Article

A Simplified Fish School Search Algorithm for Continuous Single Objective Optimisation

Elliackin Figueiredo[1], Clodomir Santana[2], Hugo Valadares Siqueira[3], Mariana Macedo[4], Attilio Converti[5], Anuradha Gokhale[6], Carmelo Bastos-Filho[1]*

1 Department of Computer Engineering, University of Pernambuco, Brazil; elliackin@gmail.com
2 Department of Internal Medicine, University of California, Davis, US; clsantana@ucdavis.edu
3 Department of Electric Engineering, Federal University of Technology – Paraná, Brazil; hugosiqueira@utfpr.edu.br
4 Department of Computer Science, Northeastern University London, UK; m.macedo@northeastern.edu
5 Department of Civil, Chemical and Environmental Engineering, University of Genoa, Genoa, Italy; converti@unige.it
6 College of Applied Science and Technology, Illinois State University, US; aagokhale@ilstu.edu
* Correspondence: carmelo.filho@upe.br;

Abstract: The Fish School Search (FSS) algorithm is a metaheuristic known for its distinctive exploration and exploitation operators and cumulative success representation approach. Despite its success across various problem domains, FSS presents issues due to its high number of parameters, making its performance susceptible to improper parameterisation. Additionally, the interplay between its operators requires a sequential execution in a specific order, requiring two fitness evaluations per iteration for each individual. This operator's intricacy and the number of fitness evaluations pose the issue of costly fitness functions and inhibit parallelisation. To address these challenges, this paper proposes a Simplified Fish School Search (SFSS) algorithm that preserves the core features of the original FSS while redesigning the fish movement operators and introducing a new turbulence mechanism to enhance population diversity and robustness against stagnation. The SFSS also reduces the number of fitness evaluations per iteration and minimises the algorithm's parameter set. Computational experiments were conducted using a benchmark suite from the CEC 2017 competition to compare the SFSS with the traditional FSS and five other well-known metaheuristics. The SFSS outperformed the FSS in 84% of the problems, and achieved the best results among all algorithms in 10 of the 26 problems.

Keywords: simplified fish school search; SFSS; swarm intelligence; metaheuristics; single objective optimization

Received:
Revised:
Accepted:
Published:

Citation: Figueiredo, E.; Santana, C.; Siqueira, H. V.; Macedo, M.; Converti, A.; Gokhale, A.; Bastos-Filho, C. A Simplified Fish School Search Algorithm for Continuous Single Objective Optimisation. *Computation* **2025**, *1*, 0. https://doi.org/

1. Introduction

In computational intelligence, swarm and evolutionary metaheuristics have garnered significant attention for their ability to solve complex optimisation problems. Leveraging nature's process to evolve efficient and effective solutions to many challenges, techniques in this domain seek to apply these principles to the design metaheuristics. Among these metaheuristics are Genetic Algorithms (GA) [1], Particle Swarm Optimisation (PSO) [2], Artificial Bee Colony (ABC) [3], and Fish School Search (FSS) [28]. Besides drawing inspiration from unique biological and evolutionary principles in their design, each one of these methods possesses unique characteristics related to their behaviour, operators,and capabilities.

For example, Genetic algorithms are a class of optimisation algorithms inspired by natural selection and belong to the broader category of evolutionary algorithms (EAs) [1]. GA leverages selection, crossover, and mutation operators to evolve a population of potential solutions over successive generations [4]. GAs typically encode solutions as strings or chromosomes, often using binary representation. This encoding scheme allows for easy manipulation and combination of solutions, making it an excellent choice for combinatorial optimisation problems [5].

Unlike the GA, particle swarm optimisation is in the swarm intelligence (SI) field. PSO is based on the social behaviour of birds flocking and fish schooling [2]. It comprises a population of candidate solutions (particles) that move through the search space to find the optimal solution. Unlike other metaheuristics, PSO uses velocity and position vectors to guide the search process. The velocity vector determines the direction and speed of a particle's movement, while the position vector represents the particle's current solution [6]. The updates are governed by equations that incorporate both the particle's best-known position and the best position discovered by the swarm.

Another example of SI metaheuristic is the artificial bee colony, which simulates the foraging behaviour of honey bees [3]. In ABC, there are three types of bees: employed bees, onlooker bees, and scout bees, representing phases of the algorithm. These phases allow for a balanced exploration and exploitation of the search space. The ABC differs from the GA and most swarm-based algorithms because the candidate solutions are not encoded as part of the agents (bees). Instead, the ABC used the analogy of food sources to represent the candidate solutions. The bee searchers exploit these food sources to find better solutions to the optimisation problem [3].

A third algorithm from the SI family is the fish school search. The Fish School Search Algorithm (FSS) is inspired by fish swarms' collective and individual behaviour [7]. In the FSS, individual fish represent potential solutions, and local and global behaviours influence their movements. Individual, collective-instinct, and collective-volitive components govern the fish movement [28]. Individual movement allows each fish to explore the search space based on its own experiences. In contrast, the collective movements direct the fish school towards promising regions in the search space, influenced by the overall school's behaviour [28]. While the ABC and the PSO use current or previous best positional information to estimate success, the FSS employs a cumulative success representation for its candidate solutions [8]. The success accumulation is represented as the fish's weight. Over iterations, fish can gain weight when they improve their solution, and the populations will tend to move to regions with the heaviest fish. FSS is known for its unique strategy to balance exploration and exploitation base on the fish movements and the feeding operator [7].

Despite their success and widespread application, these mataheuristics exhibit drawbacks. For example, GA can present imitations linked to the definition of a proper solution encoding strategy [9] and premature convergence pandey2014comparative, PSO has issues maintaining swarm diversity and avoiding premature convergence [10–12], and ABC has weak in exploration [13–15].

Regarding the FSS, it is more complex to implement and has greater algorithmic complexity than the GA and PSO, and performance issues due to improper parametrisation can occur. Also, the interplay between the movements requires them to be executed sequentially with two fitness evaluations per individual per iteration: one evaluation after the individual movement used to signal the individual guiding the collective movements and another after the collective movements to update the population's fitness. Reducing the number of fitness evaluations benefits computationally expensive fitness calculations and allows for parallel execution of the movements.

