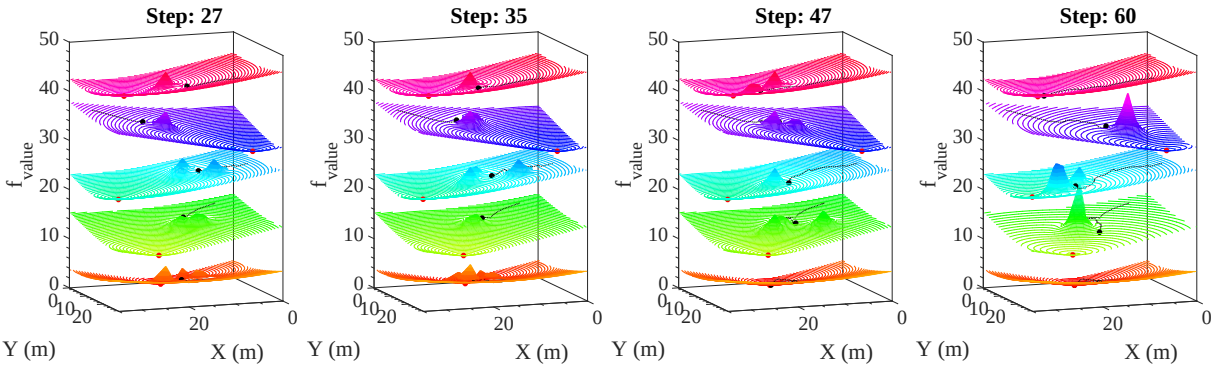


Figure 27: The detailed trajectory of the robot in the imaginary map.

Fig. 26 shows that the robots must avoid each other robots when they are close together besides avoid surrounding obstacles at the 27^{th} and 35^{th} moving steps. The 1^{st} robot detected the 4^{th} obstacle and the 2^{nd}, the 3^{rd} robots, and all of them are considered as dynamic obstacles,

leading to three small hills around the 1^{st} robot and it moved away from them to reach the target, as shown in the 27^{th} moving step in the lowest layer of Fig. 27. This happened in the same way for the other robots until all of them reached their target. The 4^{th} robot took the longest distance to reach its target, which was 102 moving steps, and the 1^{st} robot with the shortest distance took 50 moving steps. The 2^{nd}, the 3^{rd}, and the 5^{th} robots reach their targets at the 90^{th}, the 84^{th}, and the 64^{th} moving steps respectively.

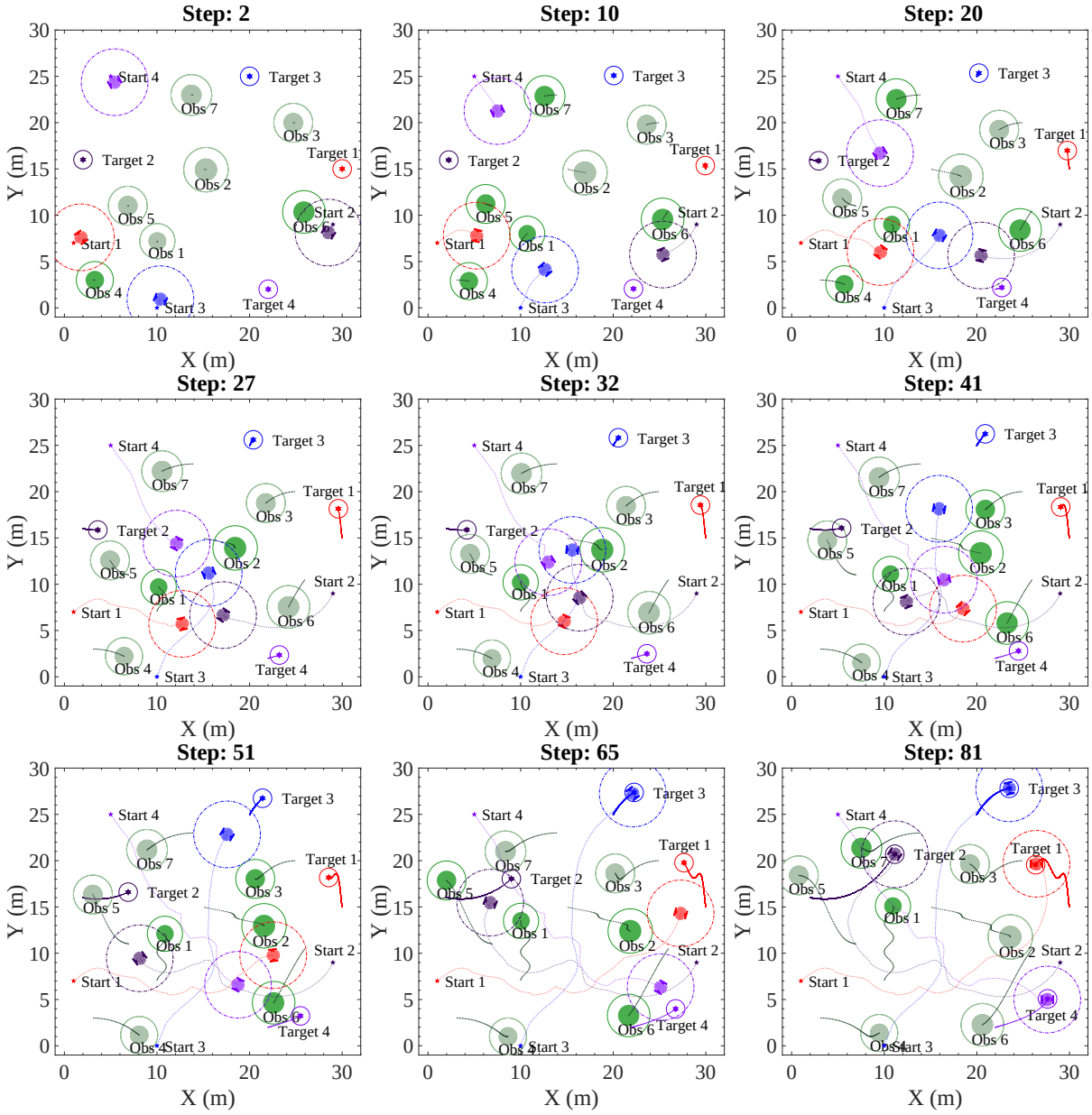


Figure 28: The moving process of the robots in the complex environment.

