algorithms

MDPI

Article

Improving Monarch Butterfly Optimization Algorithm with Self-Adaptive Population

Hui Hu [1], Zhaoquan Cai [1,*], Song Hu [2], Yingxue Cai [2], Jia Chen [2] and Sibo Huang [2]

[1] Department of Information Science and Technology, Huizhou University, Huizhou 516007, China; huhui@hzu.edu.cn

[2] Educational Technology Center, Huizhou University, Huizhou 516007, China; 13516698334@163.com (S.H.); cyx@hzu.edu.cn (Y.C.); cj@hzu.edu.cn (J.C.); huangsibo@189.cn (S.H.)

* Correspondence: 13502279833@126.com

Received: 21 March 2018; Accepted: 27 April 2018; Published: 14 May 2018

check for updates

Abstract: Inspired by the migration behavior of monarch butterflies in nature, Wang et al. proposed a novel, promising, intelligent swarm-based algorithm, monarch butterfly optimization (MBO), for tackling global optimization problems. In the basic MBO algorithm, the butterflies in land 1 (subpopulation 1) and land 2 (subpopulation 2) are calculated according to the parameter p, which is unchanged during the entire optimization process. In our present work, a self-adaptive strategy is introduced to dynamically adjust the butterflies in land 1 and 2. Accordingly, the population size in subpopulation 1 and 2 are dynamically changed as the algorithm evolves in a linear way. After introducing the concept of a self-adaptive strategy, an improved MBO algorithm, called monarch butterfly optimization with self-adaptive population (SPMBO), is put forward. In SPMBO, only generated individuals who are better than before can be accepted as new individuals for the next generations in the migration operation. Finally, the proposed SPMBO algorithm is benchmarked by thirteen standard test functions with dimensions of 30 and 60. The experimental results indicate that the search ability of the proposed SPMBO approach significantly outperforms the basic MBO algorithm on most test functions. This also implies the self-adaptive strategy is an effective way to improve the performance of the basic MBO algorithm.

Keywords: monarch butterfly optimization; migration operator; butterfly adjusting operator; greedy strategy; benchmark problems

1. Introduction

To optimize is to maximize or minimize given functions in a certain domain. In real life, human beings are driven to maximize profit or minimize cost. In mathematics and computer science, these real-world problems can be mathematically modeled, and then further tackled by various optimization techniques. In general, these optimization techniques can loosely be divided into two categories: traditional optimization methods and modern intelligent optimization algorithms. For each run, the traditional optimization methods will generate the same results under the same initial conditions; while modern intelligent optimization algorithms will generate fully different results even if the same conditions are provided. Since current problems are becoming more and more complicated, traditional optimization methods do not effeciently solve them. Therefore, more and more researchers have turned to modern intelligent optimization algorithms [1], which mainly include evolutionary computation [2], swarm intelligence, extreme learning machines [3], or artificial neural networks [4]. Among the different kinds of intelligent algorithms, swarm intelligence (SI) algorithms [5–8] are one of the most representative paradigms.

In 1995, particle swarm optimization (PSO) [9–15] was proposed based on the inspiration of bird flocking. In a sense, the development of PSO is one of the milestones in the history of swarm intelligence algorithms. Since then, many researchers have performed numerous in-depth studies on PSO, and it has been successfully used to solve various complicated engineering problems [16]. Such problems include gesture segmentation [17], scheduling [18], shape design [19], vehicle routing [20], test-sheet composition [21], malicious code detection [22], economic load dispatch [23,24], IIR system identification [25], prediction of pupylation sites [26], target assessment [27,28], unit commitment [29], path planning [30–32], directing orbits of chaotic systems [33], image processing [34], task assignment problem [35], floorplanning [36,37], clustering [38], wind generator optimization [39], reliability problems [40], knapsack problem [41–43], and fault diagnosis [44]. Recently, immediately following PSO, many excellent SI algorithms have been put forward, including the ant colony optimization (ACO) [45–47], harmony search (HS) [48,49], artificial bee colony (ABC) [50–54], cuckoo search (CS) [55–61], fireworks algorithm (FWA) [62], bat algorithm (BA) [63–67], fruit fly optimization algorithm (FOA) [68], earthworm optimization algorithm (EWA) [69], elephant herding optimization (EHO) [70–72], moth search (MS) algorithm [73,74], biogeography-based optimization (BBO) [75–77], firefly algorithm (FA) [78–80], krill herd (KH) [81–87], and monarch butterfly optimization (MBO) [88]. These various algorithms are inspired by the swarm behavior of ants, honey bees, cuckoos, bats, grey wolves, krill, and butterflies.

Recently, after a careful study of the migration behavior of monarch butterflies, Wang et al. [88] designed a novel promising swarm intelligence-based optimization technique, called the monarch butterfly optimization (MBO). In MBO, all the monarch butterflies are located at land 1 and land 2, which are updated through implementation of the migration operator and butterfly adjusting operator at each generation. Based on thorough in-depth comparative studies of thirty-eight benchmark problems selected from previous literature, it was found that MBO significantly outperforms five other state-of-the-art metaheuristic algorithms.

However, in the basic MBO algorithm, after implementing the migration operator, the generated monarch butterfly will be accepted as a new butterfly individual in the next generation regardless of whether it is better or worse. Also, the number of monarch butterflies in land 1 and land 2 are fixed and unchanged during the entire optimization process, which are calculated at the begin of the search according to the parameter p. In this paper, two main modifications are proposed to improve the performance of the basic MBO algorithm, which are self-adaptive and greedy strategies. A self-adaptive strategy is introduced to adjust the butterfly number in land 1 and land 2 in a linear fashion during the optimization process. Additionally, only an improved butterfly individual generated by the migration operator is accepted and considered as a new butterfly individual in the next generation. This greedy strategy can make the butterfly population move toward a better status at all times. In this way, this also guarantees that the generated population is at least not worse than before. After inserting the two modifications into the basic MBO algorithm, a new variant of MBO, called the self-adaptive population MBO (SPMBO), is proposed. Furthermore, the performance of SPMBO is fully investigated by experiments on thirteen standard test functions with dimensions of 30 and 60. The experimental results indicate that the proposed SPMBO approach has much better search ability than the basic MBO algorithm in most cases. This also implies the self-adaptive strategy is an effective way to improve the performance of the basic MBO algorithm when addressing high-dimensional global optimization problems.

The rest of this paper is structured as follows. Section 2 provides an overview of related work on the MBO algorithm, followed by a description of the basic MBO method in Section 3. Section 4 discusses how the self-adaptive and greedy strategies were incorporated to improve the performance of the basic MBO. Subsequently, SPMBO was fully investigated on 30-D and 60-D benchmarks and the corresponding outcomes are described in Section 5. The paper ends with Section 6, after presenting some concluding remarks as well as scope for further work.

2. Related Work

Since the monarch butterfly optimization algorithm [88] was proposed, many scholars have worked on MBO algorithm. In this section, some of the most representative work regarding MBO and other metaheuristic algorithms are summarized and reviewed.

Kaedi [89] proposed a population-based metaheuristic algorithm, namely the fractal-based algorithm, for tackling continuous optimization problems. In this algorithm, the density of high quality and promising points in an area is considered as a heuristic which estimates the degree of promise for finding the optimal solution in that area. The promising areas of state space are then iteratively detected and partitioned into self-similar and fractal-shaped subspaces, each being searched more precisely and more extensively. Comparison with some other, metaheuristic algorithms, demonstrated that this algorithm could find high quality solutions within an appropriate time.

Shams et al. [90] proposed a novel optimization algorithm, the ideal gas optimization (IGO) with the inspiration of the first law of thermodynamics and kinetic theory. In IGO, the searcher agents are a collection of molecules with pressure and temperature. IGO uses the interaction between gas systems and molecules to search the problem space for the solution. The IGO algorithm was benchmarked by an array of benchmarks. Comparison of the results with PSO and the genetic algorithm (GA) showed an advantage of the IGO approach.

Precup et al. [91] suggested a synergy of fuzzy logic and nature-inspired algorithms in the context of the nature-inspired optimal tuning of the input membership functions of a class of Takagi–Sugeno–Kang (TSK) fuzzy models dedicated to anti-lock braking systems (ABSs). The TSK fuzzy model structure and initial TSK fuzzy models were obtained by the modal equivalence principle in terms of placing local state-space models in the domain of TSK fuzzy models. The optimization problems were defined to minimize objective functions expressed as the average of the square modeling errors over the time horizon. Two representative nature-inspired algorithms, simulated annealing (SA) and PSO, were implemented to solve the optimization problems and to obtain optimal TSK fuzzy models.

Baruah et al. [92] proposed a new online evolving clustering approach for streaming data. Unlike other approaches, which consider either the data density or distance from existing cluster centers, this approach uses cluster weight and distance before generating new clusters. To capture the dynamics of the data stream, the cluster weight is defined in both data and time space in such a way that it decays exponentially with time. A distinction is made between core and noncore clusters to effectively identify the real outliers. Experimental results with developed models showed that the proposed approach obtains results at par or better than existing approaches and significantly reduces the computational overhead, which makes it suitable for real-time applications.

Yi et al. [93] proposed a novel quantum-inspired MBO methodology, called QMBO, by incorporating quantum computation into the basic MBO algorithm. In QMBO, a certain number of the worst butterflies are updated by quantum operators. The path planning navigation problem for unmanned combat air vehicles (UCAVs) was modeled, and its optimal path was obtained by the proposed QMBO algorithm. Furthermore, B-Spline curves were utilized to refine the obtained path, making it more suitable for UCAVs. The UCAV path obtained by QMBO was studied and analyzed in comparison with the basic MBO. Experimental results showed that QMBO can find a much shorter path than MBO.

Ghetas et al. [94] incorporated the harmony search (HS) algorithm into the basic MBO algorithm, and proposed a variant of MBO, called MBHS, to deal with the standard benchmark problems. In MBHS, the HS algorithm is considered as a mutation operator to improve the butterfly adjusting operator, with the aim of accelerating the convergence rate of MBO.

Feng et al. [95] presented a novel binary MBO (BMBO) method used to address the 0-1 knapsack problem (0-1 KP). In BMBO, each butterfly individual is represented as a two-tuple string. Several individual allocation techniques are used to improve BMBO's performance. In order to keep the number of infeasible solutions to a minimum, a novel repair operator was applied. The comparative study of BMBO with other optimization techniques showed the superiority of the former in solving 0-1 KP.