This paper proposes a novel simplified version of the Fish School Search algorithm. Our approach aims to retain the core advantages of FSS—such as the balance between exploration and exploitation, adaptability, and robustness—while reducing the number of parameters and fitness evaluations per iteration. The main challenges involves identifying and preserving the essential characteristics contributing to the algorithm's success while eliminating redundancies and minimising computational overhead. The simplification process aims to:

- Analyse the original FSS to identify critical elements that drive its performance and refine or eliminate non-essential components.
- Reduce the number of fitness evaluations per iteration by redesigning the interplay between the fish movement operators.
- Decrease the number of parameters to make it less susceptible to performance issues due to improper parametrisation.

The remainder of the article is organised as follows: Section 2 gives an overview of the swarm and evolutionary metaheuristics used, Section 3 presents the new version of the FSS algorithm, Section 4 shows the computational results achieved using databases from the CEC '2017 competition and a discussion about them. Finally, Section 5 presents the conclusions.

2. Evolutionary and Swarm-based Metaheuristics

This section describes the metaheuristics studied in this paper: Genetic Algorithms, Particle Swarm Optimization, Artificial Bee Colony, and Fish School Search.

2.1. Genetic Algorithms

GAs are composed of a population of individuals, each representing a potential solution to the problem. These individuals are typically encoded as strings of bits, but this representation varies depending on the problem tackled [5]. The basic idea behind them is to evolve a population of candidate solutions to a problem over multiple generations (i.e. iterations), gradually improving their quality based on a fitness function. To evolve the population, GAs leverage three core operators: selection, crossover, and mutation [4]. These operators mimic the processes of natural selection, genetic recombination, and genetic mutation, respectively.

The first operator employed is the selection. This operator chooses individuals from the current population to act as parents for the next generation [4]. Individuals with higher fitness scores are more likely to be selected, ensuring better solutions have a greater chance of passing on their genes. Next, the crossover operator combines parts of parent solutions to produce offspring, introducing new combinations of traits [4]. Lastly, the mutation introduces random changes to individual solutions, promoting population diversity and allowing the algorithm to explore new areas of the solution space [4].

Over successive generations, the population evolves towards better solutions, with the best individuals being selected more frequently and the genetic operators introducing variation. This iterative process continues until a stopping criterion is met (e.g. a maximum number of generations). It is worth mentioning that, although these operators can be considered the core of GAs, different versions of GAs have been proposed, introducing novel and hybrid operators [16–19].

2.2. Particle Swarm Optimisation

The PSO has a population of candidate solutions (particles) that move through the search space to find the optimal solution [2]. Each particle has a position representing a potential solution to the optimization problem. The particles also have a velocity that allows them to move through the search space [20]. Three components influence the movement of each particle:

- **Inertia**: The tendency of the particle to continue moving in the same direction. The inertia helps to balance the exploration and exploitation capabilities of the swarm.
- **Cognitive component**: The tendency of the particle to move towards its personal best position (i.e., the best solution it has found so far).
- **Social component**: The tendency of the particle to move towards the global best position (i.e., the best solution found by any particle in the swarm).

The PSO also has a few parameters, such as the inertia weight, cognitive coefficient, and social coefficient, which guide the particle's movement and control the balance between exploration and exploitation [21]. For example, a larger inertia weight promotes search space exploration, while a smaller one promotes the exploitation of the best solutions [22]. To introduce stochasticity into the algorithm, the movement employs randomly generated numbers between 0 and 1 to control the influence of the personal best and global best positions on the particle's movement.

2.3. Artificial Bee Colony

In the ABC, a hive (i.e., the population) is a set of artificial bees, and food sources are the metaphor for an objective function's optimum points. A metaphor with nectar amounts represents the quality of the solutions where the best solutions possess large amounts of nectar [23].

The algorithm divides the bees into employed, onlookers, and scouts. Each bee type explores each food source separately and represents a different operator designed to improve the current set of candidate solutions (e.g., employed and onlookers bees) and prevent stagnation (e.g., onlooker bees) [24].

The iterative process starts with the employed bees, which select a random solution from the current population and adjust its location based on the information gathered from neighbouring solutions [25]. Next, the onlooker bees are also used to adjust the food sources. However, instead of picking random solutions as in the employed bee phase, they are selected based on a selection probability proportional to the fitness (i.e. nectar amount) [3]. It is worth mentioning that these two bee types represent greedy operations, which will only update the current candidate solution when a better one is found.

Lastly, when a food source is depleted (i.e. a candidate solution could not be improved after successive attempts), the scout be operator is used [26]. This procedure replaces the depleted solution with a new one generated within the search space. The ABC has a specific parameter that specifies the number of unsuccessfull improvemts attemps that should trigger the scout bee.

2.4. Fish School Search

The Fish School Search has four operators: individual movement, feed operator, collective instinctive movement, and collective volitive movement [27]. The collective movements are unique to all schools.

It considers the following variables: N is the total number of fish or the school size, $\vec{z}_i^t$ is the current position of the fish i at the iteration t. The operators are described below [28]:

- Individual movement ($\vec{n}_i^{t+1}$): random search in which each fish randomly chooses a new position in its neighbourhood. It causes diversity and triggers the other operators. It is executed according to Equation 1:

$$\vec{n}_i^{t+1} = \vec{z}_i^t + step_{ind}.rand[-1,1] \tag{1}$$

where $\vec{n}_i^{t+1}$ is a new position (temporary), $step_{ind}$ is an individual step set by the user (it may decay linearly along the iterations) and $rand[-1,1]$ is a random value generated by a uniform probability density function in the interval [-1,1]. Observe that $rand[-1,1]$ must be drawn for each dimension $d = 1, ..., D$ separately, while $step_{ind}$ is constant in the current iteration. In this movement, the fish goes to the new position only if there is more food than the current position.

- Feeding operator (w_i^t): updates the fish weight and occurs after the individual movement. Firstly, the new position $\vec{n}_i^t$ is evaluated according to the fitness $f[\vec{n}_i^t]$ obtained in the previous movement and compared to the fitness of the current position $\vec{z}_i^t$, according to Equation 2:

$$\Delta f_i^{t+1} = |f[\vec{n}_i^{t+1}] - f[\vec{z}_i^t]| \tag{2}$$

The value Δf_i^{t+1} is used to update the fish weight, as shown in Equation 3

$$w_i^{t+1} = w_i^t + \left(\frac{\Delta f_i^{t+1}}{max[\Delta f_i^{t+1}]} \right) \tag{3}$$

Equation 3 shows that the weight of the fish increases according to the success rate achieved by the individual movement. The fish will move to the new position $\vec{n}_i^{t+1}$ if the movement elevates its fitness or, in other words, if the new position is better than the current (greedy search).