Fig. 27 shows the movement of the robot in form imaginary map, with five layers corresponding to five robots, the lowest layer indicating the 1^{st} robot, the highest layer indicating the 5^{th} robot respectively. When other robots or obstacles are detected, small hills pop up in the corresponding layer. As can be seen in these layers, the robot is like a spherical ball rolling

from the highest position, around these small hills, to the lowest position, as well as the robot reached its target.

Thus, all robots have completed their tasks without colliding with each other as well as colliding with obstacles in the second map.

In the last map, a complex environment was built in which all objects moved, as shown in Fig. 28. During the 27^{th} to 41^{st} moving steps, the movement of the robots became complicated as the robots were close together and surrounded by the 1^{st}, 2^{nd}, 3^{rd}, and 6^{th} obstacles. However, all robots did not collide with other robots and obstacles to move towards their target safely.

6.6 The Movement of Drones

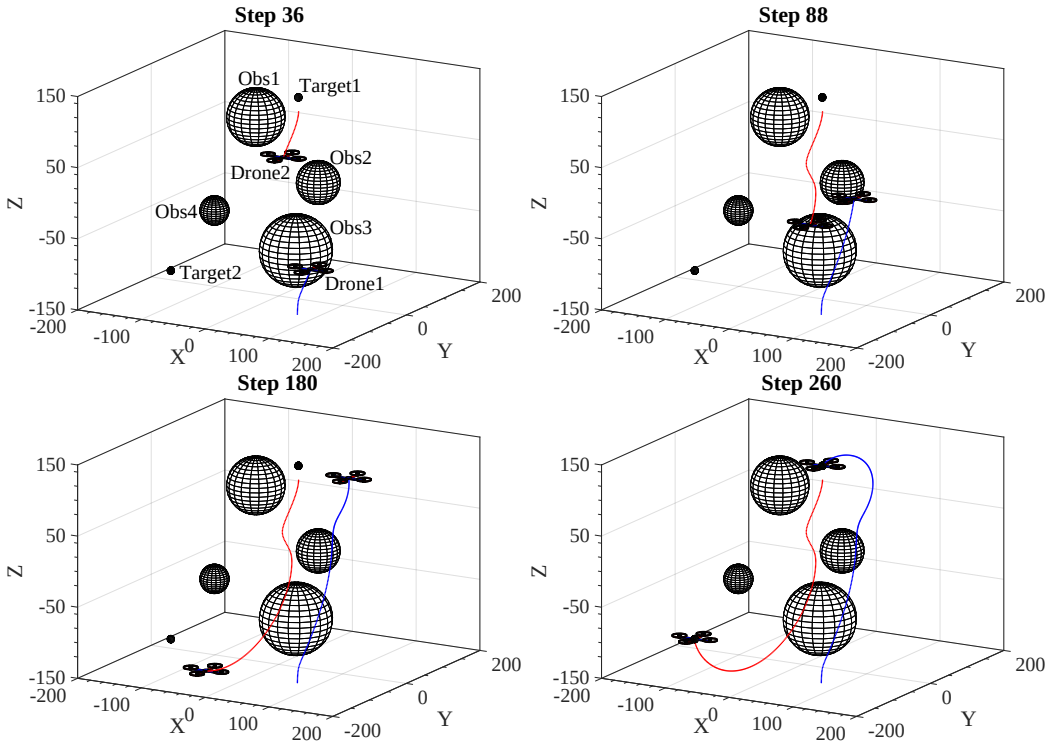


Figure 29: The results of simulating the paths of drones and avoiding obstacles in form of 3D.

Fig. 29 shows the simulation results of the flighting paths of drones, where drone 1 will start from a given initial position and catch the target 1, and drone 2 will catch the target 2. For each drone, there are five obstacles, including four given static obstacles in a spherical shape and the rest one is the other drone. The blue and red dotted lines represent the paths of drone 1 and 2, respectively. These figures were captured at the step 36^{th}, 88^{th}, 180^{th} and 260^{th}. Fig. 30, 31, and 32 present the paths of the drones at different views in more detail, X-Y, X-Z, and Y-Z view, respectively.

At the beginning of the algorithm, the position of the target is provided to the drone, and no obstacles are detected. Therefore, in the fitness function, there is only the first element, acting as a magnet to attract the drone to move towards the target. The predicted position

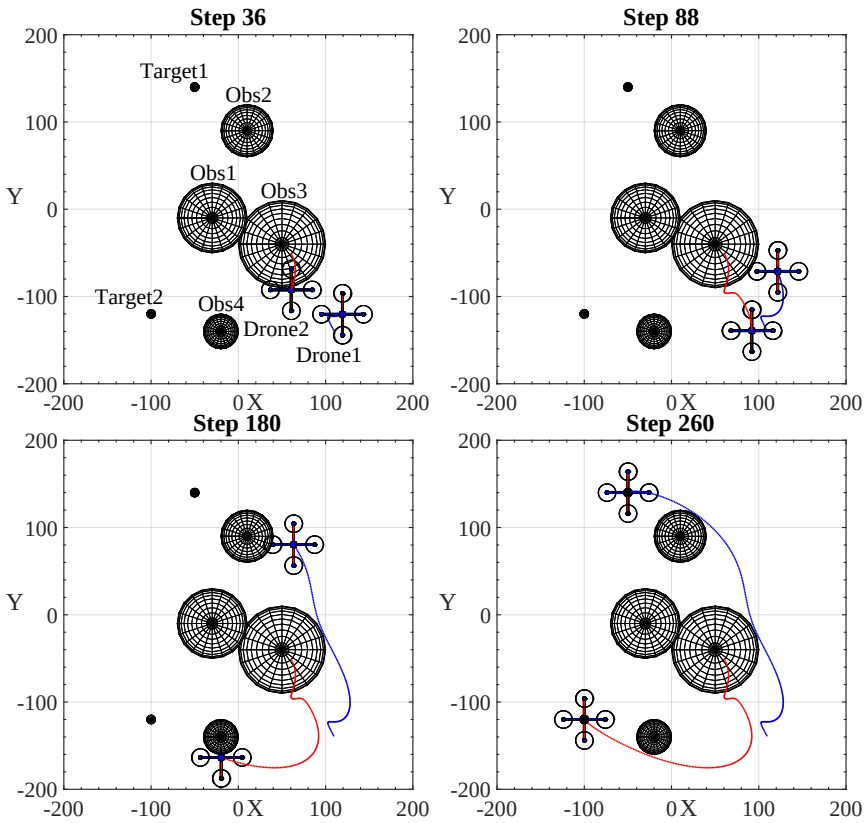


Figure 30: Simulation of the drone's path in X-Y view.

