applied sciences

MDPI

Article

Adaptive-Uniform-Experimental-Design-Based Fractional-Order Particle Swarm Optimizer with Non-Linear Time-Varying Evolution

Po-Yuan Yang [1,2], Fu-I Chou [3], Jinn-Tsong Tsai [4,*] and Jyh-Horng Chou [2,5,6,*]

1 Department of Information Engineering and Computer Science, Feng Chia University, 100, Wen-Hwa Road, Taichung 407, Taiwan; 1103404102@nkust.edu.tw
2 Department of Electrical Engineering, National Kaohsiung University of Science and Technology, 415 Chien-Kung Road, Kaohsiung 807, Taiwan
3 Department of Automation Engineering, National Formosa University, 64, Wun-Hua Road, Yunlin 632, Taiwan; alvis.cfi@gmail.com
4 Department of Computer Science, National Pingtung University, 4-18 Min-Sheng Road, Pingtung 900, Taiwan
5 Department of Mechanical Engineering, National Chung-Hsing University, 145 Xing-Da Road, Taichung 402, Taiwan
6 Department of Healthcare Administration and Medical Informatics, Kaohsiung Medical University, 100 Shi-Quan 1st Road, Kaohsiung 807, Taiwan
* Correspondence: jttsai@mail.nptu.edu.tw (J.-T.T.); choujh@nkust.edu.tw (J.-H.C.)

Received: 22 November 2019; Accepted: 12 December 2019; Published: 16 December 2019

Abstract: An adaptive-uniform-experimental-design-based fractional particle swarm optimizer (AUFPSO) with non-linear time-varying evolution (NTE) is proposed. A particle swarm optimizer (PSO) is an excellent evolutionary algorithm due to its simple structure and rapid convergence. Nevertheless, PSO has notable drawbacks. Although many proposed methods and strategies have enhanced its effectiveness and performance, PSO is limited by its tendency to fall into local optima and its tendency to obtain different solutions in each search (i.e., its weak robustness). Introducing fractional-order calculus in PSO (FPSO) can correct the order of the velocity derivative for each particle, which enhances the diversity and algorithmic effectiveness. This study used NTE of the order of the velocity derivative, inertia weight, cognitive parameter, and social parameter in an FPSO used to search for a global optimal solution. To obtain the best combination of FPSO and NTE, an adaptive uniform experimental design (AUED) method was used to deal with this essential issue. The AUED method integrates a uniform layout with the best combination phase and a stepwise ratio to assist in selecting the best combination for FPSO-NTE. Experimental applications in 15 global numerical optimization problems confirmed that the AUFPSO-NTE had a better performance and robustness than existing PSO-related algorithms.

Keywords: uniform experimental design; fractional-order particle swarm optimizer; non-linear time-varying evolution; parameter optimization

1. Introduction

Particle swarm optimization (PSO), which was first proposed in 1995 [1], is a swarm intelligence computational technique inspired by animal behavior, such as birds' flocking. Because of its many advantages, including fast convergence, simple structure, and high accuracy, PSO is used to solve optimization problems, e.g., the travelling salesman problem [2], and problems in many industrial and engineering domains [3], e.g., image processing [4], clouding computing [5], power systems [6,7], the