- Instinctive Collective Movement ($\vec{m}^t$): This movement is influenced by the fish who successfully updated their fitness from the individual movement. All the fish perform this movement, calculated via Equation 4

$$\vec{m}^{t+1} = \frac{\sum_{i=1}^{N} \Delta \vec{z}_i^{t+1} \Delta f_i^{t+1}}{\sum_{i=1}^{N} \Delta f_i^{t+1}} \tag{4}$$

where $\vec{z}_i^t$ is the displacement if the fish i caused by the individual movement and Δf_i^t is calculated by 2.
So, all the school has its position updated by Equation 5

$$\vec{z}_i^{t+1} = \vec{z}_i^{t+1} + \vec{m}^{t+1} \tag{5}$$

- Volitive Collective Movement ($\vec{B}^t$): This second collective movement is performed according to the overall success rate of the fish school, measured by the sum of the fish weights. If the total school weight has increased ($w^{t+1} > w^t$), this means that the current search was successful. So, the school should contract to increase the exploitation behaviour. However, if the school weight has decreased ($w^{t+1} < w^t$), it should expand to increase the exploration of the search space. This movement is executed according to the school barycenter calculated via 6

$$\vec{B}^{t+1} = \frac{\sum_{i=1}^{N} \vec{z}_i^{t+1} w_i^{t+1}}{\sum_{i=1}^{N} w_i^{t+1}} \tag{6}$$

If the weight of the school grows ($w^{t+1} > w^t$), the fish' positions are updated according to Equation 7:

$$\vec{z}_i^{t+1} = \vec{z}_i^{t+1} - step_{vol}.rand[0,1]\left(\vec{z}_i^{t+1} - \vec{B}^{t+1}\right) \tag{7}$$

If not ($w^{t+1} < w^t$), perform the new positions by 8:

$$\vec{z}_i^{t+1} = \vec{z}_i^{t+1} + step_{vol}.rand[0,1]\left(\vec{z}_i^{t+1} - \vec{B}^{t+1}\right) \tag{8}$$

where the $step_{vol} = 2.step_{ind}$ (previously defined in the individual movement), and $rand[0,1]$ is a random value generated by a uniform probability density function in the interval [0,1]. Observe that $rand[0,1]$ must be drawn separately for each dimension $d = 1, ..., D$, while $step_{vol}$ is constant in the current iteration.

Algorithm 1 presents the pseudocode of the original FSS.

Over the last decade, many improvements have been made to the FSS algorithm. The Density Based Fish School Search (dFSS) was developed to solve multimodal hyper-dimensional problems, adding new operators as memory and partition [28]. The Weight-based Fish School Search (wFSS) modifies the barycenter by adding the Link Formation Rule which causes niches formation [28]. Some versions are proposed to tackle premature convergence and stagnation [28]. Most recently, the FSS family was expanded to cover multi-objective problems for continuous and binary spaces [28]. Lastly, a simplified version of the FSS was also proposed for problems in the binary domain [28], which demonstrated that it was possible to reduce the complexity of the FSS while improving its performance.

Algorithm 1: FSS Pseudocode

Initialize randomly all fish positions $\vec{z}_i^0$, according to Equation 1;
Initialize randomly all fish weights W_i^0;
while *stop criterion is not reached* **do**
    foreach *fish* **do**
        Find neighbor position according to Equation 1;
        if $\Delta(f_i^{t+1}) < 0$ *(minimization)* **then**
            Evaluate the neighbor position
            Perform greedy search and calculate the displacement using 2
        else
            Stands in the same position;
        end
    end
    Feed the fish using 3;
    foreach *fish* **do**
        Calculate the collective instinctive movement via Equation 4;
        Execute the instinctive movement using 5;
    end
    Calculate barycenter using 6;
    foreach *fish* **do**
        Execute volitive movement using either 7 or 8
    end
    Calculate the collective volitive movement via Equation 4;
    Update s_{ind} and s_{vol};
end
Return the best solution found;

3. The Proposed Fish School Search

The Simplified Fish School Search (SFSS) algorithm follows the structure and inspiration of the FSS (movements and operators). The main goal was to reduce the use of fitness functions while maintaining its generation of diversity and its automatic balance of exploitation and exploration mechanisms. Moreover, another objective was to minimize the number of parameters the user needs to initialize and define, such as initial and final step sizes, initial weight, and weight limits. The only parameter preserved is the number of individuals in the swarm.

In the original FSS, the swarm evaluates twice: one time after the individual movement and another time after the volitive movement. To reduce to only one evaluation per iteration, instead of updating the individual's position after each movement, the movements generate displacements based on the fish's current position. After all displacements are calculated, the fish position is updated combining all three displacements values. More than reducing the number of fitness evaluations, this strategy also allows the three displacements to be calculated in parallel, which reduces the execution time.

- Individual Displacement ($\vec{Ind}_i^t$): For each fish in the school, it is drawn a random value generated by a uniform distribution in the interval [0,1]. If the probability of the fish i is greater than the value generated, then the displacement is calculated using the Equation 9. Otherwise, the fish do not perform an individual displacement.

$$\vec{Ind}_{i,d}^{t+1} = rand[-1,1].(\vec{x}_{i,d}^{t-1} - \vec{x}_{j,d}^{t-1}) \tag{9}$$

  where $\vec{Ind}_{i,d}^t$ is the displacement for fish i, j is a random fish selected from the swarm (10), $rand[-1,1]$ is a random value generated by a uniform probability density function in the interval [-1,1] and d is a random dimension selected from the number of problem dimensions.
  Equation 10 calculates the fish selection probability.

$$P_i^{t+1} = \frac{w_i^{t+1}}{max[W^{t+1}]} \tag{10}$$

  where w_i is the weight of the fish i and $max[W^{t+1}]$ returns weight of the heaviest fish in the school.

- Instinctive Displacement ($\vec{Ins}_i^t$): For each fish that had improved in the school, the new position is calculated using the Equation 11:

$$\vec{Ins}_{i,d}^{t+1} = \frac{select(\vec{x}_{i,d}^{t-1} - \vec{x}_{i,d}^{t})}{\sum_{i=1}^{N} w_i^t} \tag{11}$$

  where $\vec{Ins}_i^t$ represents the instinctive displacement of fish i in dimension d, $select([-1,1])$ is a function which selects and returns 1 or -1 and w_i^t is the weight of fish i at time t.