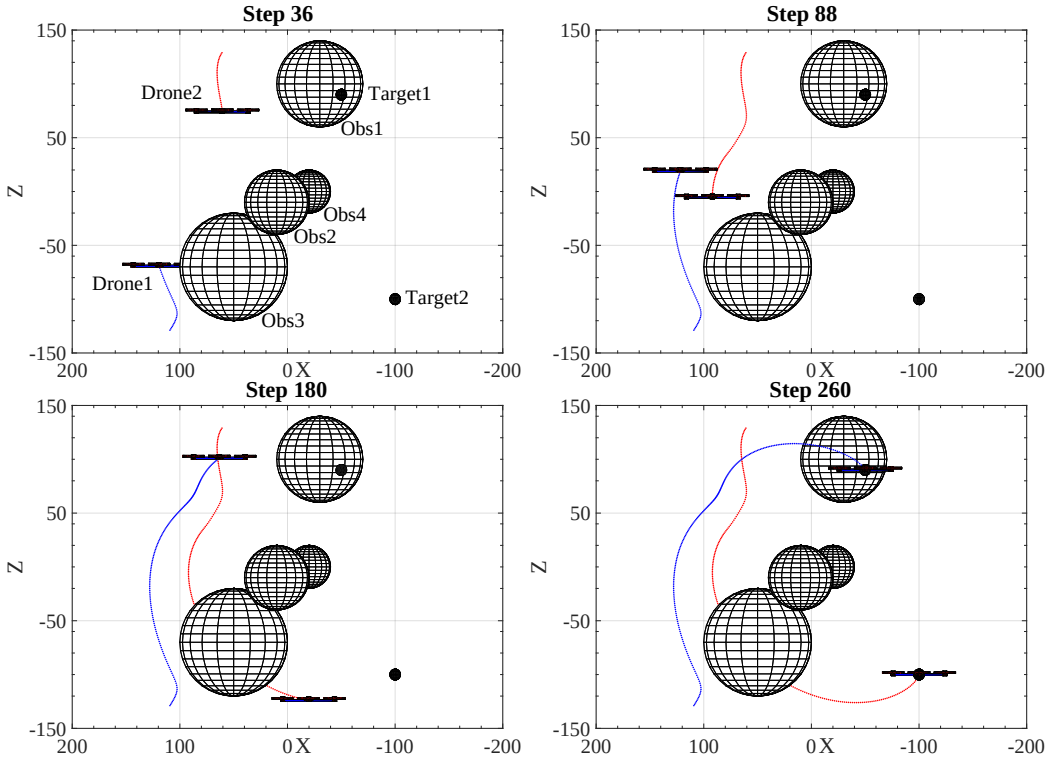


Figure 31: Simulation of the drone's path in X-Z view.

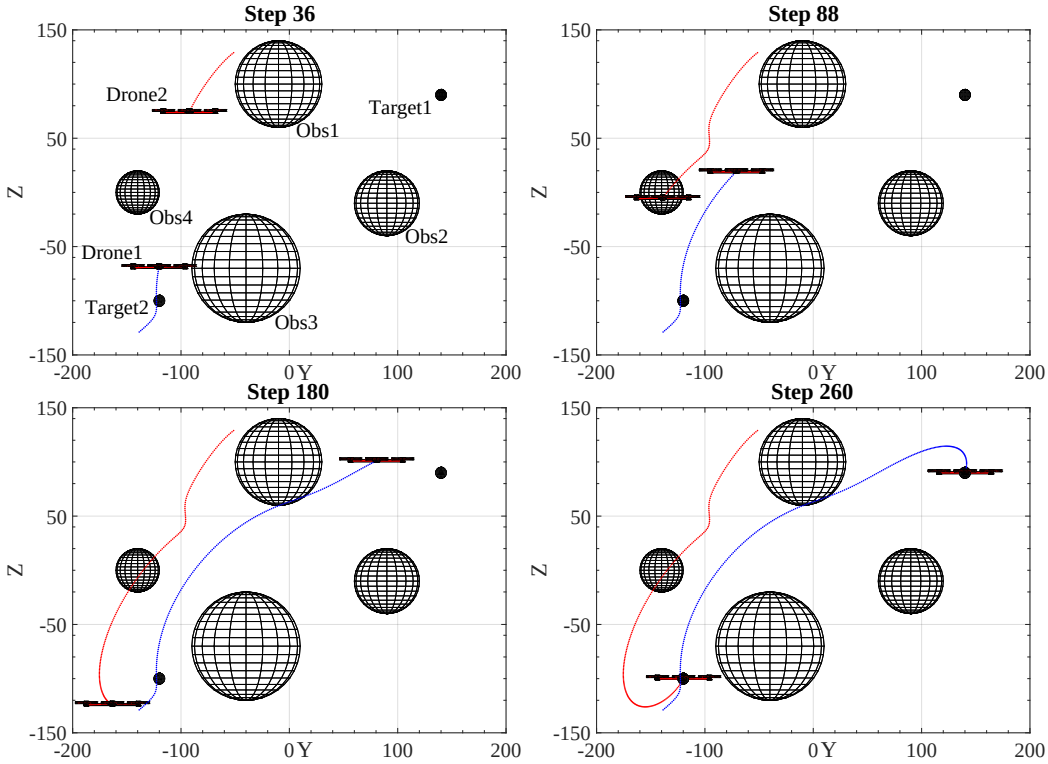


Figure 32: Simulation of the drone's path in Y-Z view.

that the drone will move to is created by the iSOMA algorithm. This position satisfies both the requirement of the drone's limited step and the minimum distance to the target.