chemical composition of a steel bar [8], and aircraft landing systems [9]. Although PSO is an excellent optimization algorithm, it still has drawbacks that must be addressed. Shi and Eberhart proposed an inertia weight, a linear time-varying inertia weight, and a random inertia weight to enhance the search ability [10–12]. Ratnaweera et al. [13] used a time-varying acceleration coefficient, mutation, and a self-organizing hierarchy to prevent the premature convergence in PSO. Chatterjee and Siarry [14] used a non-linear time-varying inertia weight to enhance convergence. Yang et al. [15] proposed an improved approach for velocity and afforded a different inertia weight for each particle. Ko et al. [16] applied the concept of a non-linear time-varying inertia weight, cognitive coefficient, and social coefficient. Ali and Kaelo [17] and Chen and Li [18] proposed strategies for adjusting the self-learning strategy for each particle in a PSO. Huang [19] and Li [20] focused on improving the topology to improve the PSO performance. Tsai et al. [21] divided particles into several groups and allowed particles to be shared among different groups. Chen et al. [22] proposed a method of aging leaders and challengers in PSO to overcome prematurity. Pehlivanoglu [23] proposed a periodic mutation strategy to increase diversity and to avoid falling into the local optimum. Li et al. [24] employed a new weighted particle to improve the PSO. Wang et al. [25] developed a dual-factor strategy for increasing diversity. Cheng and Jin [26] improved the learning model of particles to increase the search effectiveness. Lynn and Suganthan [27] divided the swarm population into two subpopulations to explore or exploit exemplars by using the personal best experiences of the particles or the whole swarm, respectively. Tsai et al. [28] used a sliding-level Taguchi method to enhance the PSO performance. Many other optimization algorithms have been used to increase PSO effectiveness and performance [29–32]. Although the above improved PSO algorithms do improve the PSO performance, common limitations are their lack of robustness and tendency to fall into the local optimum. Moreover, all the above PSO-based algorithms are based on an integer-order derivative, which cannot be used to describe natural phenomena, because fractional-order derivatives have a great influence in nature for objects that are observed, touched, and controlled by humans [33]. Therefore, some researchers have proposed the use of a fractional-order derivative to improve the PSO performance. A fractional-order particle swarm optimizer (FPSO) was first proposed by Solteiro Pires et al. [34]. Solteiro Pires and coworkers [34,35] later introduced the Grünwald–Letnikov definition of the fractional-order derivative in PSO and modified the velocity update approach. In FPSO, memories of past particle movements influence the next flight. Gao et al. [36], Couceiro and Ghamisi [37], Guo et al. [38], and Hosseini et al. [39] also applied a fraction-order (FO) in PSO and Darwinian PSO (DPSO). However, the most recent studies in FPSO have talked about the applications, such as the direction of arrival estimation for electromagnetic plane waves [40], fractional order filters design [41], fractional fixed-structure H_∞ controller design [42], image segmentation [43–46], image border detection [47], design of a complementary metal-oxide-semiconductor (CMOS) power amplifier [39], design for an electric power transmission system [48], optimization for a pressurized water reactor (PWR) core loading pattern [49], and optimization of extreme learning machine assignments [50]. This paper proposes an FPSO algorithm with a non-linear time-varying evolution (NTE) based on Ko et al. [16] and Solteiro Pires et al. [34]. In the proposed FPSO-NTE algorithm, NTE was used to set the inertia weight, cognitive coefficient, and social coefficient. The proposed FPSO-NTE algorithm used four constant coefficients to influence the order of the velocity derivative, the inertia weight, the cognitive parameter, and the social parameter.

Therefore, an important issue is how to determine the best combination of the four constant coefficients. The many proposed methods for determining the best combination include the trial-and-error method, one-factor-at-a-time experimental design, full-factorial experimental design, Taguchi method [51], and uniform experimental design [52,53]. Although most proposed methods and experimental designs are systematic, they all have limitations. To solve this problem, Tsai et al. [54] proposed a data-driven approach to uniform experimental design (DAUED) for optimizing the parameters of an auto-alignment machine. The DAUED method optimizes parameters by integrating a ten-level uniform layout with the best combination stage and a stepwise ratio. The FPSO-NTE

algorithm proposed in this study applied the underlying concept of the DAUED method by using DAUED to select the best combination of four constant coefficients. Because a data-driven approach was not used, DAUED was renamed to an adaptive uniform experimental design (AUED) method for this study. The AUFPSO-NTE algorithm automatically obtained the best combination of four constant coefficients because AUED was integrated in FPSO-NTE.

This paper is organized as follows. FPSO and the proposed FPSO-NTE are briefly described in Section 2. Section 3 describes how AUED was used in FPSO-NTE to search for the best combination. Section 4 presents and discusses the experimental and simulation results. Finally, Section 5 concludes the study.

2. FPSO and FPSO-NTE

Solteiro Pires and coworkers [34,35] first proposed FPSO, which introduced a fractional-order derivative in PSO for rearranging and modifying the order of the velocity derivative. In a fractional-order derivative, common definitions include the Riemann–Liouville definition, the Caputo definition, and the Grünwald–Letnikov definition [55–57]. In the FPSO proposed by Solteiro Pires and coworkers [34,35], the order of the velocity derivative is derived from the Grünwald–Letnikov definition, as shown below:

$$D^{\lambda}[x(t)] = \lim_{h \to 0}\left[\frac{1}{h^{\lambda}}\sum_{k=0}^{+\infty}\frac{(-1)^{k}\Gamma(\lambda+1)x(t-kh)}{\Gamma(k+1)\Gamma(\lambda-k+1)}\right], \tag{1}$$

where D is the derivative operator, λ is the fractional order of the derivative, h is the difference operator, Γ is the Gamma function, and $x(t)$ is the object for the derivative.

For an application in discrete time, Expression (1) can be approximated using:

$$D^{\lambda}[x(t)] = \frac{1}{T^{\lambda}}\sum_{k=0}^{r}\frac{(-1)^{k}\Gamma(\lambda+1)x(t-kT)}{\Gamma(k+1)\Gamma(\lambda-k+1)}, \tag{2}$$

where T and r are the sampling period and the truncation order, respectively.