Wang et al. [96,97] put forward another variant of the MBO method, the GCMBO. In GCMBO, two modification strategies, including a self-adaptive crossover (SAC) operator and a greedy strategy, were utilized to improve its search ability.

Feng et al. [98] combined chaos theory with the basic MBO algorithm, and proposed a novel chaotic MBO (CMBO) algorithm. The proposed CMBO algorithm enhanced the search effectiveness significantly. In CMBO, in order to tune the two main operators, the best chaotic map is selected from 12 maps. Meanwhile, some of the worst individuals are improved by using a Gaussian mutation operator to avoid premature convergence.

Ghanem and Jantan [99] combined ABC with elements from MBO to proposed a new hybrid metaheuristic algorithm named hybrid ABC/MBO (HAM). The combined method uses an updated butterfly adjusting operator, considered to be a mutation operator, with the aim of sharing information with the employee bees in ABC.

Wang et al. [100] proposed a discrete version of MBO (DMBO) that was applied successfully to tackle the Chinese TSP (CTSP). They also studied and analyzed the parameter butterfly adjusting rate (BAR). The chosen BAR was used to find the best solution for the CTSP.

Feng et al. [101] proposed a type of multi-strategy MBO (MMBO) technique for the discounted 0-1 knapsack problem (DKP). In MMBO, two modifications, including neighborhood mutation and Gaussian perturbation, are utilized to retain the diversity of the population. An array of experimental results showed that the neighborhood mutation and Gaussian perturbation were quite capable of providing significant improvement in the exploration and exploitation of the MMBO approach, respectively. Accordingly, two kinds of NMBO were proposed: NCMBO and GMMBO.

Feng et al. [102] combined MBO with seven kinds of DE mutation strategies, using the intrinsic mechanism of the search process of MBO and the character of the differential mutation operator. They presented a novel DEMBO based on MBO and an improved DE mutation strategy. In this work, the migration operator was replaced by a differential mutation operator with the aim of improving its global optimization ability. The overall performance of DEMBO was fully assessed using thirty typical discounted 0-1 knapsack problem instances. The experimental results demonstrated that DEMBO could enhance the search ability while not increasing the time complexity. Meanwhile, the approximation ratio of all the 0-1 KP instances obtained by DEMBO were close to 1.0.

Wang et al. [103] proposed a new population initialization strategy in order to improve MBO's performance. First, the whole search space is equally divided into *NP* (population size) parts at each dimension. Subsequently, two random distributions (the *T* and *F* distributions) are used to mutate the equally divided population. Accordingly, five variants of MBOs are proposed with a new initialization strategy.

Feng et al. [104] presented OMBO, a generalized opposition-based learning (OBL) [105,106] MBO with Gaussian perturbation. The authors used the OBL strategy on the portion of the individuals in the late stage of evolution, and used Gaussian perturbation on the individuals with poor fitness in each evolution. OBL guaranteed a higher convergence speed of OMBO, and Gaussian perturbation avoided the possibility of falling into a local optimum. For the sake of testing and verifying the effectiveness of OMBO, three categories of 15 large-scale 0-1 KP cases from 800 to 2000 dimensions were used. The experimental results indicated that OMBO could find high-quality solutions.

Chen et al. [107] proposed a new variant of MBO by introducing a greedy strategy to solve dynamic vehicle routing problems (DVRPs). In contrast to the basic MBO algorithm, the proposed algorithm accepted only butterfly individuals that had better fitness than before implementation of the migration and butterfly adjusting operators. Also, a later perturbation procedure was introduced to make a trade-off between global and local search.

Meng et al. [108] proposed an improved MBO (IMBO) for the sake of enhancing the optimization ability of MBO. In IMBO, the authors divided the two subpopulations in a dynamic and random fashion at each generation, instead of using the fixed strategy applied in the original MBO approach. Also, the butterfly individuals were updated in two different ways for the sake of maintaining the diversity of the population.

Faris et al. [109] modified the position updating strategy used in the basic MBO algorithm by utilizing both the previous solutions and the butterfly individuals with the best fitness at the time. For the sake of fully exploring the search behavior of the improved MBO (IMBO), it was benchmarked by 23 functions. Furthermore, the IMBO was applied to train neural networks. The IMBO-based trainer was verified on 15 machine learning datasets from the UCI repository. Experimental results showed that the IMBO algorithm could enhance the learning ability of neural networks significantly.

Ehteram et al. [110] used the MBO algorithm to address the utilization of a multi-reservoir system for the sake of improving production of hydroelectric energy. They studied three periods of dry (1963–1964), wet (1951–1952), and normal (1985–1986) conditions in a four reservoir system. The experiments indicated that MBO can generate more energy compared to particle swarm optimization (PSO) and the genetic algorithm (GA).

Though many scholars have made several in-depth studies of the MBO algorithm from different aspects. The number of monarch butterflies in land 1 and 2 is unchanged. In this paper, a self-adaptive strategy is introduced to update the subpopulation sizes during the optimization process. A detailed description of the proposed algorithm will be given in the following sections.

3. MBO Algorithm

3.1. Migration Operator

The number of butterflies located at land 1 and land 2 can be calculated as ceil($p*NP$) (NP_1, subpopulation 1, SP1) and $NP - NP_1$ (NP_2, subpopulation 2, SP2), respectively. We use SP1 and SP2 to denote subpopulation 1 and subpopulation 2, respectively. Here, ceil(x) rounds x to the nearest integer not less than x. Therefore, when $r \leq p$, then $x_{i,k}^{t+1}$ is generated by the following equation [88]:

$$x_{i,k}^{t+1} = x_{r_1,k}^{t}, \tag{1}$$

where $x_{i,k}^{t+1}$ is the kth element of x_i, and $x_{r_1,k}^{t}$ is the kth element of x_{r_1}. Butterfly r_1 is chosen from SP1 in a random fashion. In Equation (1), r is given in the following form:

$$r = rand * peri, \tag{2}$$

where $peri$ is the migration period [88]. In comparison, when $r > p$, then $x_{r_1,k}^{t}$ is given by:

$$x_{i,k}^{t+1} = x_{r_2,k}^{t}, \tag{3}$$

where $x_{r_2,k}^{t}$ is the kth element of x_{r_2}, and butterfly r_2 is chosen from SP2 in a random fashion.

3.2. Butterfly Adjusting Operator

For butterfly j, if $rand$ is not more than p, the kth element is given as [88]:

$$x_{j,k}^{t+1} = x_{best,k}^{t}, \tag{4}$$

where $x_{j,k}^{t+1}$ is the kth element of x_j. Similarly, $x_{best,k}^{t}$ is the kth element of the best individual x_{best}. On the other hand, when $rand$ is bigger than p, it can be expressed as:

$$x_{j,k}^{t+1} = x_{r_3,k}^{t}, \tag{5}$$

where $x_{r_3,k}^{t}$ is the kth element of x_{r_3}. Here, $r_3 \in \{1, 2, \ldots, NP_2\}$.

In this case, when *rand* is bigger than *BAR*, it can be calculated in another form [88]:

$$x_{j,k}^{t+1} = x_{j,k}^{t+1} + \alpha \times (dx_k - 0.5) \ , \tag{6}$$

where dx is the walk step of butterfly j.

4. SPMBO Algorithm

Even though the MBO algorithm was proposed only three years ago, it has received more and more attention from scholars and engineers [88]. They have put forward many techniques to improve the search ability of the basic MBO algorithm. Also, MBO has been used to successfully solve all kinds of real-world problems. However, as mentioned previously, MBO uses a fixed number of butterflies in land 1 and land 2, and all the new butterfly individuals generated by the migration operator are accepted. In this paper, a new variant of the MBO algorithm will be proposed by introducing self-adaptive and greedy strategies. A detailed description of the SPMBO algorithm will be given below.

4.1. Self-Adaptive Strategy

As mentioned in Section 3.1, the number of butterflies in land 1 and land 2 are ceil(p*NP) (NP_1, subpopulation 1) and NP-NP_1 (NP_2, subpopulation 2), respectively. They are fixed and unchanged during the entire optimization process. Here, the parameter p is adjusted by a self-adaptive strategy in a dynamic way, which is updated as follows:

$$p = a + bt \ , \tag{7}$$

where t is current generation, and a and b are constants given by:

$$a = \frac{p_{\min} t_m - p_{\max}}{t_m - 1} \ , \tag{8}$$

$$b = \frac{p_{\max} - p_{\min}}{t_m - 1} \ , \tag{9}$$

where t_m is the maximum generation, and $p_{\min}$ and $p_{\max}$ are lower and upper bounds of parameter p, respectively. Clearly, $p_{\min}$ and $p_{\max}$ are in the range [0, 1].

For the basic MBO algorithm, when $p = 0$, all the butterflies are updated by the butterfly adjusting operator, while when $p = 1$, all the butterflies are updated by the migration operator. Apart from these two special cases, in order to extend the range of the parameter p, $p_{\min}$ and $p_{\max}$ are respectively assigned to 0.1 and 0.9 in our following experiments. From Equation (7), we can see, the parameter p is changed in a linear way from the lower bound $p_{\min}$ to upper bound $p_{\max}$.

4.2. Greedy Strategy

In this subsection, we will make a further in-depth study on the migration operator. In the basic MBO algorithm, all the newly-generated butterfly individuals are accepted as the new butterfly individuals for the next generation. If the newly-generated butterfly individual is worse than before, this updating will lead to population degradation, and slow the convergence speed. More seriously, if this happens at the later stages of the search, the population will oscillate.

In this paper, a greedy strategy is introduced in the basic MBO algorithm. Only newly-generated butterfly individuals with better fitness will be accepted and passed to the next generation. This selection scheme guarantees the generated population is not worse than before, and the algorithm evolves in the proper way. After introducing the greedy strategy, for minimal problems, the new butterfly individual is given as:

$$x_{i,new}^{t+1} = \begin{cases} x_i^{t+1}, & f(x_i^{t+1}) < f(x_i^t) \\ x_i^t, & else \end{cases} , \tag{10}$$

where $x_{i,new}^{t+1}$ is a newly-generated butterfly that will be passed to the next generation, and $f(x_i^{t+1})$ and $f(x_i^t)$ are the fitness of butterfly x_i^{t+1} and x_i^t, respectively.