- Volitive Collective Displacement ($\vec{Vol}_i^t$): For each fish in the school, the displacements are generated by Equation 12:

$$\vec{Vol}_i^{t+1} = sign\left(\vec{x}_i^t - \vec{x}_j^t\right) \tag{12}$$

  where $\vec{Vol}_i^{t+1}$ is the volitive displacement for fish i, j is a fish selected from the swarm using a binary tournament process, $sign$ is a function which returns a random value generated by a uniform probability density function in the interval [-1,0], if the weight of the fish j is greater than the weight of fish i, or [1, 0] otherwise. This means that the fish i will move towards the fish j if the fish j is heavier.

The new position of the fish is generated by combining the three displacements, as can be seen in Equation 13

$$\vec{x}_i^{t+1} = \vec{Ind}_i^{t+1} + \vec{Ins}_i^{t+1} + \vec{Vol}_i^{t+1} \tag{13}$$

Another modification made is related to the feeding operator (W_i^t). The fish's weight reflects how good the solution is found, and it determines the degree of influence a fish has on the swarm. Initially, when a fish moves to a better region, it gains weight, and if it is not able to improve, its weight remains the same. Even though it will be punished by not having the instinctive movement and its influence will decrease as other fish get heavier, this process might be slow. In SFSS, weight loss was introduced to penalize even more fish that do not improve. First, the (Δf_i^{t+1}) is calculated using Equation 2, if $\Delta f_i^{t+1} > 0$, fish weight is updated with the FSS weight gain (Equation 3), otherwise use (14).

$$w_i^{t+1} = w_i^t.e^{-\left(\left|\frac{\Delta f_{t+1}^t}{max[\Delta f_i^{t+1}]}\right|\right)} \tag{14}$$

where w_i is the weight of fish i, e is the exponential function and $max[\Delta f_i^{t+1}]$ return the maximum variation of fitness in the school.

In preliminary experiments, we observed the algorithm's performance in different problems by analysing the population weight over the iterations. In these experiments, we noticed that in some cases, the population weight could reach values below one after several iterations without improvements and losing diversity, causing stagnation issues. To address this issue, the SFSS features a turbulence mechanism to promote population diversity and increase the probability of improvements in the population. This mechanism is triggered only in stagnation situations (e.g., swarm weight below one) and for a limited number of fish in the population. In preliminary experiments, we noticed that applying the perturbation to 10% of the worst individuals in the school was enough to produce satisfactory results. This perturbation is not applied on consecutive iterations to prevent the adverse effects of introducing too much diversity. Also, we chose the Gaussian perturbation as our turbulence operator as it is simple to implement, has a low computational cost and produces the expected results.

The SFSS is described in the Algorithm 2. It is important to mention that in line 5 the turbulence will only be applied on the worst ten percent fish of the school.

3.1. *SFSS: Trials*

During the development of the proposal, the following ideas were also considered as candidates to replace the movements and operators of the FSS. However, they were not used in the final version because they do not improve the algorithm performance, or another simpler solution reveals similar or better results than these.

3.1.1. Movements Trials

- Uses a roulette wheel to select a fish that will try to move and another roulette to select a fish that will attract fish. The fish moves if the new location is better than the previous one.
- Similar to the previous one, but instead of a roulette wheel to select a fish that will try to move, all fish try to move.
- All fish try to move, and selecting a random fish from the school will attract fish.

Algorithm 2: SFSS Pseudocode

```
Initialize randomly all fish positions z⃗_i^0, according to Equation 1;
Initialize all fish weights (W_i^0) as 0 and initial probability as 1/(school size);
while stop criterion is not reached do
    if (school weight < 1) and (turbulence was not used in last iteration) then
        Apply Gaussian turbulence;
    else
        foreach fish do
            Calculate the displacements using equations 9, 11 and 12;
            Generate the new position using Equation 13 and evaluate it;
            Move to new position only if cost of new position is greater than the
              current cost
        end
        foreach fish do
            Feed the fish with Equation 3 if fish improved, otherwise use Equation
              14;
            Update swarm weight and the probability applying Equation 10;
        end
    end
end
```

- Generating a new position in all dimensions VS modifying only one random dimension.

3.1.2. Feeding Operator Trials

The variations described in this section aimed to find a more appropriate form to penalize or reward the fish when necessary. The weight loss means that a fish could not improve in the current iteration, and when the school weight decreases, it might indicate that the swarm converged or is trapped in a local minimal.

- Exponential weight gain and loss:

$$w_i^t = w_i^{t-1} e^{\frac{\Delta f_i^{t+1}}{max[\Delta f_i^{t+1}]}} \tag{15}$$

- Nonlinear weight gain attempt 1:

$$w_i^t = w_i^{t-1} + \left(\frac{|\Delta f_i^{t+1} - \Delta f_i^t|}{max[\Delta f_i^{t+1} - \Delta f_i^t]}\right)\left(\frac{-1}{\left|\frac{\Delta f_i^{t+1} - \Delta f_i^t}{max[\Delta f_i^{t+1} - \Delta f_i^t]}\right| - 1} - 1\right) \tag{16}$$

- Nonlinear weight gain attempt 2:
  It is similar to the previous one but with an addition operation instead of multiplication between the normalized variation of the delta cost with the last term.
- Nonlinear weight gain attempt 3:

$$w_i^t = w_i^{t-1} + \left(\frac{|\Delta f_i^{t+1} - \Delta f_i^t|}{max[\Delta f_i^{t+1} - \Delta f_i^t]}\right)\left(\frac{|\Delta f_i^{t+1} - \Delta f_i^t|}{max[\Delta f_i^{t+1} - \Delta f_i^t]}\right)^5 \tag{17}$$

- Nonlinear weight gain attempt 4:

$$w_i^t = w_i^{t-1}\left(\frac{|\Delta f_i^{t+1} - \Delta f_i^t|}{max[\Delta f_i^{t+1} - \Delta f_i^t]}\right)^5 \tag{18}$$

- Nonlinear weight gain attempt 5:

$$w_i^t = w_i^{t-1} + \left(\frac{|\Delta f_i^{t+1} - \Delta f_i^t|}{max[\Delta f_i^{t+1} - \Delta f_i^t]}\right).100^{\left(\frac{|\Delta f_i^{t+1} - \Delta f_i^t|}{max[\Delta f_i^{t+1} - \Delta f_i^t]}\right)-1} \tag{19}$$

All the exponential weight gain attempts produced similar results. For this reason, the criteria for selecting one of the approaches were simplicity and computational cost.