At the step 36^{th}, drone 1 detected obstacle 3 and drone 2 detected obstacles 1 and 2. At this point, the second component in Eq. 12 appeared. This component makes the drone stay away from obstacles because if it moves closer to the obstacles, the value of the fitness function will increase. So the next predicted position provided by SOMA will both avoid obstacles and shorten the distance to the target.

The process continues as shown at the step 88^{th} and 180^{th} of Fig. 29 until the drones accomplished their purpose of catching targets at the step 260^{th}.

From these Figs. 29, 30, 31, and 32, it can be seen that the drone's paths change when obstacles were detected. Drone 1 and drone 2 have accomplished the goal of catching targets 1 and 2 without colliding with each other as well as hitting obstacles from 1 to 4. This fact proves the correctness and effectiveness of the proposed method.

7 CONCLUSION

The study pointed out the important role of swarm intelligence in solving optimization problems, analyzed the advantages and disadvantages of the SI that SOMA is a representative of, proposed novel versions, evaluated them on well-known test suites, compared their performance to well-known algorithms, and applied them to solve practical problems in swarm robotics.

Optimization is, in fact, ubiquitous in industrial practice as well as academic problems. But not any optimization problem can be solved by conventional mathematical methods. In that circumstance, swarm intelligence was extremely effective at solving them by simulating the intelligent behavior of creatures, something alien to traditional mathematics but is a trend of artificial intelligence, and SOMA is a brilliant representative.

There is no doubt in concluding that SOMA is a strong representation of the SI because of its accomplishments. Let's take CEC 2019 as a demonstration. One can see that the most obvious characteristic in the CEC 2019 competition is the dominance of DE-based algorithms, one of the most powerful representations in years of the Evolutionary algorithms class. Only the isolated SOMA algorithm that ranked in the top 3 of the strongest algorithms among the other competitors are DE variants, while the other representatives of SI do not do the same thing.

Specifically, the main goals that the study achieved against the initial expectations are shown as follows:

- State of the art in the research field, published in [rel13], [rel15], [rel16].
- Analysis and design the novel algorithms to deal with well-known test suites of CEC 2013, 2015, 2017, 2019, and real-world problems, including:
  - Self-organizing migrating algorithm team to team adaptive, named SOMA T3A, already published in [rel1].
  - Pareto-based self-organizing migrating algorithm, named SOMA Pareto, already published in [rel2].
  - Self-organizing migrating algorithm with narrowing search space strategy, named iSOMA [rel10].
- Determine the proposed algorithms position compared to well-known existing algorithms:
  - Evaluate the performance on the IEEE CEC 2013, 2015, 2017, 2019, already published in [rel1], [rel2], [rel3], [rel4], [rel10], [rel11], [rel12].
  - Assert the ranking against various types of algorithms, even evolutionary algorithms or swarm intelligence, already published in [rel1], [rel2], [rel10], [rel11].
- Apply the proposed algorithms to swarm robotics
  - The movement of swarm robots, already published in [rel5], [rel6], [rel7], [rel8],
  - Unmanned aerial vehicles control, already published in [rel8], [rel10].

References

[1] Javier Del Ser et al. “Bio-inspired computation: Where we stand and what’s next”. In: *Swarm and Evolutionary Computation* 48 (2019), pp. 220–250.

[2] Wei-feng Gao and San-yang Liu. “A modified artificial bee colony algorithm”. In: *Computers and Operations Research* 39.3 (2012), pp. 687 –697. ISSN: 0305-0548.

[3] Dervis Karaboga and Bahriye Basturk. “Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems”. In: *Foundations of Fuzzy Logic and Soft Computing*. Ed. by Patricia Melin et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 789–798. ISBN: 978-3-540-72950-1.

[4] J. Kennedy and R. Eberhart. “Particle swarm optimization”. In: *Proceedings of ICNN’95 - International Conference on Neural Networks*. Vol. 4. 1995, 1942–1948 vol.4.

[5] Jagdish Chand Bansal. “Particle Swarm Optimization”. In: *Evolutionary and Swarm Intelligence Algorithms*. Ed. by Jagdish Chand Bansal, Pramod Kumar Singh, and Nikhil R. Pal. Cham: Springer International Publishing, 2019, pp. 11–23. ISBN: 978-3-319-91341-4.

[6] Ivan Zelinka. “SOMA – Self-Organizing Migrating Algorithm”. In: *New Optimization Techniques in Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 167–217. ISBN: 978-3-540-39930-8.

[7] Ivan Zelinka. “SOMA—Self-organizing Migrating Algorithm”. In: *Self-Organizing Migrating Algorithm: Methodology and Implementation*. Ed. by Donald Davendra and Ivan Zelinka. Cham: Springer International Publishing, 2016, pp. 3–49. ISBN: 978-3-319-28161-2.

[8] Donald Davendra, Ivan Zelinka, et al. “Self-organizing migrating algorithm”. In: *New optimization techniques in engineering* (2016).

[9] Lukáš Tomaszek and Ivan Zelinka. “Conversion of soma algorithm into complex networks”. In: *Evolutionary Algorithms, Swarm Dynamics and Complex Networks*. Springer, 2018, pp. 101–114.

[10] Ivan Zelinka et al. “Do evolutionary algorithms dynamics create complex network structures?” In: *Complex Systems* 20.2 (2011), p. 127.

[11] Ivan Zelinka and Guanrong Chen. *Evolutionary Algorithms, Swarm Dynamics and Complex Networks: Methodology, Perspectives and Implementation*. Vol. 26. Springer, 2017.

[12] Lenka Skanderova, Tomas Fabian, and Ivan Zelinka. “Self-adapting self-organizing migrating algorithm”. In: *Swarm and Evolutionary Computation* 51 (2019), p. 100593. ISSN: 2210-6502.

[13] Dipti Singh, Seema Agrawal, and Kusum Deep. “C-SOMAQI: Self Organizing Migrating Algorithm with Quadratic Interpolation Crossover Operator for Constrained Global Optimization”. In: *Self-Organizing Migrating Algorithm: Methodology and Implementation*. Ed. by Donald Davendra and Ivan Zelinka. Cham: Springer International Publishing, 2016, pp. 147–165. ISBN: 978-3-319-28161-2.