After introducing this greedy strategy to the migration operator, the mainframe of the updated migration operator can be constructred, as shown in Algorithm 1.

Algorithm 1 Updated Migration Operator.

```
1: for $i = 1$ to $NP_1$ do   // butterflies in SP1
2:     for $k = 1$ to $D$ do   // $D$ is the length of a butterfly individual
3:         Randomly generate a number $rand$.
4:         $r = rand * peri$.
5:         if $r \leq p$ then
6:             Randomly select a butterfly in subpopulation 1 (say $r_1$);
7:             Generate $x_{i,k}^{t+1}$ by Equation (1);
8:         else
9:             Randomly select a butterfly in subpopulation 2 (say $r_2$);
10:            Generate $x_{i,k}^{t+1}$ by Equation (3);
11:        end if
12:    end for
13:    Generate $x_{i,new}^{t+1}$ according to greedy strategy as shown in Equation (10).
14: end for
```

After incorporating the self-adaptive strategy and greedy strategy into the basic MBO algorithm, the SPMBO approach is complete; a description of the approach is given in Algorithm 2.

Algorithm 2 SPMBO Algorithm.

```
1: Initialization. Set the generation counter $t = 1$, and set the maximum generation
   $t_m$, $NP_1$, $NP_2$, $BAR$, $peri$, and the lower ($p_{\min}$) and upper ($p_{\max}$) bounds of parameter $p$.
2: Population evaluation. Calculate the fitness according to the objective function.
3: while $t < t_m$ do
4:     Sort the butterfly population.
5:     Compute parameter $p$ according to Equation (7).
6:     Determine the number of butterflies in land 1 ($NP_1$) and land 2 ($NP_2$).
7:     Divide butterfly individuals into two subpopulations.
8:     for $i = 1$ to $NP_1$ do   // butterflies in SP1
9:         Perform updated migration operator as Algorithm 1.
10:    end for
11:    for $j = 1$ to $NP_2$ do   // butterflies in SP2
12:        Perform butterfly adjusting operator as the basic MBO algorithm.
13:    end for
14:    Calculate the fitness of newly-generated butterfly individuals.
15:    $t = t + 1$.
16: end while
17: Print the final solution.
```

In order to verify the performance of the prosed SPMBO algorithm, thirteen benchmark problems are solved by the proposed approach. The thirteen benchmark problems are minimal functions, so the

MBO and SPMBO algorithms are striving to search for the smallest possible function values. A more detailed description of these experiments can be found in Section 5.

5. Simulation Results

In this section, the proposed SPMBO algorithm will be fully verified from various aspects on thirteen 30-D and 60-D standard benchmark problems, as shown in Table 1. A more detailed description of the benchmark problems can be found on the website: http://www.sfu.ca/~ssurjano/optimization.html. In order to get a fair comparison, all implementations are carried out under the same conditions [49,111].

Table 1. Benchmark functions.

No.	Name	lb	ub	opt	Separability	Modality
F01	Alpine	-10	10	0	Separable	Multimodal
F02	Brown	-1	4	0	Nonseparable	Unimodal
F03	Dixon & Price	-10	10	0	Nonseparable	Multimodal
F04	Fletcher-Powell	$-\pi$	π	0	Nonseparable	Multimodal
F05	Holzman 2	-10	10	0	Separable	Multimodal
F06	Levy	-10	10	0	Nonseparable	Multimodal
F07	Penalty #1	-50	50	0	Nonseparable	Multimodal
F08	Penalty #2	-50	50	0	Nonseparable	Multimodal
F09	Perm	$-D$	D	0	Separable	Multimodal
F10	Powell	-4	5	0	Separable	Unimodal
F11	Rastrigin	-5.12	5.12	0	Nonseparable	Multimodal
F12	Schwefel 2.21	-100	100	0	Nonseparable	Unimodal
F13	Zakharov	-5	10	0	Nonseparable	Unimodal

5.1. MBO vs. SPMBO

In this subsection, we compare the proposed SPMBO algorithm with the basic MBO approach. For MBO and SPMBO, the corresponding parameters are set as follows: max step $S_{\max} = 1.0$, butterfly adjusting rate $BAR = 5/12$, maximum generation $t_m = 50$, migration period $peri = 1.2$, the migration ratio $p = 5/12$, the upper bound $p_{\max} = 0.9$, lower bound $p_{\min} = 0.1$, and population size $NP = 50$. Therefore, for the basic MBO, the number of butterflies in land 1 and land 2, i.e., NP_1 and NP_2, are 21 and 29, respectively. According to Equations (7)–(9), we find $a = 41/490$, $b = 8/490$, and the parameter p, is given by:

$$p = \frac{41}{490} + \frac{8}{490}t\,, \tag{11}$$

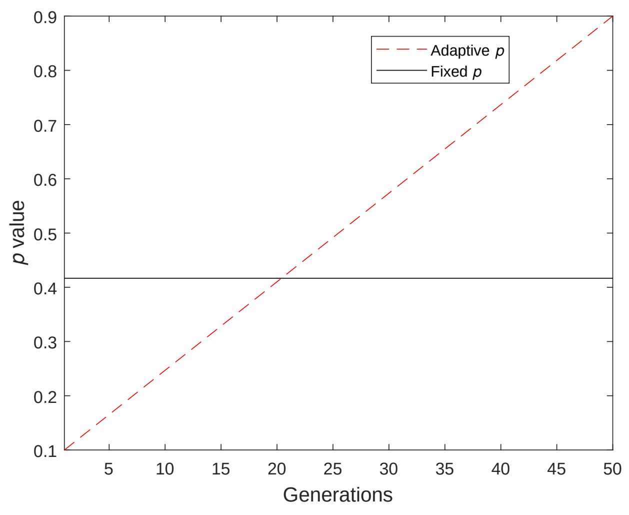
where t is current generation, which is an integer between 1 and 50 in our present work.

According to our above analyses, the trend of parameter p, the number of butterflies in land 1 NP_1, and the number of butterflies in land 2 NP_2 for both the basic MBO algorithm and the proposed SPMBO algorithm are illustrated in Figure 1.

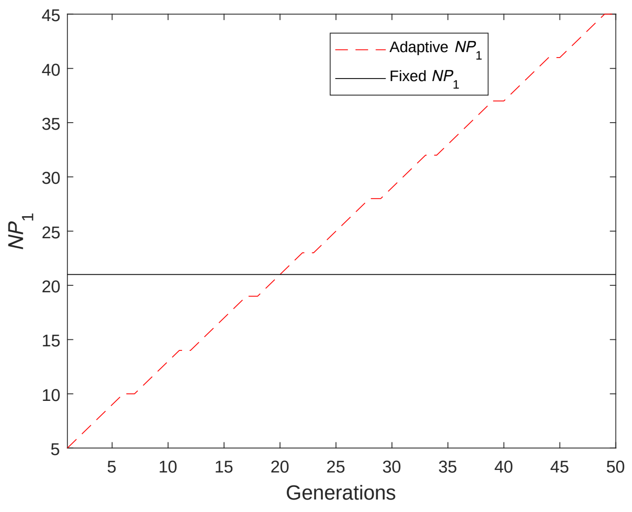
In essence, all the intelligent algorithms are stochastic algorithms, therefore, in order to remove the influence of randomness, thirty independent runs are performed. In the following experiments, the optimal solution for each test problem is highlighted in **bold** font.

5.1.1. $D = 30$

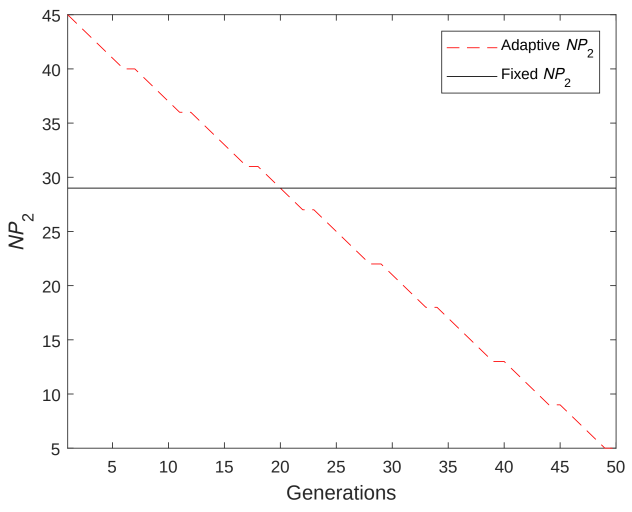
In this subsection, the dimension of thirteen benchmarks are set to 30. For MBO and SPMBO, they have the same function evaluations (FEs) at each generation. Therefore, the maximum generation (t_m) is considered as the stop condition, which is 50 as mentioned above. For the thirty implementations, the best, mean, and worst function values and Standard deviation (Std) values obtained by MBO and SPMBO are recorded in Table 2.



(**a**)



(**b**)



(**c**)

Figure 1. Trends of the parameter p, the number of butterflies in land 1 NP_1, and the number of butterflies in land 2 NP_2 for the tested monarch butterfly optimization (MBO) and self-adaptive population MBO (SPMBO) algorithms. (**a**) p; (**b**) NP_1; (**c**) NP_2.

From Table 2, it can be observed that, in terms of the mean and worst function values, SPMBO has the absolute advantage over the MBO algorithm on the thirteen benchmarks. By studying the SD values, we can see, the final functions values obtained by SPMBO are located in a smaller range than the basic MBO algorithm. This indicates that SPMBO is an effective intelligent algorithm that performs in a more stable way. Unfortunately, for best function values, SPMBO only narrowly defeated the basic MBO algorithm (7 vs. 6).

The superiority of SPMBO on functions F01–F13 is also shown in Figure 2, which clearly reveals that SPMBO performs better than MBO throughout the entire optimization process.

Table 2. Best, mean, and worst function values and SD values obtained by MBO and SPMBO algorithms with dimension $D = 30$.