4. Case Study

We tested the algorithms using 26 optimization problems from the IEEE Congress on Evolutionary Computation (CEC) 2017 test suit [29]. Although the test suit has 28 problems, we excluded the F17 and F21 because they show unstable behaviour possibly caused by the source code. The Python code for the CEC'17 test suite can be downloaded from the GitHub page [1]. All the functions are tested in 30 dimensions, and among this problem, we have unimodal, multimodal, shifted, rotated, and composed functions. The experiments were conducted in a 12th Gen Intel(R) Core(TM) i7-12700H 2.30 GHz processor, 40GB of RAM, 1TB of hard drive, and running Windows 11 Pro version 23H2 64-bit operating system.

Since the ABC and the FSS algorithms have more than one fitness evaluation per individual per iteration, to provide a fair comparison, we decided to use the number of fitness evaluations as the stop criteria of the execution. Furthermore, considering that the CEC functions can be challenging, it was decided, after previous experiments, that the number of fitness evaluations adopted would be five hundred thousand. All the algorithms were executed 30 times for each function and had a population of 30 individuals.

The PSO was implemented with a global best topology and used $w0 = 0.72984$, C_1, and C_2 equal to $(2.05w)$ and a maximum velocity of 100000. ABC algorithm employed the trial limit of 100. The GA algorithm was configured with a mutation rate of 0.05 and a crossover constant equal to 0.9. The FSS has initial and final individual steps, respectively, equal to 0.1 and 0.0001, the initial volitive step of 0.01, and a final volitive step of 0.001. Moreover, the initial weight and weight scale of FSS were one and (number of fitness evaluation)/4.0, respectively. The SFSS and FA do not have additional parameters to set aside from the population size.

Figure 1 shows an example of the convergence curve of all seven algorithms in six CEC problems, while Table 1 and Figure 2 compare their performance in all 26 problems. As seen in Table 1 and Figure 2, the SFSS overcame the FSS algorithm in 23 of the 26 CEC problems. The SFSS performed best in 10 of the 26 problems compared to the other algorithms. These results suggest that the SFSS reduced the number of fitness evaluations per iteration and presented performance gains.

[1] more information available at https://github.com/tilleyd/cec2017-py/tree/master

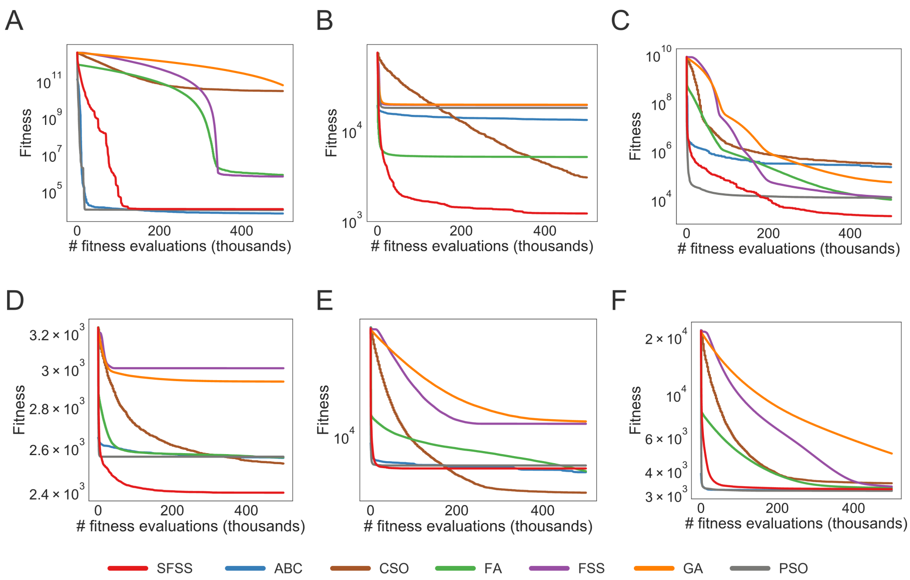

Figure 1. Example of convergence curves of the algorithms in (A) F1, (B) F9, (C) F14, (D) F21, (E) F26, and (F) F28. We present the results for 26 CEC problems with 30 dimensions. The algorithms were interrupted after 500 thousand fitness evaluations.

Figure 2 illustrates the results of the Wilcoxon test comparing the SFSS to the other algorithms. In this figures, the blue square means that the SFSS was superior, the red square denotes that the SFSS was inferior, and the grey square means that there is no statistical difference between them. The statistical results in Figure 2 reinforce the superiority of the SFSS over the FSS. Furthermore, comparing the SFSS to each algorithm reveals statistical significance in most of the results when the SFSS was better than the other algorithms.

Table 1. Performance evaluation in terms of average fitness value and (standard deviation) for all algorithms. The results for 26 CEC problems with 30 dimensions. The algorithms were interrupted after 500 thousand fitness evaluations. In bold, we have the best results.