[14] Lukas Tomaszek, Ivan Zelinka, and Mohammed Chadli. “On the Leader Selection in the Self-Organizing Migrating Algorithm”. In: *MENDEL* 25.1 (2019), pp. 171–178.

[15] Michal Pluhacek et al. “Self-organizing Migrating Algorithm with Non-binary Perturbation”. In: *Swarm, Evolutionary, and Memetic Computing and Fuzzy and Neural Computing*. Ed. by Aleš Zamuda et al. Cham: Springer International Publishing, 2020, pp. 43–57. ISBN: 978-3-030-37838-7.

[16] T. Kadavy et al. “Introducing Self-Adaptive Parameters to Self-organizing Migrating Algorithm”. In: *2019 IEEE Congress on Evolutionary Computation (CEC)*. 2019, pp. 2908–2914.

[17] Quoc Bao Diep. “Self-organizing migrating algorithm Team To Team adaptive–SOMA T3A”. In: *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2019, pp. 1182–1187.

[18] Quoc Bao Diep et al. “SOMA T3A for Solving the 100-Digit Challenge”. In: *Swarm, Evolutionary, and Memetic Computing and Fuzzy and Neural Computing*. Springer, 2019, pp. 155–165.

[19] Quoc Bao Diep, Ivan Zelinka, and Swagatam Das. “Self-organizing migrating algorithm pareto”. In: *Mendel*. Vol. 25. 1. 2019, pp. 111–120.

[20] Thanh Cong Truong et al. “Pareto-Based Self-organizing Migrating Algorithm Solving 100-Digit Challenge”. In: *Swarm, Evolutionary, and Memetic Computing and Fuzzy and Neural Computing*. Springer, 2019, pp. 13–20.

[21] Fernando Lezama et al. “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*. GECCO ’19. Prague, Czech Republic: Association for Computing Machinery, 2019, 7–8. ISBN: 9781450367486.

[22] KV Price et al. “The 2019 100-Digit Challenge on Real-Parameter, Single Objective Optimization: Analysis of Results”. In: 2019. URL: `https://github.com/P-N-Suganthan/CEC2019`.

[23] Ivan Zelinka et al. “Artificial Intelligence in Astrophysics”. In: *Intelligent Astrophysics*. Springer, 2021, pp. 1–28.

[24] Quoc Bao Diep, Thanh Cong Truong, and Ivan Zelinka. “Obstacle Avoidance for Drones Based on the Self-Organizing Migrating Algorithm”. In: *International Conference on Artificial Intelligence and Soft Computing*. Springer. 2020, pp. 376–386.

[25] Quoc Bao Diep and Ivan Zelinka. “The Movement of Swarm Robots in an Unknown Complex Environment”. In: *International Conference on Advanced Engineering Theory and Applications*. Springer. 2018, pp. 949–959.

[26] Slawomir Koziel and Xin-She Yang. *Computational optimization, methods and algorithms*. Vol. 356. Springer, 2011.

[27] Daniel Câmara. “1 - Evolution and Evolutionary Algorithms”. In: *Bio-inspired Networking*. Ed. by Daniel Câmara. Elsevier, 2015, pp. 1–30. ISBN: 978-1-78548-021-8.

[28] G Beni and J Wang. “Swarm Intelligence in Cellular Robotic Systems, Proceed. NATO Advanced Workshop on Robots and Biological Systems, Tuscany, Italy, June 26-30”. In: *NY: NATO* (1989).

[29] Peter Swirski. “Of Games with the Universe: Preconceptions of Science in Stanislaw Lem's" The Invincible"”. In: *Contemporary Literature* 35.2 (1994), pp. 324–342.

[30] Andries P Engelbrecht. *Computational intelligence: an introduction*. John Wiley & Sons, 2007.

[31] Marco Dorigo and Gianni Di Caro. “Ant colony optimization: a new meta-heuristic”. In: *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*. Vol. 2. IEEE. 1999, pp. 1470–1477.

[32] James Kennedy and Russell Eberhart. “Particle swarm optimization”. In: *Proceedings of ICNN'95-international conference on neural networks*. Vol. 4. IEEE. 1995, pp. 1942–1948.

[33] Dervis Karaboga and Bahriye Akay. “A comparative study of artificial bee colony algorithm”. In: *Applied mathematics and computation* 214.1 (2009), pp. 108–132.

[34] Kevin M Passino. “Biomimicry of bacterial foraging for distributed optimization and control”. In: *IEEE control systems magazine* 22.3 (2002), pp. 52–67.

[35] Xin-She Yang et al. “Firefly algorithm”. In: *Nature-inspired metaheuristic algorithms* 20 (2008), pp. 79–90.

[36] Xin-She Yang. “A new metaheuristic bat-inspired algorithm”. In: *Nature inspired cooperative strategies for optimization (NICSO 2010)*. Springer, 2010, pp. 65–74.

[37] Seyedali Mirjalili and Andrew Lewis. “The whale optimization algorithm”. In: *Advances in engineering software* 95 (2016), pp. 51–67.

[38] Ivan Zelinka and Lampinen Jouni. “SOMA–Self-Organizing Migrating Algorithm Mendel”. In: *6th International Conference on Soft Computing, Brno, Czech Republic*. 2000.

[39] Kusum Deep and Dipti. “A self-organizing migrating genetic algorithm for constrained optimization”. In: *Applied Mathematics and Computation* 198.1 (2008), pp. 237–250.

[40] Kusum Deep et al. “A new hybrid self organizing migrating genetic algorithm for function optimization”. In: *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*. IEEE. 2007, pp. 2796–2803.

[41] Leandro dos Santos Coelho and Viviana Cocco Mariani. “An efficient cultural self-organizing migrating strategy for economic dispatch optimization with valve-point effect”. In: *Energy Conversion and Management* 51.12 (2010), pp. 2580–2587.

[42] Dipti Singh and Seema Agrawal. “Hybridization of self organizing migrating algorithm with quadratic approximation and non uniform mutation for function optimization”. In: *Proceedings of Fourth International Conference on Soft Computing for Problem Solving*. Springer. 2015, pp. 373–387.