	Best		**Mean**		**Worst**		**SD**	
	MBO	**SPMBO**	**MBO**	**SPMBO**	**MBO**	**SPMBO**	**MBO**	**SPMBO**
F01	0.10	**0.05**	12.93	**7.12**	58.87	**34.25**	17.19	**8.24**
F02	0.02	$\mathbf{9.13 \times 10^{-4}}$	196.00	**85.99**	1.85×10^{3}	**512.20**	493.80	**148.50**
F03	31.53	**7.22**	3.57×10^{8}	$\mathbf{1.97 \times 10^{8}}$	1.16×10^{9}	$\mathbf{6.52 \times 10^{8}}$	3.54×10^{8}	$\mathbf{2.19 \times 10^{8}}$
F04	4.63×10^{5}	$\mathbf{3.23 \times 10^{5}}$	8.46×10^{5}	$\mathbf{7.03 \times 10^{5}}$	1.58×10^{6}	$\mathbf{1.01 \times 10^{6}}$	2.85×10^{5}	$\mathbf{1.97 \times 10^{5}}$
F05	22.94	$\mathbf{3.98 \times 10^{-3}}$	2.53×10^{5}	$\mathbf{1.21 \times 10^{5}}$	5.90×10^{5}	$\mathbf{5.78 \times 10^{5}}$	1.93×10^{5}	$\mathbf{1.36 \times 10^{5}}$
F06	$\mathbf{2.64 \times 10^{-3}}$	9.71×10^{-3}	46.65	**19.89**	206.80	**121.90**	62.63	**33.74**
F07	$\mathbf{1.49 \times 10^{-5}}$	2.18×10^{-4}	7.25×10^{7}	$\mathbf{5.58 \times 10^{7}}$	4.43×10^{8}	$\mathbf{4.23 \times 10^{8}}$	1.26×10^{8}	$\mathbf{9.18 \times 10^{7}}$
F08	**0.27**	1.15	3.64×10^{8}	$\mathbf{8.89 \times 10^{7}}$	1.12×10^{9}	$\mathbf{6.83 \times 10^{8}}$	4.28×10^{8}	$\mathbf{1.66 \times 10^{8}}$
F09	0.85	**0.21**	2.29×10^{22}	$\mathbf{3.35 \times 10^{16}}$	6.77×10^{23}	$\mathbf{1.09 \times 10^{18}}$	1.22×10^{23}	$\mathbf{1.80 \times 10^{17}}$
F10	**0.67**	1.09	3.16×10^{3}	$\mathbf{3.02 \times 10^{3}}$	1.24×104	$\mathbf{1.11 \times 10^{4}}$	3.58×10^{3}	$\mathbf{3.11 \times 10^{3}}$
F11	**0.05**	3.39	106.00	**80.14**	234.40	**201.30**	84.42	**55.02**
F12	**0.47**	0.48	45.50	**23.55**	121.10	**86.12**	45.26	**28.11**
F13	31.23	**1.49**	541.90	**417.60**	1.93×10^{3}	$\mathbf{1.06 \times 10^{3}}$	375.30	**318.70**
Total	6	7	0	13	0	13	0	13

5.1.2. $D = 60$

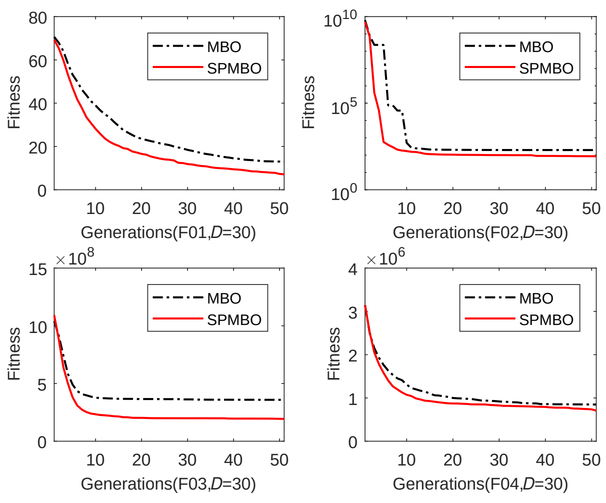
In this subsection, the dimension of the thirteen benchmark problems are set to 60. As before, the maximum generation (t_m), i.e., function evaluations (FEs), is considered as the stop condition, which is again set to 50. After thirty implementations, the best, mean, and worst function values and SD values were obtained by MBO and SPMBO and are recorded in Table 3.

From Table 3, it can be observed that, for mean and worst function values, SPMBO has the absolute advantage over the MBO algorithm on the thirteen benchmarks, although MBO does perform better than SPMBO on two of the test functions. On ten benchmark problems, SPMBO has smaller SD values than MBO. This indicates that SPMBO performs more stably than MBO in most cases. However, for best function values, SPMBO is narrowly defeated by the basic MBO algorithm (6 vs. 7). Future research of SPMBO should foucus on how to improve the best function values.

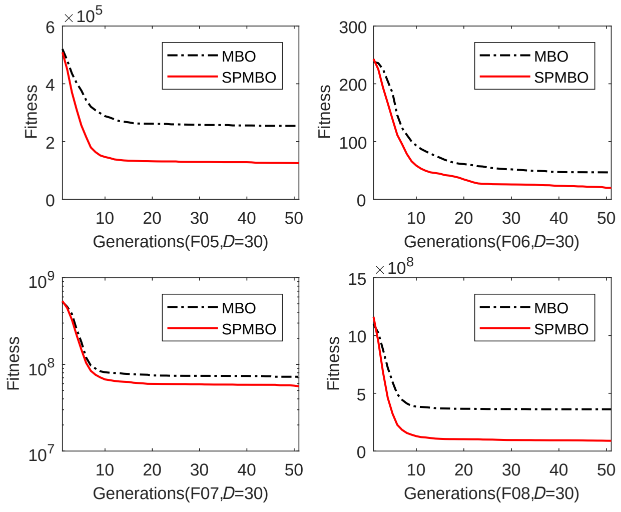
The superiority of SPMBO on functions F01–F13 can also be seen in Figure 3, which clearly reveals that SPMBO performs better than MBO during the entire optimization process.

5.2. PSO vs. SPMBO

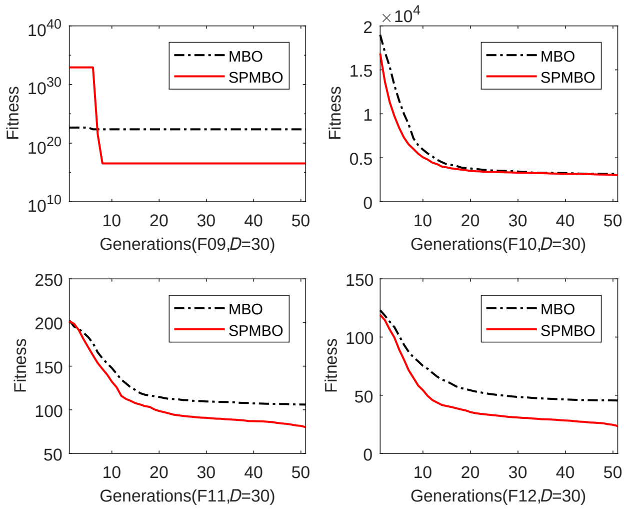
In order to further show the superiority of the proposed SPMBO algorithm, we compare the SPMBO with other metaheuristic algorithms. Here, PSO is taken as an example. The parameters in SPMBO are the same as in Section 5.1. For PSO, the parameters are chosen as: the inertial constant =0.3, the cognitive constant =1, and the social constant for swarm interaction =1. As mentioned in Section 5.1, thirty independent runs are performed with the aim of getting representative results. The optimal solution for each test problem is highlighted in **bold** font.



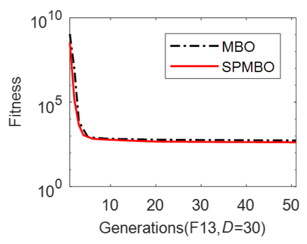
(**a**)



(**b**)

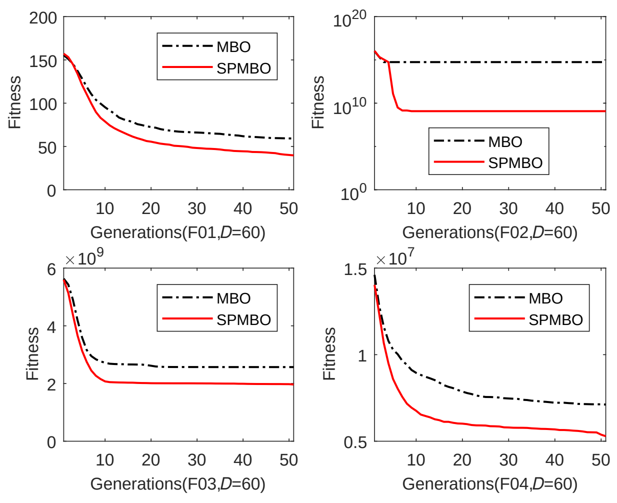


(**c**)

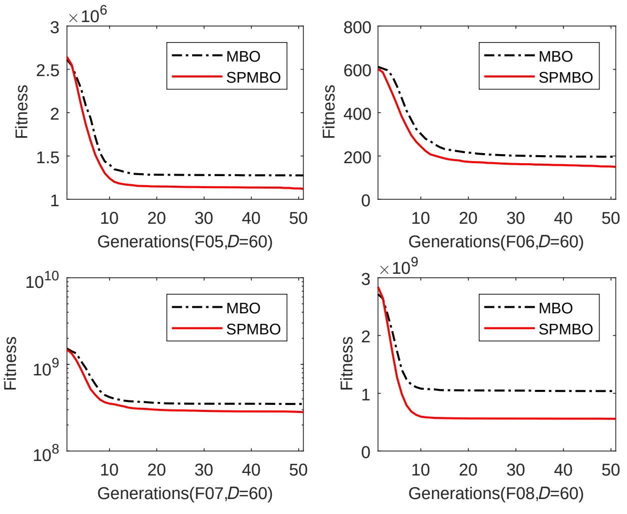


(**d**)

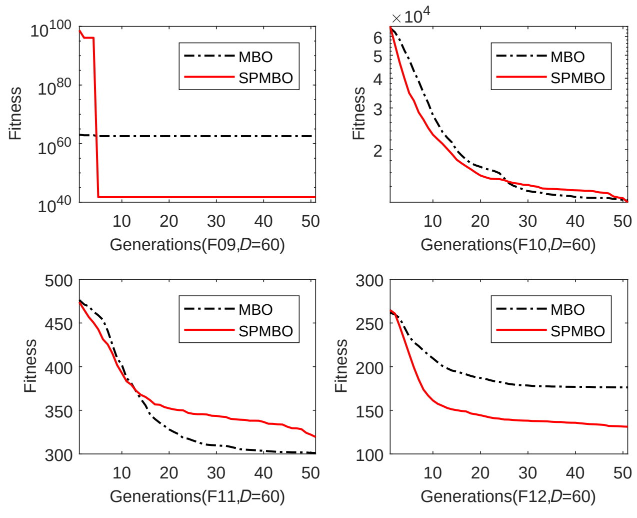
Figure 2. Convergence curves for the thirteen the functions with dimension $D = 30$. (**a**) F01–F04; (**b**) F05–F08; (**c**) F09–F12; and (**d**) F13.