Function	SFSS	ABC	CSO	FA	FSS	GA	PSO
F1	1.09E+04	**6.30E+03**	2.87E+10	7.87E+05	6.46E+05	5.95E+10	9.71E+03
	(6.17E+03)	(**2.84E+03**)	(4.46E+09)	(7.86E+04)	(8.34E+04)	(1.64E+10)	(5.43E+03)
F2	1.99E+32	**1.03E+08**	5.46E+30	2.74E+15	1.47E+14	2.06E+29	3.95E+11
	(4.65E+32)	(**1.52E+08**)	(1.39E+31)	(3.02E+15)	(1.13E+14)	(2.90E+29)	(2.12E+12)
F3	9.24E+04	1.89E+05	9.48E+04	**9.99E+03**	1.09E+04	1.38E+05	5.02E+04
	(7.02E+04)	(1.48E+04)	(1.65E+04)	(**5.81E+02**)	(2.04E+03)	(1.33E+04)	(2.90E+04)
F4	4.98E+02	4.68E+02	7.23E+02	5.11E+02	5.07E+02	6.01E+02	**4.21E+02**
	(2.82E+01)	(1.80E+01)	(4.08E+01)	(7.17E+00)	(1.92E+01)	(3.66E+01)	(**2.76E+01**)
F5	**6.25E+02**	7.11E+02	7.39E+02	7.60E+02	1.59E+03	1.48E+03	7.79E+02
	(**4.23E+01**)	(1.89E+01)	(1.58E+01)	(1.45E+01)	(1.33E+02)	(5.23E+01)	(7.30E+01)
F6	**6.22E+02**	6.93E+02	6.40E+02	6.72E+02	7.20E+02	7.22E+02	7.10E+02
	(**7.43E+00**)	(2.50E+00)	(6.09E+00)	(3.86E+00)	(9.24E+00)	(5.51E+00)	(1.28E+01)
F7	1.04E+03	**8.60E+02**	1.02E+03	1.30E+03	6.63E+03	6.32E+03	1.25E+03
	(1.46E+02)	(**1.73E+01**)	(1.73E+01)	(1.49E+01)	(9.01E+02)	(2.17E+02)	(1.66E+02)
F8	**9.47E+02**	1.18E+03	1.04E+03	9.73E+02	1.63E+03	1.58E+03	1.09E+03
	(**4.98E+01**)	(3.15E+01)	(1.96E+01)	(1.18E+01)	(9.38E+01)	(3.52E+01)	(1.34E+02)
F9	**1.21E+03**	1.32E+04	3.04E+03	5.13E+03	1.94E+04	1.95E+04	1.80E+04
	(**4.42E+02**)	(5.76E+02)	(8.07E+02)	(1.47E+02)	(2.28E+03)	(1.11E+03)	(2.08E+03)
F10	5.75E+03	**4.13E+03**	8.67E+03	4.68E+03	6.73E+03	5.96E+03	5.69E+03
	(8.61E+02)	(**2.17E+02**)	(2.47E+02)	(2.29E+02)	(4.12E+02)	(5.63E+02)	(5.68E+02)
F11	1.29E+03	2.30E+03	2.37E+03	**1.17E+03**	1.47E+03	1.84E+03	1.36E+03
	(2.89E+02)	(6.97E+02)	(3.42E+02)	(**4.17E+00**)	(4.34E+01)	(1.51E+02)	(6.25E+01)
F12	2.80E+05	2.30E+06	2.99E+09	1.17E+07	5.28E+06	7.88E+06	**7.27E+04**
	(2.33E+05)	(5.79E+05)	(6.01E+08)	(2.52E+06)	(5.62E+05)	(2.32E+06)	(**9.55E+04**)

F13	**4.77E+03** (**3.43E+03**)	1.16E+04 (4.17E+03)	1.41E+09 (4.54E+08)	5.92E+04 (1.16E+04)	3.14E+05 (4.78E+04)	1.49E+04 (2.18E+03)	7.17E+03 (5.14E+03)
F14	**2.07E+03** (**2.36E+02**)	2.23E+05 (8.56E+04)	2.89E+05 (1.07E+05)	1.01E+04 (1.44E+03)	9.67E+03 (3.84E+03)	5.22E+04 (1.09E+04)	1.19E+04 (1.20E+04)
F15	1.08E+05 (1.30E+05)	6.12E+03 (3.93E+03)	8.55E+07 (4.76E+07)	1.60E+04 (1.53E+03)	1.20E+05 (1.78E+04)	**2.31E+03** (**4.47E+02**)	9.72E+03 (9.40E+03)
F16	2.32E+03 (3.64E+02)	**2.24E+03** (**1.06E+02**)	3.40E+03 (1.79E+02)	3.36E+03 (2.68E+02)	3.24E+03 (3.59E+02)	3.37E+03 (3.88E+02)	2.73E+03 (3.66E+02)
F18	**8.81E+04** (**3.76E+04**)	3.17E+05 (9.84E+04)	2.52E+06 (1.25E+06)	9.46E+04 (8.78E+03)	1.56E+05 (1.75E+04)	1.06E+06 (2.04E+05)	1.43E+05 (8.95E+04)
F19	7.10E+05 (2.23E+06)	2.56E+04 (1.43E+04)	1.64E+08 (5.89E+07)	9.96E+05 (2.62E+05)	1.30E+06 (2.41E+05)	**4.66E+03** (**9.24E+02**)	9.02E+03 (6.24E+03)
F21	**2.40E+03** (**3.40E+01**)	2.55E+03 (1.67E+01)	2.53E+03 (1.42E+01)	2.56E+03 (7.90E+01)	3.00E+03 (6.62E+01)	2.93E+03 (3.19E+01)	2.56E+03 (6.84E+01)
F22	7.04E+03 (1.39E+03)	**5.76E+03** (**2.45E+02**)	7.95E+03 (2.86E+03)	7.41E+03 (1.66E+02)	8.19E+03 (4.37E+02)	7.69E+03 (7.58E+02)	7.00E+03 (5.86E+02)
F23	**2.74E+03** (**4.10E+01**)	2.82E+03 (1.81E+01)	2.93E+03 (3.20E+01)	3.88E+03 (1.06E+02)	5.00E+03 (2.44E+02)	4.69E+03 (1.57E+02)	3.24E+03 (4.52E+02)
F24	**2.89E+03** (**3.13E+01**)	3.14E+03 (3.17E+01)	3.08E+03 (2.41E+01)	3.46E+03 (1.61E+02)	4.05E+03 (1.29E+02)	4.03E+03 (4.69E+01)	3.19E+03 (1.53E+02)
F25	2.90E+03 (1.30E+01)	2.88E+03 (5.62E-02)	3.03E+03 (1.80E+01)	2.90E+03 (2.02E+00)	**2.88E+03** (**1.72E+00**)	2.91E+03 (1.47E+01)	2.89E+03 (1.25E+01)
F26	5.76E+03 (1.03E+03)	5.40E+03 (1.32E+03)	**3.79E+03** (**9.98E+01**)	5.49E+03 (1.06E+03)	1.24E+04 (1.66E+03)	1.29E+04 (3.37E+03)	6.06E+03 (2.20E+03)
F27	3.24E+03 (1.90E+01)	**3.22E+03** (**3.66E+00**)	3.37E+03 (3.37E+01)	4.94E+03 (2.63E+02)	4.06E+03 (1.56E+02)	3.97E+03 (2.16E+02)	3.40E+03 (1.06E+02)
F28	3.21E+03 (2.44E+01)	3.16E+03 (4.08E+01)	3.45E+03 (3.76E+01)	3.26E+03 (2.91E+00)	3.32E+03 (4.03E+01)	4.93E+03 (3.77E+02)	**3.15E+03** (**6.30E+01**)