[43] Seema Agrawal and Dipti Singh. “Modified Nelder-Mead self organizing migrating algorithm for function optimization and its application”. In: *Applied Soft Computing* 51 (2017), pp. 341–350.

[44] Zhiyi Lin and Li juan Wang. “Hybrid self-organizing migrating algorithm based on estimation of distribution”. In: *2014 International Conference on Mechatronics, Electronic, Industrial and Control Engineering (MEIC-14)*. Atlantis Press. 2014.

[45] Dipti Singh and Seema Agrawal. “Self organizing migrating algorithm with quadratic interpolation for solving large scale global optimization problems”. In: *Applied Soft Computing* 38 (2016), pp. 1040–1048.

[46] Ivan Zelinka et al. “Impact of chaotic dynamics on the performance of metaheuristic optimization algorithms: An experimental analysis”. In: *Information Sciences* (2021). ISSN: 0020-0255.

[47] Md Arafat Hossain and Israt Ferdous. “Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique”. In: *Robotics and Autonomous Systems* 64 (2015), pp. 137–141.

[48] Oscar Montiel, Ulises Orozco-Rosas, and Roberto Sepúlveda. “Path planning for mobile robots using Bacterial Potential Field for avoiding static and dynamic obstacles”. In: *Expert Systems with Applications* 42.12 (2015), pp. 5177–5191.

[49] Abdulmuttalib Turky Rashid et al. “Path planning with obstacle avoidance based on visibility binary tree algorithm”. In: *Robotics and Autonomous Systems* 61.12 (2013), pp. 1440–1449.

[50] Quoc Bao Diep et al. “Self-organizing migrating algorithm with narrowing search space strategy for robot path planning”. In: *Applied Soft Computing* (2021). ISSN: 1568-4946.

[51] Vilfredo Pareto. *Cours d’économie politique*. Vol. 1. Librairie Droz, 1964.

[52] Yoram Koren and Johann Borenstein. “Potential field methods and their inherent limitations for mobile robot navigation”. In: *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*. IEEE. 1991, pp. 1398–1404.

[53] Johann Borenstein and Yoram Koren. “The vector field histogram-fast obstacle avoidance for mobile robots”. In: *IEEE transactions on robotics and automation* 7.3 (1991), pp. 278–288.

[54] Volkan Sezer and Metin Gokasan. “A novel obstacle avoidance algorithm:“Follow the Gap Method””. In: *Robotics and Autonomous Systems* 60.9 (2012), pp. 1123–1134.

[55] Yuxiao Chen, Huei Peng, and Jessy Grizzle. “Obstacle avoidance for low-speed autonomous vehicles with barrier function”. In: *IEEE Transactions on Control Systems Technology* 99 (2017), pp. 1–13.

[56] Yanrong Hu and Simon X Yang. “A knowledge based genetic algorithm for path planning of a mobile robot”. In: *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*. Vol. 5. IEEE. 2004, pp. 4350–4355.

[57] Mihai Duguleana and Gheorghe Mogan. “Neural networks based reinforcement learning for mobile robots obstacle avoidance”. In: *Expert Systems with Applications* 62 (2016), pp. 104–115.

[58] Diep Quoc Bao and Ivan Zelinka. “Obstacle Avoidance for Swarm Robot Based on Self-Organizing Migrating Algorithm”. In: *Procedia Computer Science* 150 (2019), pp. 425–432.

[59] Quoc Bao Diep, Ivan Zelinka, and Roman Senkerik. “An algorithm for swarm robot to avoid multiple dynamic obstacles and to catch the moving target”. In: *International Conference on Artificial Intelligence and Soft Computing*. Springer. 2019, pp. 666–675.

[60] Quoc Bao Diep, Thanh Cong Truong, and Ivan Zelinka. “Swarm intelligence and swarm robotics in the path planning problem”. In: *Modern Trends in Controlled Stochastic Processes:* Springer, 2021, pp. 313–327.

[61] Yucong Lin and Srikanth Saripalli. “Sampling-based path planning for UAV collision avoidance”. In: *IEEE Transactions on Intelligent Transportation Systems* 18.11 (2017), pp. 3179–3192.

[62] Peng Yao, Honglun Wang, and Zikang Su. “Cooperative path planning with applications to target tracking and obstacle avoidance for multi-UAVs”. In: *Aerospace science and technology* 54 (2016), pp. 10–22.

[63] Tauã M Cabreira et al. “Grid-Based Coverage Path Planning With Minimum Energy Over Irregular-Shaped Areas With Uavs”. In: *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2019, pp. 758–767.

[64] JJ Liang et al. “Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization”. In: *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report* 201212 (2013), pp. 3–18.

[65] JJ Liang et al. “Problem definitions and evaluation criteria for the CEC 2015 competition on learning-based real-parameter single objective optimization”. In: *Technical Report201411A, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore* (2014).

[66] NH Awad et al. “Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective real-parameter numerical optimization”. In: *Tech. Rep.* (2016).

[67] K. V. Price et al. “Problem Definitions and Evaluation Criteria for the 100-Digit Challenge Special Session and Competition on Single Objective Numerical Optimization”. In: *Technical Report, Nanyang Technological University, Singapore*. November, 2018.

[68] Quoc Bao Diep, Ivan Zelinka, and Swagatam Das. “Self-Organizing Migrating Algorithm for the 100-Digit Challenge”. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO ’19. Prague, Czech Republic: Association for Computing Machinery, 2019, 3–4. ISBN: 9781450367486.

[69] Bahriye Akay and Dervis Karaboga. “A modified artificial bee colony algorithm for real-parameter optimization”. In: *Information sciences* 192 (2012), pp. 120–142.

[70] Marjan Mernik et al. “On clarifying misconceptions when comparing variants of the Artificial Bee Colony Algorithm by offering a new implementation”. In: *Information Sciences* 291 (2015), pp. 115–127.

[71] Niki Veček et al. “On the importance of the artificial bee colony control parameter ‘Limit’”. In: *Information Technology And Control* 46.4 (2017), pp. 566–604.

[72] İbrahim Berkan Aydilek. “A hybrid firefly and particle swarm optimization algorithm for computationally expensive numerical problems”. In: *Applied Soft Computing* 66 (2018), pp. 232–249.