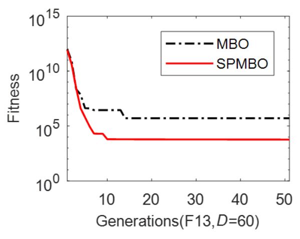
(**a**)



(**b**)



(**c**)



(**d**)

Figure 3. Convergence curves of the thirteen test functions with $D = 60$. (**a**) F01–F04; (**b**) F05–F08; (**c**) F09–F12; and (**d**) F13.

Table 3. Best, mean, and worst function values and SD values obtained by the MBO and SPMBO algorithms with $D = 60$.

	Best		**Mean**		**Worst**		**SD**	
	MBO	**SPMBO**	**MBO**	**SPMBO**	**MBO**	**SPMBO**	**MBO**	**SPMBO**
F01	**0.03**	0.12	59.21	**39.72**	153.90	**133.30**	61.82	**43.95**
F02	**0.05**	0.38	5.50×10^{14}	$\mathbf{1.14 \times 10^{9}}$	1.68×10^{16}	$\mathbf{1.78 \times 10^{10}}$	3.05×10^{15}	$\mathbf{4.34 \times 10^{9}}$
F03	1.34×10^{4}	**16.56**	2.50×10^{9}	$\mathbf{1.96 \times 10^{9}}$	6.33×10^{9}	$\mathbf{6.03 \times 10^{9}}$	2.12×10^{9}	$\mathbf{2.03 \times 10^{9}}$
F04	5.20×10^{6}	$\mathbf{3.18 \times 10^{6}}$	7.19×10^{6}	$\mathbf{5.21 \times 10^{6}}$	9.68×10^{6}	$\mathbf{7.14 \times 10^{6}}$	1.33×10^{6}	$\mathbf{9.78 \times 10^{5}}$
F05	**24.82**	117.20	1.26×10^{6}	$\mathbf{1.11 \times 10^{6}}$	3.02×10^{6}	$\mathbf{2.76 \times 10^{6}}$	$\mathbf{1.07 \times 10^{6}}$	1.17×10^{6}
F06	**0.02**	0.21	196.60	**149.60**	**625.20**	635.90	228.00	**160.10**
F07	$\mathbf{8.62 \times 10^{-4}}$	0.08	3.44×10^{8}	$\mathbf{2.84 \times 10^{8}}$	1.54×10^{9}	$\mathbf{1.50 \times 10^{9}}$	5.46×10^{8}	$\mathbf{4.93 \times 10^{8}}$
F08	$\mathbf{9.85 \times 10^{-3}}$	5.03	1.08×10^{9}	$\mathbf{5.55 \times 10^{8}}$	3.23×10^{9}	$\mathbf{2.69 \times 10^{9}}$	1.10×10^{9}	$\mathbf{8.06 \times 10^{8}}$
F09	**3.49**	4.18	3.57×10^{62}	$\mathbf{5.23 \times 10^{41}}$	1.01×10^{64}	$\mathbf{7.89 \times 10^{42}}$	1.97×10^{63}	$\mathbf{1.91 \times 10^{42}}$
F10	25.18	**2.91**	1.29×10^{4}	$\mathbf{1.22 \times 10^{4}}$	5.67×10^{4}	$\mathbf{3.98 \times 10^{4}}$	$\mathbf{1.22 \times 10^{4}}$	1.37×10^{4}
F11	60.44	**36.61**	**301.10**	319.50	**492.80**	517.00	158.90	**142.00**
F12	4.32	**1.51**	176.20	**131.20**	271.60	**260.80**	**86.66**	95.25
F13	165.70	**8.76**	5.02×10^{5}	$\mathbf{5.82 \times 10^{3}}$	9.16×10^{6}	$\mathbf{6.75 \times 10^{4}}$	1.93×10^{6}	$\mathbf{1.44 \times 10^{4}}$
Total	7	6	1	12	2	11	3	10

5.2.1. $D = 30$

In this subsection, the dimension of the thirteen benchmarks is set to 30. PSO and SPMBO have the same function evaluations (FEs) at each generation. Therefore, the maximum generation (t_m) is considered as the stop condition, which is 50 as mentioned above. After thirty implementations, the best, mean, and worst function values and SD values obtained by PSO and SPMBO were recorded and are shown in Table 4.

From Table 4, it can be observed that, for best and mean function values, SPMBO has the absolute advantage over the PSO algorithm. Through studying the worst values, we can see, the final function values obtained by SPMBO are a little better than PSO. Unfortunately, for SD function values, SPMBO is defeated by PSO .

Table 4. Best, mean, worst function values and SD values obtained by the PSO and SPMBO algorithms with dimension $D = 30$.

	Best		**Mean**		**Worst**		**SD**	
	PSO	**SPMBO**	**PSO**	**SPMBO**	**PSO**	**SPMBO**	**PSO**	**SPMBO**
F01	29.95	**0.04**	36.00	**5.28**	43.13	**24.37**	**3.06**	6.45
F02	96.30	**0.10**	357.50	**96.33**	1.37×10^{3}	$\mathbf{1.32 \times 10^{3}}$	**251.00**	274.20
F03	2.22×10^{7}	$\mathbf{6.96 \times 10^{3}}$	2.93×10^{8}	$\mathbf{2.83 \times 10^{8}}$	1.28×10^{9}	$\mathbf{8.86 \times 10^{8}}$	4.31×10^{8}	$\mathbf{3.06 \times 10^{8}}$
F04	9.63×10^{5}	$\mathbf{4.07 \times 10^{5}}$	1.71×10^{6}	$\mathbf{8.43 \times 10^{5}}$	2.87×10^{6}	$\mathbf{1.49 \times 10^{6}}$	3.49×10^{5}	$\mathbf{2.56 \times 10^{5}}$
F05	2.39×10^{4}	$\mathbf{7.19 \times 10^{-3}}$	$\mathbf{4.62 \times 10^{4}}$	7.49×10^{4}	$\mathbf{2.09 \times 10^{5}}$	3.52×10^{5}	$\mathbf{3.24 \times 10^{4}}$	1.13×10^{5}
F06	38.02	$\mathbf{5.18 \times 10^{-3}}$	80.32	**36.60**	**115.80**	189.20	**16.08**	58.80
F07	4.03×10^{6}	$\mathbf{2.28 \times 10^{-4}}$	$\mathbf{2.66 \times 10^{7}}$	7.46×10^{7}	$\mathbf{6.70 \times 10^{7}}$	5.75×10^{8}	$\mathbf{1.62 \times 10^{7}}$	1.60×10^{8}
F08	3.39×10^{7}	**0.02**	$\mathbf{9.37 \times 10^{7}}$	1.92×10^{8}	$\mathbf{1.96 \times 10^{8}}$	8.58×10^{8}	$\mathbf{3.99 \times 10^{7}}$	2.75×10^{8}
F09	$\mathbf{5.53 \times 10^{-3}}$	6.68	**3.14**	2.29×10^{22}	**27.13**	6.77×10^{23}	**6.10**	1.22×10^{23}
F10	1.73×10^{3}	**3.27**	3.16×10^{3}	$\mathbf{3.10 \times 10^{3}}$	$\mathbf{5.21 \times 10^{3}}$	9.37×10^{3}	**976.60**	2.94×10^{3}
F11	180.00	**1.20**	204.80	**89.07**	240.00	**221.30**	**18.34**	60.96
F12	107.00	**0.86**	120.40	**36.15**	138.00	**103.90**	**6.47**	35.62
F13	231.80	**20.83**	569.30	**459.80**	2.39×10^{3}	**751.00**	372.00	**220.90**
Total	1	12	4	9	6	7	10	3

5.2.2. $D = 60$

In this subsection, the dimension of the thirteen benchmark problems is set to 60. As before, the maximum generation (t_m), i.e., function evaluations (FEs), is considered as the stop condition and is set to 50. After thirty implementations, the best, mean, and worst function values and SD values were obtained for PSO and SPMBO and are recorded in Table 5.

From Table 5, it can be observed that, for best and mean function values, SPMBO has the absolute advantage over the PSO algorithm on all thirteen benchmarks. For worst values, SPMBO and PSO have similar performance. Unfortunately, for SD values, SPMBO is defeated by PSO.

Table 5. Best, mean, worst function values and SD values obtained by the PSO and SPMBO algorithms with dimension $D = 60$.