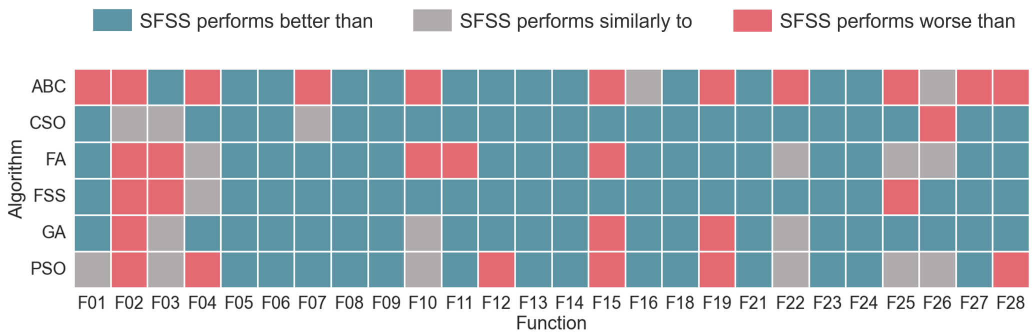

Figure 2. Results of Wilcoxon test comparing the proposed metaheuristic to the other algorithms. We present the results for 26 CEC problems with 30 dimensions. The algorithms were interrupted after 500 thousand fitness evaluations. Blue signifies superiority of the SFSS, while the red indicates inferiority. Grey denotes no statistical difference between them.

Regarding the algorithm complexity, we employed a calculation similar to the one adopted at CEC'17 [29], which can be described as follows:

1. Calculate the function complexity T_i by computing time of 10000 evaluations for problem i.
2. Compute the algorithm complexity TA_i by computing the time of 10000 evaluations for problem i. To accommodate variations in performance due to the algorithms' stochastic nature, the TA_i is the average of 15 runs.
3. The final complexity is given by $AC = (TA_i - T_i)/T_i$

The main difference between this definition and the CEC'17 is that here, we present the complexity per function, while the CEC'17 calculates the overall complexity in all problems in the test suit.

The results of the algorithm's complexity across the benchmark problems are presented in Table 2. As shown in Table 2, the PSO, the CSO, and GA are the algorithms with the lowest complexity, while the FA, FSS and SFSS presented the highest values. As this definition assesses the time required to execute a given number of fitness evaluations, we expect the SFSS to exhibit higher values than the FSS. This results arises from the SFSS combining its operators' complexity within a unique fitness evaluation per iteration. In contrast, the FSS has two evaluations per iteration.

Table 2. Analysis of the algorithm's complexity on the 26 benchmark problems. Note that, because this definition gauges performance based on the time it takes to perform a certain number of fitness evaluations, the SFSS is expected to present greater complexity than the FSS.

Function	SFSS	ABC	CSO	FA	FSS	GA	PSO
F1	8.166	1.937	0.348	5.067	4.615	1.396	0.957
F2	5.416	1.828	0.315	4.874	4.367	1.336	0.937
F3	3.517	0.933	0.112	2.783	2.448	0.577	0.331
F4	5.283	1.091	0.037	3.239	2.913	0.704	0.463
F5	6.422	1.820	0.308	5.059	4.398	1.311	0.915
F6	4.109	1.090	0.155	3.253	2.782	0.741	0.456
F7	3.279	0.902	0.154	2.392	1.992	0.600	0.411
F8	4.869	1.544	0.441	3.937	3.637	1.070	0.774
F9	3.128	0.864	0.144	2.745	2.389	0.588	0.385
F10	2.649	0.780	0.154	2.399	1.962	0.508	0.358
F11	1.533	0.567	0.181	1.531	1.266	0.391	0.292
F12	1.880	0.561	0.197	1.315	1.204	0.370	0.303
F13	1.676	0.544	0.191	1.321	1.110	0.358	0.278
F14	1.305	0.489	0.184	1.252	1.032	0.333	0.244
F15	1.584	0.524	0.172	1.327	1.117	0.350	0.267
F16	1.228	0.413	0.153	1.046	0.897	0.262	0.198
F18	1.253	0.452	0.186	1.093	0.933	0.298	0.226
F19	0.968	0.343	0.162	0.845	0.718	0.227	0.174
F21	0.942	0.306	0.131	0.799	0.734	0.163	0.114
F22	0.850	0.341	0.196	0.741	0.748	0.204	0.161
F23	0.686	0.217	0.135	0.557	0.484	0.111	0.124
F24	0.782	0.287	0.162	0.650	0.576	0.175	0.139
F25	0.755	0.204	0.171	0.606	0.443	0.111	0.077
F26	0.672	0.210	0.159	0.595	0.448	0.119	0.090
F27	0.543	0.123	0.120	0.404	0.331	0.053	0.048
F28	0.661	0.189	0.179	0.484	0.396	0.120	0.098

5. Conclusion

This paper presented the Simplified Fish School Search as an alternative algorithm for optimization problems. The experiment results show that it could overcome the FSS in most of the problems analysed (22 of 26) and compete with well-known algorithms such as PSO, ABC, GA, CSO and FA.

The proposed algorithm's performance in unimodal, multimodal, and composition problems was satisfactory, showing the SFSS's versatility. Furthermore, the computational cost and the number of fitness evaluations per individual per iteration were reduced. Reducing the number of calls to the fitness function is essential when dealing with functions with elevated costs.

Finally, we reduced the number of parameters, which led to a less user-dependent and problem-dependent algorithm with no parameter specification required besides the population size.

We aim to apply our proposal to other challenging real-world problems in future works. Moreover, it is recommended that the contributions of the three displacements to the swarm performance be assessed and the effectivity of improvements in the turbulence mechanism further studied.

Author Contributions: Conceptualization, E.F, C.S., H.F, C.B and M.M.; methodology, E.F and C.S.; software, C.S and M.M.; validation, E.F., C.S, and M.M.; investigation, E.F. and C.S.; resources, C.B and A.G..; writing—original draft preparation, E.F, C.S. and M.M. .; writing—review and editing, all authors.; visualization, C.S.; supervision, C.B. and A.G.; project administration, C.B. and A.G.; funding acquisition,C.B. and A.G.. All authors have read and agreed to the published version of the manuscript.