[73] Seyedali Mirjalili et al. “Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems”. In: *Advances in Engineering Software* 114 (2017), pp. 163–191.

[74] Ravipudi Venkata Rao. *Jaya: An Advanced Optimization Algorithm and its Engineering Applications*.

[75] Jacinto Carrasco et al. “Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review”. In: *Swarm and Evolutionary Computation* 54 (2020), p. 100665.

[76] Joaquín Derrac et al. “A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms”. In: *Swarm and Evolutionary Computation* 1.1 (2011), pp. 3–18.

[77] Fabio Caraffini et al. “Super-fit multicriteria adaptive differential evolution”. In: *2013 IEEE Congress on Evolutionary Computation*. IEEE. 2013, pp. 1678–1685.

[78] Fabio Caraffini et al. “A CMA-ES super-fit scheme for the re-sampled inheritance search”. In: *2013 IEEE Congress on Evolutionary Computation*. IEEE. 2013, pp. 1123–1130.

[79] Saber M Elsayed, Ruhul A Sarker, and Tapabrata Ray. “Differential evolution with automatic parameter configuration for solving the CEC2013 competition on real-parameter optimization”. In: *2013 IEEE Congress on Evolutionary Computation*. IEEE. 2013, pp. 1932–1937.

[80] Filipe V Nepomuceno and Andries P Engelbrecht. “A self-adaptive heterogeneous pso for real-parameter optimization”. In: *2013 IEEE congress on evolutionary computation*. IEEE. 2013, pp. 361–368.

[81] Yu-Jun Zheng and Xiao-Bei Wu. “Tuning maturity model of ecogeography-based optimization on CEC 2015 single-objective optimization test problems”. In: *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2015, pp. 1018–1024.

[82] L. Chen et al. “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.

[83] JJ Liang et al. “A self-adaptive dynamic particle swarm optimizer”. In: *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2015, pp. 3206–3213.

[84] Chao Yu, Ling Chen Kelley, and Ying 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)*. IEEE. 2015, pp. 1106–1112.

[85] Rafał 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)*. IEEE. 2017, pp. 1489–1494.

[86] Andrea Tangherloni, Leonardo Rundo, and Marco 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)*. IEEE. 2017, pp. 1940–1947.

[87] Remya Kommadath and Prakash Kotecha. “Teaching learning based optimization with focused learning and its performance on CEC2017 functions”. In: *2017 IEEE congress on evolutionary computation (CEC)*. IEEE. 2017, pp. 2397–2403.

[88] Ravipudi Venkata Rao. *Jaya: an advanced optimization algorithm and its engineering applications*. Springer, 2019.

[89] R. Tanabe and A. Fukunaga. “Evaluating the performance of SHADE on CEC 2013 benchmark problems”. In: *2013 IEEE Congress on Evolutionary Computation*. 2013, pp. 1952–1959.

[90] F. Caraffini et al. “A CMA-ES super-fit scheme for the re-sampled inheritance search”. In: *2013 IEEE Congress on Evolutionary Computation*. 2013, pp. 1123–1130.

[91] 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.

[92] S. M. Elsayed, R. A. Sarker, and 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.

[93] N. Awad, M. Z. Ali, and 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.

[94] D. Jagodziński and J. Arabas. “A differential evolution strategy”. In: *2017 IEEE Congress on Evolutionary Computation (CEC)*. 2017, pp. 1872–1876.

[95] 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.

[96] A. Tangherloni, L. Rundo, and 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.

[97] D. Maharana, R. Kommadath, and 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.

[98] R. Kommadath and 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.

List of Publications

Author's Publications Related to the Thesis

[rel1] **Diep Q. B.** (2019). Self-organizing migrating algorithm Team To Team adaptive-SOMA T3A. In: *2019 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1182-1187). IEEE.

[rel2] **Diep Q. B.**, Zelinka I., and Das S. (2019). Self-organizing migrating algorithm pareto. In: *MENDEL* (Vol. 25, No. 1, pp. 111-120). (Q3, SJR 2020 = 0.22)

[rel3] **Diep Q. B.**, Zelinka I., Das S., and Senkerik R. (2020). SOMA T3A for Solving the 100-Digit Challenge. In: *Swarm, Evolutionary, and Memetic Computing and Fuzzy and Neural Computing. SEMCCO 2019, FANCCO 2019. Communications in Computer and Information Science*, vol 1092. Springer, Cham.

[rel4] **Diep Q. B.**, Zelinka I., and Das S. (2019). Self-organizing migrating algorithm for the 100-digit challenge. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO'19). Association for Computing Machinery*, New York, NY, USA, 3-4.

[rel5] **Diep Q. B.**, Truong T. C., and Zelinka I. (2021). Swarm Intelligence and Swarm Robotics in the Path Planning Problem. In: *Modern Trends in Controlled Stochastic Processes:. Emergence, Complexity and Computation*, vol 41. Springer, Cham.

[rel6] **Diep Q. B.**, and Zelinka I. (2020). The Movement of Swarm Robots in an Unknown Complex Environment. In: *Recent Advances in Electrical Engineering and Related Sciences: Theory and Application. AETA 2018. Lecture Notes in Electrical Engineering,* vol 554. Springer, Cham.

[rel7] **Diep Q. B.**, and Zelinka I. (2019). Obstacle avoidance for swarm robot based on selforganizing migrating algorithm. In: *Procedia Computer Science*, 150, 425-432.

[rel8] **Diep Q. B.**, Truong T. C., and Zelinka I. (2020). Obstacle Avoidance for Drones Based on the Self-Organizing Migrating Algorithm. In: *Artificial Intelligence and Soft Computing. ICAISC 2020. Lecture Notes in Computer Science*, vol 12415. Springer, Cham.

[rel9] **Diep Q. B.**, Zelinka I., and Senkerik R. (2019). An Algorithm for Swarm Robot to Avoid Multiple Dynamic Obstacles and to Catch the Moving Target. In: *Artificial Intelligence and Soft Computing. ICAISC 2019. Lecture Notes in Computer Science*, vol 11509. Springer, Cham.