	Best		Mean		Worst		SD	
	PSO	**SPMBO**	**PSO**	**SPMBO**	**PSO**	**SPMBO**	**PSO**	**SPMBO**
F01	74.61	**0.45**	85.45	**53.63**	**92.23**	135.90	**4.27**	43.22
F02	1.97×10^{3}	**0.81**	$\mathbf{4.91 \times 10^{5}}$	6.48×10^{8}	$\mathbf{4.92 \times 10^{6}}$	1.78×10^{10}	$\mathbf{1.06 \times 10^{6}}$	3.11×10^{9}
F03	5.23×10^{8}	$\mathbf{2.46 \times 10^{4}}$	4.00×10^{9}	$\mathbf{1.63 \times 10^{9}}$	8.91×10^{9}	$\mathbf{6.08 \times 10^{9}}$	3.04×10^{9}	$\mathbf{1.87 \times 10^{9}}$
F04	8.48×10^{6}	$\mathbf{3.95 \times 10^{6}}$	1.07×10^{7}	$\mathbf{6.09 \times 10^{6}}$	1.33×10^{7}	$\mathbf{8.45 \times 10^{6}}$	1.18×10^{6}	$\mathbf{1.00 \times 10^{6}}$
F05	2.03×10^{5}	**18.17**	1.49×10^{6}	$\mathbf{1.00 \times 10^{6}}$	3.78×10^{6}	$\mathbf{3.07 \times 10^{6}}$	1.26×10^{6}	$\mathbf{9.74 \times 10^{5}}$
F06	185.80	**0.07**	237.00	**131.20**	**303.80**	522.10	**31.63**	163.90
F07	1.03×10^{8}	$\mathbf{9.80 \times 10^{-4}}$	$\mathbf{1.95 \times 10^{8}}$	2.58×10^{8}	$\mathbf{3.17 \times 10^{8}}$	1.41×10^{9}	$\mathbf{5.91 \times 10^{7}}$	4.68×10^{8}
F08	2.01×10^{8}	**0.77**	$\mathbf{4.29 \times 10^{8}}$	5.82×10^{8}	$\mathbf{7.92 \times 10^{8}}$	2.94×10^{9}	$\mathbf{1.30 \times 10^{8}}$	9.43×10^{8}
F09	0.04	0.50	**19.92**	3.57×10^{62}	**115.80**	1.01×10^{64}	**27.03**	1.97×10^{63}
F10	$\mathbf{6.80 \times 10^{3}}$	**3.44**	1.31×10^{4}	$\mathbf{5.51 \times 10^{3}}$	$\mathbf{1.80 \times 10^{4}}$	3.65×10^{4}	$\mathbf{3.10 \times 10^{3}}$	8.30×10^{3}
F11	352.00	**37.45**	467.70	**290.60**	522.90	**513.00**	**40.55**	159.70
F12	228.00	**18.07**	261.80	**115.30**	288.00	**251.80**	**11.26**	73.14
F13	914.10	**12.95**	1.82×10^{9}	$\mathbf{2.28 \times 10^{4}}$	4.37×10^{10}	$\mathbf{4.88 \times 10^{5}}$	7.97×10^{9}	$\mathbf{9.05 \times 10^{4}}$
Total	1	12	4	9	7	6	9	4

From the results in Tables 4 and 5, we can draw a brief conclusion, SPMBO performs better than PSO on most cases, though the SD of the SPMBO algorithm must be further improved.

6. Conclusions

Inspired by the migration behavior of monarch butterflies, one of the most promising swarm intelligence algorithms, monarch butterfly optimization (MBO), was proposed by Wang et al. in 2015. In the basic MBO algorithm, the number of butterflies in land 1 (NP_1) and land 2 (NP_2) is fixed, which is calculated according to the parameter p at the begin of the search. MBO includes two main operators: a migration operator and a butterfly adjusting operator. For the migration operator, all the generated butterfly individuals are accepted and passed to the next generation. In some cases, this is an ineffective way to find the best function values. In this paper, we introduced two techniques to overcome these drawbacks: self-adaptive and greedy strategies. The parameter p is linearly adjusted in a dynamic way. Therefore, at the beginning of the search, the number of butterflies in land 1 (NP_1) and land 2 (NP_2) is determined by the parameter p. Additionally, only newly-generated butterfly individuals having better fitness will be accepted and passed to the next generation. This greedy strategy will surely accelerate the convergence speed. The proposed SPMBO algorithm is tested by thirteen 30-D and 60-D test functions. The experimental results indicate that the search ability of the proposed SPMBO approach outperforms significantly the basic MBO algorithm on most test functions.

Despite showing various advantages of the SPMBO approach, the following points should be highlighted in our future research. On one hand, the parameter p is changed during the entire optimization process. In fact, if the algorithm performs in a good manner, there is no need to adjust the parameter p. Therefore, developing a method to adjust the parameter p in a more intelligent way is worthy of further studies. Second, for the updated migration operator, only better butterfly individuals are accepted and passed to the next generation. The butterfly individuals having worse fitness may include better elements, which may help the search algorithm. Thus, the migration operator should accept a few butterfly individuals with worse fitness. Last, only thirteen benchmarks were used to test our proposed SPMBO approach. In the future, more benchmark problems, especially real-world applications, should be used for further verifying SPMBO, such as image processing, video coding, and wireless sensor networks.

Author Contributions: H.H., Z.C., and S.H. implemented the algorithms and designed the experiments; Z.C., S.H., and Y.C. performed the experiments; Y.C., J.C., and S.H. collected and analyzed the data; H.H., Z.C., and J.C. wrote the paper.

Acknowledgments: This work was supported by the National Natural Science Foundation of China (No. 61772225), the Foundation for Distinguished Young Talents in Higher Education of Guangdong (No. 2015KQNCX153), Science and Technology Program of Huizhou (No. 2015B010002002, No. 2016X0431046, No. 2016X0434049, No. 2016X0432047, No. 2017c0406022, No. 2017c0407023, No. 2017c0414030).

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

References

1. Wang, G.G.; Tan, Y. Improving metaheuristic algorithms with information feedback models. *IEEE Trans. Cybern.* **2017**. [CrossRef]
2. Wang, G.G.; Cai, X.; Cui, Z.; Min, G.; Chen, J. High performance computing for cyber physical social systems by using evolutionary multi-objective optimization algorithm. *IEEE Trans. Emerg. Top. Comput.* **2017**. [CrossRef]
3. Wang, G.G.; Lu, M.; Dong, Y.Q.; Zhao, X.J. Self-adaptive extreme learning machine. *Neural Comput. Appl.* **2016**, *27*, 291–303. [CrossRef]
4. Yi, J.; Xu, W.; Chen, Y. Novel back propagation optimization by cuckoo search algorithm. *Sci. World J.* **2014**, *2014*, 1–8. [CrossRef] [PubMed]
5. Cui, Z.; Gao, X. Theory and applications of swarm intelligence. *Neural Comput. Appl.* **2012**, *21*, 205–206. [CrossRef]
6. Duan, H.; Luo, Q. New progresses in swarm intelligence-based computation. *Int. J. Bio-Inspir. Comput.* **2015**, *7*, 26–35. [CrossRef]
7. Torres-Treviño, L.M. Let the swarm be: an implicit elitism in swarm intelligence. *Int. J. Bio-Inspir. Comput.* **2017**, *9*, 65–76. [CrossRef]
8. Reddy, S.S.; Panigrahi, B. Application of swarm intelligent techniques with mixed variables to solve optimal power flow problems. *Int. J. Bio-Inspir. Comput.* **2017**, *10*, 283–292. [CrossRef]
9. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceeding of the IEEE International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
10. Mirjalili, S.; Wang, G.G.; Coelho, L.S. Binary optimization using hybrid particle swarm optimization and gravitational search algorithm. *Neural Comput. Appl.* **2014**, *25*, 1423–1435. [CrossRef]
11. Wang, G.G.; Gandomi, A.H.; Alavi, A.H.; Deb, S. A hybrid method based on krill herd and quantum-behaved particle swarm optimization. *Neural Comput. Appl.* **2016**, *27*, 989–1006. [CrossRef]
12. Wang, G.G.; Gandomi, A.H.; Yang, X.S.; Alavi, A.H. A novel improved accelerated particle swarm optimization algorithm for global numerical optimization. *Eng. Comput.* **2014**, *31*, 1198–1220. [CrossRef]
13. Zhao, X.; Song, B.; Huang, P.; Wen, Z.; Weng, J.; Fan, Y. An improved discrete immune optimization algorithm based on PSO for QoS-driven web service composition. *Appl. Soft Comput.* **2012**, *12*, 2208–2216. [CrossRef]
14. Zhao, X. A perturbed particle swarm algorithm for numerical optimization. *Appl. Soft Comput.* **2010**, *10*, 119–124. [CrossRef]
15. Sun, Y.; Jiao, L.; Deng, X.; Wang, R. Dynamic network structured immune particle swarm optimisation with small-world topology. *Int. J. Bio-Inspir. Comput.* **2017**, *9*, 93–105. [CrossRef]
16. Abdel-Basset, M.; Wang, G.G.; Sangaiah, A.K.; Rushdy, E. Krill herd algorithm based on cuckoo search for solving engineering optimization problems. *Multimed. Tools Appl.* **2017**. [CrossRef]
17. Liu, K.; Gong, D.; Meng, F.; Chen, H.; Wang, G.G. Gesture segmentation based on a two-phase estimation of distribution algorithm. *Inf. Sci.* **2017**, *394–395*, 88–105. [CrossRef]
18. Hu, Y.; Yin, M.; Li, X. A novel objective function for job-shop scheduling problem with fuzzy processing time and fuzzy due date using differential evolution algorithm. *Int. J. Adv. Manuf. Technol.* **2011**, *56*, 1125–1138. [CrossRef]
19. Rizk-Allah, R.M.; El-Sehiemy, R.A.; Deb, S.; Wang, G.G. A novel fruit fly framework for multi-objective shape design of tubular linear synchronous motor. *J. Supercomput.* **2017**, *73*, 1235–1256. [CrossRef]
20. Chen, S.; Chen, R.; Wang, G.G.; Gao, J.; Sangaiah, A.K. An adaptive large neighborhood search heuristic for dynamic vehicle routing problems. *Comput. Electr. Eng.* **2018**. [CrossRef]