Data Availability Statement: No new data were created, and code can be sent upon request.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ABC	Artificial Bee Colony
EAs	Evolutionary Algorithms
FSS	Fish School Search
GA	Genetic Algorithms
PSO	Particle Swarm Optimization
SFSS	Simplified Fish School Search
SI	Swarm Intelligence
wFSS	Weight-based Fish School Search

References

1. J. H. Holland, Genetic algorithms, Scientific american 267 (1) (1992) 66–73.
2. R. Eberhart, J. Kennedy, Particle swarm optimization, in: Proceedings of the IEEE international conference on neural networks, Vol. 4, 1995, pp. 1942–1948.
3. D. Karaboga, B. Basturkl, A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (abc) algorithm, Journal of Global Optimization 39 (3) (2007) 459–471.
4. L. M. Schmitt, Theory of genetic algorithms, Theoretical Computer Science 259 (1-2) (2001) 1–61.
5. A. Kumar, Encoding schemes in genetic algorithm, International Journal of Advanced Research in IT and Engineering 2 (3) (2013) 1–7.
6. I. Sousa-Ferreira, D. Sousa, A review of velocity-type pso variants, Journal of Algorithms & Computational Technology 11 (1) (2017) 23–30.
7. C. J. Bastos Filho, F. B. de Lima Neto, A. J. Lins, A. I. Nascimento, M. P. Lima, A novel search algorithm based on fish school behavior, in: Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on, IEEE, 2008, pp. 2646–2651.
8. I. M. de Albuquerque, J. Monteiro Filho, F. B. de Lima Neto, A. M. de Oliveira Silva, Solving assembly line balancing problems with fish school search algorithm, in: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2016, pp. 1–8.
9. S. Ronald, Robust encodings in genetic algorithms: A survey of encoding issues, in: Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC'97), IEEE, 1997, pp. 43–48.
10. A. R. Jordehi, Enhanced leader pso (elpso): a new pso variant for solving global optimisation problems, Applied Soft Computing 26 (2015) 401–417.
11. R. F. Abdel-Kader, An improved pso algorithm with genetic and neighborhood-based diversity operators for the job shop scheduling problem, Applied Artificial Intelligence 32 (5) (2018) 433–462.
12. Q. Tang, J. Zeng, H. Li, C. Li, Y. Liu, A particle swarm optimization algorithm based on genetic selection strategy, in: Advances in Neural Networks–ISNN 2009: 6th International Symposium on Neural Networks, ISNN 2009 Wuhan, China, May 26-29, 2009 Proceedings, Part III 6, Springer, 2009, pp. 126–135.
13. T. Ye, H. Wang, W. Wang, T. Zeng, L. Zhang, Z. Huang, Artificial bee colony algorithm with an adaptive search manner and dimension perturbation, Neural Computing and Applications 34 (19) (2022) 16239–16253.
14. H. Makas, N. YUMUŞAK, Balancing exploration and exploitation by using sequential execution cooperation between artificial bee colony and migrating birds optimization algorithms, Turkish Journal of Electrical Engineering and Computer Sciences 24 (6) (2016) 4935–4956.

15. W.-J. Yu, Z.-H. Zhan, J. Zhang, Artificial bee colony algorithm with an adaptive greedy position update strategy, Soft Computing 22 (2018) 437–451.
16. K. Yamamoto, O. Inoue, New evolutionary direction operator for genetic algorithms, AIAA journal 33 (10) (1995) 1990–1993.
17. J. E. Smith, T. C. Fogarty, Operator and parameter adaptation in genetic algorithms, Soft computing 1 (1997) 81–87.
18. K. Deep, M. Thakur, A new mutation operator for real coded genetic algorithms, Applied mathematics and Computation 193 (1) (2007) 211–230.
19. T. A. El-Mihoub, A. A. Hopgood, L. Nolle, A. Battersby, Hybrid genetic algorithms: A review., Eng. Lett. 13 (2) (2006) 124–137.
20. Z. H. Zhan, J. Zhang, Y. Li, H. S. H. Chung, Adaptive particle swarm optimization, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 39 (6) (2009) 1362–1381. doi:10.1109/TSMCB.2009.2015956.
21. A. Nickabadi, M. M. Ebadzadeh, R. Safabakhsh, A novel particle swarm optimization algorithm with adaptive inertia weight, Applied soft computing 11 (4) (2011) 3658–3670.
22. D. Wang, D. Tan, L. Liu, Particle swarm optimization algorithm: an overview, Soft computing 22 (2) (2018) 387–408.
23. L. Cui, K. Zhang, G. Li, X. Wang, S. Yang, Z. Ming, J. Z. Huang, N. Lu, A smart artificial bee colony algorithm with distance-fitness-based neighbor search and its application, Future Generation Computer Systems 89 (2018) 478–493.
24. J. Huo, Y. Zhang, H. Zhao, An improved artificial bee colony algorithm for numerical functions, International Journal of Reasoning-based Intelligent Systems 7 (3-4) (2015) 200–208.
25. B. Akay, D. Karaboga, A modified artificial bee colony algorithm for real-parameter optimization, Information Sciences 192 (2012) 120–142.
26. D. Karaboga, C. Ozturk, A novel clustering approach: Artificial bee colony (abc) algorithm, Applied Soft Computing 11 (1) (2011) 652–657.
27. C. J. Santana, C. J. Bastos-Filho, M. Macedo, H. Siqueira, Sbfss: Simplified binary fish school search, in: 2019 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2019, pp. 2595–2602.
28. Bastos-Filho C.J.A., de Lima-Neto F.B., Lins A.J.D.C.C., de Lacerda M.G.P., da Motta Macedo M.G., de Santana Junior C.J., Siqueira H.V., da Silva R.C.L., Neto H.A., de Melo Menezes B.A., Albuquerque I.M.C., Fish School Search: Account for the First Decade, Handbook of AI-based Metaheuristics, 2021, CRC Press, pp. 21–42.
29. G. Wu, R. Mallipeddi, P. Suganthan, Problem definitions and evaluation criteria for the cec 2017 competition and special session on constrained single objective real-parameter optimization, Nanyang Technol. Univ., Singapore, Tech. Rep (2016) 1–18.