[rel10] **Diep Q. B.**, Truong T. C., Das S., and Zelinka I. (2021). Self-Organizing Migrating Algorithm with Narrowing Search Space Strategy for Robot Path Planning. In: *Applied Soft Computing.* ISSN: 1568-4946. (Q1, IF 2020 = 6.725)

[rel11] Zelinka I., **Diep Q. B.**, Snasel V., Das S., Innocenti G., Tesi A., Schoen F., and Kuznetsov N. V. (2021). Impact of Chaotic Dynamics on the Performance of Metaheuristc Optimization Algorithms: an Experimental Analysis. In: *Information Sciences.* ISSN: 0020-0255. (Q1, IF 2020 = 6.795)

[rel12] Truong T. C., **Diep Q. B.**, Zelinka I., and Senkerik R. (2020). Pareto-Based Self-

organizing Migrating Algorithm Solving 100-Digit Challenge. In: *Swarm, Evolutionary, and Memetic Computing and Fuzzy and Neural Computing. SEMCCO 2019, FANCCO 2019. Communications in Computer and Information Science*, vol 1092. Springer, Cham.

[rel13] Zelinka I., Truong T. C., **Diep Q. B.**, Kojecky L., and Amer E. (2021). Artificial Intelligence in Astrophysics. In: *Intelligent Astrophysics. Emergence, Complexity and Computation*, vol 39. Springer, Cham.

[rel14] Senkerik R., Viktorin A., Kadavy T., Pluhacek M., Kazikova A., **Diep Q. B.**, and Zelinka I. (2019). Population Diversity Analysis in Adaptive Differential Evolution Variants with Unconventional Randomization Schemes. In: *Artificial Intelligence and Soft Computing. ICAISC 2019. Lecture Notes in Computer Science*, vol 11508. Springer, Cham.

[rel15] Truong T. C., **Diep Q. B.**, and Zelinka I. (2020). Artiffcial intelligence in the cyber domain: Offense and defense. *Symmetry*, 12(3), 410. (Q2, IF 2020 = 2.713)

[rel16] Truong T. C., **Diep Q. B.**, and Zelinka I. (2020). Swarm Intelligence in Cybersecurity. *Swarm Intelligence: From Social Bacteria to Humans 1st ed.* (pp. 90-107). CRC Press.

Other Publications of the Author

[oth1] **Diep Q. B.** (2017). Control of pipe cutting robot: A more effective method. *Journal of Advanced Engineering and Computation*, 1(2), 95-105.

[oth2] Truong T. C., **Diep Q. B.**, Zelinka I., and Dao T. T. (2020). X-Swarm: The Upcoming Swarm Worm. In *MENDEL* (Vol. 26, No. 1, pp. 7-14). (Q3, SJR 2020 = 0.22)

[oth3] Truong T. C., **Diep Q. B.**, Plucar J., and Zelinka I. (2021). X-ware: A proof of concept malware utilizing artificial intelligence. *International Journal of Electrical and Computer Engineering* (2088-8708). (Q2, SJR 2020 = 0.277)

[oth4] Truong T. C., **Diep Q. B.**, Zelinka I., and Senkerik R. (2020). Supervised Classification Methods for Fake News Identification. In: *Artificial Intelligence and Soft Computing. ICAISC 2020. Lecture Notes in Computer Science*, vol 12416. Springer, Cham.

[oth5] Nguyen L., Zelinka I. , and **Diep Q. B.** (2021). CCGraMi: An Effective Method for Mining Frequent Subgraphs in a Single Large Graph. In *MENDEL* (Vol. 27, No. 2, pp. 90-99). (Q3, SJR 2020 = 0.22)

About the Author

ORCID 0000-0003-4050-648X

publons AAG-2138-2019

Google Scholar Quoc Bao Diep

Web of Science ResearcherID: AAG-2138-2019

Metrics overview: (accessed on December 31, 2021)

6 Publications in Web of Science

18 Sum of times cited (17 without self-citations)

3 *h*-index

Scopus Author Identifier: 57209022263

Metrics overview: (accessed on December 31, 2021)

15 Documents by author in Scopus

73 Citations by 51 documents (49 exclude self-citations)

6 *h*-index

List of Projects

[pro1] Projekt RPP2019/62. Rozšíření a tvorba nových podpor pro předmět PVBPS (Počítačové viry a bezpečnost počítačových systémů). Řešitel prof. Ing. Ivan Zelinka, Ph.D. Pracoviště VŠB-Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky. 01.01.2019 - 31.12.2019. TO/spec. RPP-TO-1/a.

[pro2] Projekt RPP2020/142. Digitální forenzni analýza - moderní metody a nástroje při vyšetřování kybernetických zločinů. Řešitel prof. Ing. Ivan Zelinka, Ph.D. Pracoviště VŠB-Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky. 01.01.2020 - 31.12.2020. TO/spec. RPP-TO-1/a.

[pro3] Projekt SP2018/177. Nekonvenční algoritmy a počítačová bezpečnost. Řešitel Zelinka Ivan prof. Ing., Ph.D. - 460. Pracoviště VŠB-Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky. 01.01.2018 - 31.12.2018.

[pro4] Projekt SP2019/137. Umělá inteligence v problémech počítačové bezpečnosti a gamesourcingu. Řešitel Zelinka Ivan prof. Ing., Ph.D. - 460. Pracoviště VŠB-Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky. 01.01.2019 - 31.12.2019.

[pro5] Projekt SP2020/78. Hejnová inteligence v počítačové bezpečnosti a příbuzných problémech. Řešitel Zelinka Ivan prof. Ing., Ph.D. - 460. Pracoviště VŠB-Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky. 01.01.2020 - 31.12.2020.

[pro6] Projekt SP2021/72. Umělá inteligence a její aplikace v počítačové bezpečnosti, Esportu a příbuzných problémech. Řešitel Zelinka Ivan prof. Ing., Ph.D. - 460. Pracoviště VŠB-Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky. 01.12.2021 - 31.12.2021.

[pro7] Projekt SP2022/22. Inteligentní kybernetická bezpečnost. Řešitel Zelinka Ivan prof. Ing., Ph.D. - 460. Pracoviště VŠB-Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky. 01.12.2022 - 31.12.2022.