21. Duan, H.; Zhao, W.; Wang, G.; Feng, X. Test-sheet composition using analytic hierarchy process and hybrid metaheuristic algorithm TS/BBO. *Math. Probl. Eng.* **2012**, *2012*, 1–22. [CrossRef]
22. Cui, Z.; Xue, F.; Cai, X.; Cao, Y.; Wang, G.G.; Chen, J. Malicious code variants detection based on deep learning technique. *IEEE Trans. Ind. Inform.* **2018**, 1–18. [CrossRef]
23. Rizk-Allah, R.M.; El-Sehiemy, R.A.; Wang, G.G. A novel parallel hurricane optimization algorithm for secure emission/economic load dispatch solution. *Appl. Soft Comput.* **2018**, *63*, 206–222. [CrossRef]
24. Zou, D.; Li, S.; Wang, G.G.; Li, Z.; Ouyang, H. An improved differential evolution algorithm for the economic load dispatch problems with or without valve-point effects. *Appl. Energy* **2016**, *181*, 375–390. [CrossRef]
25. Zou, D.X.; Deb, S.; Wang, G.G. Solving IIR system identification by a variant of particle swarm optimization. *Neural Comput. Appl.* **2016**. [CrossRef]
26. Nan, X.; Bao, L.; Zhao, X.; Zhao, X.; Sangaiah, A.K.; Wang, G.G.; Ma, Z. EPuL: An enhanced positive-unlabeled learning algorithm for the prediction of pupylation sites. *Molecules* **2017**, *22*, 1463. [CrossRef] [PubMed]
27. Wang, G.; Guo, L.; Duan, H. Wavelet neural network using multiple wavelet functions in target threat assessment. *Sci. World J.* **2013**, *2013*, 1–7. [CrossRef] [PubMed]
28. Wang, G.G.; Guo, L.; Duan, H.; Liu, L.; Wang, H. The model and algorithm for the target threat assessment based on Elman_AdaBoost strong predictor. *Acta Electron. Sin.* **2012**, *40*, 901–906. [CrossRef]
29. Srikanth, K.; Panwar, L.K.; Panigrahi, B.K.; Herrera-Viedma, E.; Sangaiah, A.K.; Wang, G.G. Meta-heuristic framework: Quantum inspired binary grey wolf optimizer for unit commitment problem. *Comput. Electr. Eng.* **2017**. [CrossRef]
30. Wang, G.; Guo, L.; Duan, H.; Liu, L.; Wang, H.; Shao, M. Path planning for uninhabited combat aerial vehicle using hybrid meta-heuristic DE/BBO algorithm. *Adv. Sci. Eng. Med.* **2012**, *4*, 550–564. [CrossRef]
31. Wang, G.; Guo, L.; Duan, H.; Liu, L.; Wang, H. A modified firefly algorithm for UCAV path planning. *Int. J. Hybrid Inf. Technol.* **2012**, *5*, 123–144.
32. Wang, G.; Guo, L.; Duan, H.; Liu, L.; Wang, H.; Wang, J. A hybrid meta-heuristic DE/CS algorithm for UCAV path planning. *J. Inf. Comput. Sci.* **2012**, *9*, 4811–4818.
33. Cui, Z.; Fan, S.; Zeng, J.; Shi, Z. APOA with parabola model for directing orbits of chaotic systems. *Int. J. Bio-Inspir. Comput.* **2013**, *5*, 67–72. [CrossRef]
34. Zhang, J.W.; Wang, G.G. Image matching using a bat algorithm with mutation. *Appl. Mech. Mater.* **2012**, *203*, 88–93. [CrossRef]
35. Zou, D.; Liu, H.; Gao, L.; Li, S. An improved differential evolution algorithm for the task assignment problem. *Eng. Appl. Artif. Intell.* **2011**, *24*, 616–624. [CrossRef]
36. Zou, D.X.; Wang, G.G.; Pan, G.; Qi, H. A modified simulated annealing algorithm and an excessive area model for the floorplanning with fixed-outline constraints. *Front. Inf. Technol. Electr. Eng.* **2016**, *17*, 1228–1244. [CrossRef]
37. Zou, D.X.; Wang, G.G.; Sangaiah, A.K.; Kong, X. A memory-based simulated annealing algorithm and a new auxiliary function for the fixed-outline floorplanning with soft blocks. *J. Ambient Intell. Humaniz. Comput.* **2017**. [CrossRef]
38. Li, Z.Y.; Yi, J.H.; Wang, G.G. A new swarm intelligence approach for clustering based on krill herd with elitism strategy. *Algorithms* **2015**, *8*, 951–964. [CrossRef]
39. Gao, X.Z.; Wang, X.; Jokinen, T.; Ovaska, S.J.; Arkkio, A.; Zenger, K. A hybrid PBIL-based harmony search method. *Neural Comput. Appl.* **2012**, *21*, 1071–1083. [CrossRef]
40. Zou, D.; Gao, L.; Wu, J.; Li, S.; Li, Y. A novel global harmony search algorithm for reliability problems. *Comput. Ind. Eng.* **2010**, *58*, 307–316. [CrossRef]
41. Zou, D.; Gao, L.; Li, S.; Wu, J. Solving 0-1 knapsack problem by a novel global harmony search algorithm. *Appl. Soft Comput.* **2011**, *11*, 1556–1564. [CrossRef]
42. Feng, Y.; Wang, G.G.; Gao, X.Z. A novel hybrid cuckoo search algorithm with global harmony search for 0-1 Knapsack problems. *Int. J. Comput. Intell. Syst.* **2016**, *9*, 1174–1190. [CrossRef]
43. Feng, Y.; Wang, G.G.; Wang, L. Solving randomized time-varying knapsack problems by a novel global firefly algorithm. *Eng. Comput.* **2017**. [CrossRef]
44. Yi, J.H.; Wang, J.; Wang, G.G. Improved probabilistic neural networks with self-adaptive strategies for transformer fault diagnosis problem. *Adv. Mech. Eng.* **2016**, *8*, 1–13. [CrossRef]
45. Dorigo, M.; Maniezzo, V.; Colorni, A. Ant system: optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **1996**, *26*, 29–41. [CrossRef] [PubMed]

46. Rajput, U.; Kumari, M. Mobile robot path planning with modified ant colony optimisation. *Int. J. Bio-Inspir. Comput.* **2017**, *9*, 106–113. [CrossRef]
47. Gonzalez-Pardo, A.; Ser, J.D.; Camacho, D. Solving strategy board games using a CSP-based ACO approach. *Int. J. Bio-Inspir. Comput.* **2017**, *10*, 136–144. [CrossRef]
48. Geem, Z.; Kim, J.; Loganathan, G. A new heuristic optimization algorithm: harmony search. *Simulation* **2001**, *76*, 60–68. [CrossRef]
49. Wang, G.; Guo, L.; Wang, H.; Duan, H.; Liu, L.; Li, J. Incorporating mutation scheme into krill herd algorithm for global numerical optimization. *Neural Comput. Appl.* **2014**, *24*, 853–871. [CrossRef]
50. Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *J. Glob. Optim.* **2007**, *39*, 459–471. [CrossRef]
51. Wang, H.; Yi, J.H. An improved optimization method based on krill herd and artificial bee colony with information exchange. *Memet. Comput.* **2017**. [CrossRef]
52. Yaghoobi, T.; Esmaeili, E. An improved artificial bee colony algorithm for global numerical optimisation. *Int. J. Bio-Inspir. Comput.* **2017**, *9*, 251–258. [CrossRef]
53. Sulaiman, N.; Mohamad-Saleh, J.; Abro, A.G. Robust variant of artificial bee colony (JA-ABC4b) algorithm. *Int. J. Bio-Inspir. Comput.* **2017**, *10*, 99–108. [CrossRef]
54. Liu, F.; Sun, Y.; Wang, G.G.; Wu, T. An artificial bee colony algorithm based on dynamic penalty and chaos search for constrained optimization problems. *Arab. J. Sci. Eng.* **2017**. [CrossRef]
55. Yang, X.S.; Deb, S. Cuckoo search via Lévy flights. In Proceeding of the World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), Coimbatore, India, 9–11 December 2009; Abraham, A., Carvalho, A., Herrera, F., Pai, V., Eds.; IEEE Publications: Piscataway, NJ, USA, 2009; pp. 210–214,
56. Wang, G.G.; Deb, S.; Gandomi, A.H.; Zhang, Z.; Alavi, A.H. Chaotic cuckoo search. *Soft Comput.* **2016**, *20*, 3349–3362. [CrossRef]
57. Wang, G.G.; Gandomi, A.H.; Zhao, X.; Chu, H.E. Hybridizing harmony search algorithm with cuckoo search for global numerical optimization. *Soft Comput.* **2016**, *20*, 273–285. [CrossRef]
58. Wang, G.G.; Gandomi, A.H.; Yang, X.S.; Alavi, A.H. A new hybrid method based on krill herd and cuckoo search for global optimization tasks. *Int. J. Bio-Inspir. Comput.* **2016**, *8*, 286–299. [CrossRef]
59. Yang, X.; Deb, S. Engineering optimisation by cuckoo search. *Int. J. Math. Model. Numer. Optim.* **2010**, *1*, 330–343. [CrossRef]
60. Cui, Z.; Sun, B.; Wang, G.G.; Xue, Y.; Chen, J. A novel oriented cuckoo search algorithm to improve DV-Hop performance for cyber-physical systems. *J. Parallel Distrib. Comput.* **2017**, *103*, 42–52. [CrossRef]
61. Kumaresan, T.; Palanisamy, C. E-mail spam classification using S-cuckoo search and support vector machine. *Int. J. Bio-Inspir. Comput.* **2017**, *9*, 142–156. [CrossRef]
62. Tan, Y. *Fireworks Algorithm—A Novel Swarm Intelligence Optimization Method*; Springer: Berlin/Heidelberg, Germany, 2015.
63. Yang, X.S. *Nature-Inspired Metaheuristic Algorithms*, 2nd ed.; Luniver Press: Frome, UK, 2010.
64. Wang, G.; Guo, L. A novel hybrid bat algorithm with harmony search for global numerical optimization. *J. Appl. Math.* **2013**, *2013*, 1–21. [CrossRef]
65. Xue, F.; Cai, Y.; Cao, Y.; Cui, Z.; Li, F. Optimal parameter settings for bat algorithm. *Int. J. Bio-Inspir. Comput.* **2015**, *7*, 125–128. [CrossRef]
66. Wang, G.G.; Chu, H.E.; Mirjalili, S. Three-dimensional path planning for UCAV using an improved bat algorithm. *Aerosp. Sci. Technol.* **2016**, *49*, 231–238. [CrossRef]
67. Gandomi, A.H.; Yang, X.S.; Alavi, A.H.; Talatahari, S. Bat algorithm for constrained optimization tasks. *Neural Comput. Appl.* **2013**, *22*, 1239–1255. [CrossRef]
68. Pan, W.T. A new Fruit Fly Optimization Algorithm: Taking the financial distress model as an example. *Knowl. Based Syst.* **2012**, *26*, 69–74. [CrossRef]
69. Wang, G.G.; Deb, S.; Coelho, L.S. Earthworm optimization algorithm: a bio-inspired metaheuristic algorithm for global optimization problems. *Int. J. Bio-Inspir. Comput.* **2015**. [CrossRef]
70. Wang, G.G.; Deb, S.; Coelho, L.S. Elephant herding optimization. In Proceedings of the 2015 3rd International Symposium on Computational and Business Intelligence (ISCBI 2015), Bali, Indonesia, 7–9 December 2015; pp. 1–5.
71. Wang, G.G.; Deb, S.; Gao, X.Z.; Coelho, L.S. A new metaheuristic optimization algorithm motivated by elephant herding behavior. *Int. J. Bio-Inspir. Comput.* **2016**, *8*, 394–409. [CrossRef]

72. Meena, N.K.; Parashar, S.; Swarnkar, A.; Gupta, N.; Niazi, K.R. Improved elephant herding optimization for multiobjective DER accommodation in distribution systems. *IEEE Trans. Ind. Inform.* **2017**. [CrossRef]
73. Wang, G.G. Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems. *Memet. Comput.* **2016**. [CrossRef]
74. Feng, Y.; Wang, G.G. Binary moth search algorithm for discounted 0-1 knapsack problem. *IEEE Access* **2018**, *6*, 10708–10719. [CrossRef]
75. Simon, D. Biogeography-based optimization. *IEEE Trans. Evol. Comput.* **2008**, *12*, 702–713. [CrossRef]
76. Wang, G.; Guo, L.; Duan, H.; Wang, H.; Liu, L.; Shao, M. Hybridizing harmony search with biogeography based optimization for global numerical optimization. *J. Comput. Theor. Nanosci.* **2013**, *10*, 2318–2328. [CrossRef]
77. Wang, G.G.; Gandomi, A.H.; Alavi, A.H. An effective krill herd algorithm with migration operator in biogeography-based optimization. *Appl. Math. Model.* **2014**, *38*, 2454–2462. [CrossRef]
78. Yang, X.S. Firefly algorithm, stochastic test functions and design optimisation. *Int. J. Bio-Inspir. Comput.* **2010**, *2*, 78–84. [CrossRef]
79. Wang, G.G.; Guo, L.; Duan, H.; Wang, H. A new improved firefly algorithm for global numerical optimization. *J. Comput. Theor. Nanosci.* **2014**, *11*, 477–485. [CrossRef]
80. Guo, L.; Wang, G.G.; Wang, H.; Wang, D. An effective hybrid firefly algorithm with harmony search for global numerical optimization. *Sci. World J.* **2013**, *2013*, 1–10. [CrossRef] [PubMed]
81. Gandomi, A.H.; Alavi, A.H. Krill herd: a new bio-inspired optimization algorithm. *Commun. Nonlinear Sci. Numer. Simul.* **2012**, *17*, 4831–4845. [CrossRef]
82. Wang, G.G.; Gandomi, A.H.; Alavi, A.H. A chaotic particle-swarm krill herd algorithm for global numerical optimization. *Kybernetes* **2013**, *42*, 962–978. [CrossRef]
83. Wang, G.G.; Gandomi, A.H.; Alavi, A.H.; Hao, G.S. Hybrid krill herd algorithm with differential evolution for global numerical optimization. *Neural Comput. Appl.* **2014**, *25*, 297–308. [CrossRef]
84. Wang, G.G.; Guo, L.; Gandomi, A.H.; Hao, G.S.; Wang, H. Chaotic krill herd algorithm. *Inf. Sci.* **2014**, *274*, 17–34. [CrossRef]
85. Wang, G.G.; Gandomi, A.H.; Alavi, A.H.; Deb, S. A multi-stage krill herd algorithm for global numerical optimization. *Int. J. Artif. Intell. Tools* **2016**, *25*, 1550030. [CrossRef]
86. Guo, L.; Wang, G.G.; H. Gandomi, A.; H. Alavi, A.; Duan, H. A new improved krill herd algorithm for global numerical optimization. *Neurocomputing* **2014**, *138*, 392–402. [CrossRef]
87. Wang, G.G.; Gandomi, A.H.; Alavi, A.H.; Gong, D. A comprehensive review of krill herd algorithm: Variants, hybrids and applications. *Artif. Intell. Rev.* **2017**. [CrossRef]
88. Wang, G.G.; Deb, S.; Cui, Z. Monarch butterfly optimization. *Neural Comput. Appl.* **2015**, 1–20. [CrossRef]
89. Kaedi, M. Fractal-based algorithm: A new metaheuristic method for continuous optimization. *Int. J. Artif. Intell.* **2017**, *15*, 76–92.
90. Shams, M.; Rashedi, E.; Dashti, S.M.; Hakimi, A. Ideal gas optimization algorithm. *Int. J. Artif. Intell.* **2017**, *15*, 116–130.
91. Precup, R.E.; Sabau, M.C.; Petriu, E.M. Nature-inspired optimal tuning of input membership functions of Takagi-Sugeno-Kang fuzzy models for Anti-lock Braking Systems. *Appl. Soft Comput.* **2015**, *27*, 575–589. [CrossRef]
92. Baruah, R.D.; Angelov, P. DEC: Dynamically evolving clustering and its application to structure identification of evolving fuzzy models. *IEEE Trans. Cybern.* **2014**, *44*, 1619–1631. [CrossRef] [PubMed]
93. Yi, J.H.; Lu, M.; Zhao, X.J. Quantum inspired monarch butterfly optimization for UCAV path planning navigation problem. *Int. J. Bio-Inspir. Comput.* **2017**, in press.
94. Ghetas, M.; Yong, C.H.; Sumari, P. Harmony-based monarch butterfly optimization algorithm. In Proceedings of the 2015 IEEE International Conference on Control System, Computing and Engineering (ICCSCE), George Town, Malaysia, 27–29 November 2015; pp. 156–161.
95. Feng, Y.; Wang, G.G.; Deb, S.; Lu, M.; Zhao, X. Solving 0-1 knapsack problem by a novel binary monarch butterfly optimization. *Neural Comput. Appl.* **2017**, *28*, 1619–1634. [CrossRef]
96. Wang, G.G.; Zhao, X.; Deb, S. A novel monarch butterfly optimization with greedy strategy and self-adaptive crossover operator. In Proceedings of the 2015 2nd International Conference on Soft Computing & Machine Intelligence (ISCMI 2015), Hong Kong, China, 23–24 November 2015; pp. 45–50.

97. Wang, G.G.; Deb, S.; Zhao, X.; Cui, Z. A new monarch butterfly optimization with an improved crossover operator. *Oper. Res. Int. J.* **2016**. [CrossRef]
98. Feng, Y.; Yang, J.; Wu, C.; Lu, M.; Zhao, X.J. Solving 0-1 knapsack problems by chaotic monarch butterfly optimization algorithm. *Memet. Comput.* **2016**. [CrossRef]
99. Ghanem, W.A.H.M.; Jantan, A. Hybridizing artificial bee colony with monarch butterfly optimization for numerical optimization problems. *Neural Comput. Appl.* **2016**. [CrossRef]
100. Wang, G.G.; Hao, G.S.; Cheng, S.; Qin, Q. A discrete monarch butterfly optimization for Chinese TSP problem. In *Advances in Swarm Intelligence, Proceedings of the 7th International Conference, ICSI 2016, Bali, Indonesia, 25–30 June 2016*; Tan, Y., Shi, Y., Niu, B., Eds.; Springer International Publishing: Cham, Switzerland, 2016; Volume 9712, pp. 165–173.
101. Feng, Y.; Wang, G.G.; Li, W.; Li, N. Multi-strategy monarch butterfly optimization algorithm for discounted 0-1 knapsack problem. *Neural Comput. Appl.* **2017**. [CrossRef]
102. Feng, Y.; Yang, J.; He, Y.; Wang, G.G. Monarch butterfly optimization algorithm with differential evolution for the discounted 0-1 knapsack problem. *Acta Electron. Sin.* **2017**, *45*.
103. Wang, G.G.; Hao, G.S.; Cheng, S.; Cui, Z. An improved monarch butterfly optimization with equal partition and F/T mutation. In Proceedings of the 8th International Conference on Swarm Intelligence (ICSI 2017), Fukuoka, Japan, 27 July 2017–1 August 2017.
104. Feng, Y.; Wang, G.G.; Dong, J.; Wang, L. Opposition-based learning monarch butterfly optimization with Gaussian perturbation for large-scale 0-1 knapsack problem. *Comput. Electr. Eng.* **2017**. [CrossRef]
105. Wang, G.G.; Deb, S.; Gandomi, A.H.; Alavi, A.H. Opposition-based krill herd algorithm with Cauchy mutation and position clamping. *Neurocomputing* **2016**, *177*, 147–157. [CrossRef]
106. Mahata, S.; Saha, S.K.; Kar, R.; Mandal, D. Enhanced colliding bodies optimisation-based optimal design of wideband digital integrators and differentiators. *Int. J. Bio-Inspir. Comput.* **2017**, *9*, 165–181. [CrossRef]
107. Chen, S.; Chen, R.; Gao, J. A monarch butterfly optimization for the dynamic vehicle routing problem. *Algorithms* **2017**, *10*, 107. [CrossRef]
108. Meng, L.; Wang, Y.; Huang, H. Improved monarch butterfly optimization by using strategy of dynamic-dividing population. *Comput. Eng. Appl.* **2017**, *53*, 149–156.
109. Faris, H.; Aljarah, I.; Mirjalili, S. Improved monarch butterfly optimization for unconstrained global search and neural network training. *Appl. Intell.* **2018**, *48*, 445–464. [CrossRef]
110. Ehteram, M.; Karami, H.; Mousavi, S.F.; Farzin, S.; Kisi, O. Optimization of energy management and conversion in the multi-reservoir systems based on evolutionary algorithms. *J. Clean. Prod.* **2017**, *168*, 1132–1142. [CrossRef]
111. Wang, G.G.; Gandomi, A.H.; Alavi, A.H. Stud krill herd algorithm. *Neurocomputing* **2014**, *128*, 363–370. [CrossRef]

© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).