The update expression of the velocity for PSO is:

$$V_i(t+1) = V_i(t) + c_1 \cdot r_1\left(P_i^{l}(t) - p_i(t)\right) + c_2 \cdot r_2(P^{g}(t) - p_i(t)). \tag{3}$$

The update expression of the position for PSO is:

$$P_i(t+1) = p_i(t) + V_i(t+1), \tag{4}$$

where $i = 1, 2, \ldots, S$; S is the number of particles; t is the current iteration; λ is the fractional order of the derivative; $V_i(t)$ is the velocity; c_1 and c_2 are the cognitive and social coefficients, respectively; $p_i(t)$ is the current position; ${P_i}^{l}(t)$ and $P^{g}(t)$ are the position of the self-best solution in the current iteration and the position of the global solution in the current iteration, respectively; and r_1 and r_2 are random constants between 0 and 1.

In Expression (3), the position term p_i of $({P_i}^{l} - p_i)$ is rewritten as follows using fractional-order derivation:

$$\begin{aligned} D^{\lambda}[p_i(t+1)] = & \sum_{k=0}^{r}\frac{(-1)^{k}\Gamma(\lambda+1)x(t-k)}{\Gamma(k+1)\Gamma(\lambda-k+1)} \\ & = \lambda p_i(t) + \tfrac{\lambda}{2}(1-\lambda)p_i(t-1) + \tfrac{\lambda}{6}(1-\lambda)(2-\lambda)p_i(t-2) \\ & + \tfrac{\lambda}{24}(1-\lambda)(2-\lambda)(3-\lambda)p_i(t-3). \end{aligned} \tag{5}$$

According to Solteiro Pires et al. [34] and Gao et al. [36], r is set to 4 to obtain the best balance of convergence rate and accuracy. Additionally, tests show that an r larger than 4 obtains the same results.

Therefore, the updated expression of the velocity of the FPSO is:

$$V_i(t+1) = V_i(t) + c_1 \cdot r_1 \Big[P_i^l(t) - \lambda p_i(t) - \tfrac{\lambda}{2}(1-\lambda) p_i(t-1) - \tfrac{\lambda}{6}(1-\lambda)(2-\lambda) \\ p_i(t-2) - \tfrac{\lambda}{24}(1-\lambda)(2-\lambda)(3-\lambda) p_i(t-3) \Big] + c_2 \cdot r_2 (P^g(t) - p_i(t)). \tag{6}$$

To improve the PSO performance and effectiveness, Ko et al. [16] and Cui and Zeng [58] introduced a non-linear NTE in PSO. Gao et al. [36] introduced a non-linear time-varying inertia weight in FO-DPSO. In the current study, NTE is integrated in FPSO to enhance the performance and effectiveness. The resulting FPSO-NTE is expressed as follows:

$$\begin{aligned} &V_i(t+1) = \omega(t) \cdot V_i(t) \\ &+ c_1(t) \cdot r_1 \Big[P_i^l(t) - \lambda p_i(t) - \tfrac{\lambda}{2}(1-\lambda) p_i(t-1) - \tfrac{\lambda}{6}(1-\lambda)(2-\lambda) p_i(t-2) \\ &- \tfrac{\lambda}{24}(1-\lambda)(2-\lambda)(3-\lambda) p_i(t-3) \Big] + c_2(t) \cdot r_2 (P^g(t) - p_i(t)) \end{aligned} \tag{7}$$

$$p_i(t+1) = p_i(t) + V_i(t+1), \tag{8}$$

where $\omega(t)$ is the time-varying inertia weight, $\omega_{\min} \le \omega(t) \le \omega_{\max}$, $c_1(t)$ is the time-varying cognitive coefficient, $c_{1\min} \le c_1(t) \le c_{1\max}$, $c_2(t)$ is the time-varying social coefficient, and $c_{2\min} \le c_2(t) \le c_{2\max}$.

$$\omega(t) = \omega_{min} + \left(\frac{t_{max} - t}{t_{max}} \right)^{\alpha} \times (\omega_{max} - \omega_{min}), \tag{9}$$

$$c_1(t) = c_{1min} + \left(\frac{t_{max} - t}{t_{max}} \right)^{\beta} \times (c_{1max} - c_{1min}), \tag{10}$$

$$c_2(t) = c_{2max} + \left(\frac{t_{max} - t}{t_{max}} \right)^{\gamma} \times (c_{2min} - c_{2max}),, \tag{11}$$

where t and t_{max} are the current iteration and the maximum of iteration, respectively.

The constant coefficient λ influences the algorithmic performance, and coefficients α, β, and γ influence ω, c_1, and c_2, respectively. The problem is how to obtain the best parameter value for the constant coefficients λ, α, β, and γ. The solution proposed in this study is to use AUED to obtain the best combination of constant coefficient values.

3. AUED-Based FPSO (AUFPSO) with NTE

This study used the AUED method in the proposed FPSO-NTE algorithm to assist in the search for the best combination of the four constant coefficients. The three main steps of AUED method are initializing, performing ten-level uniform layout experiments, and calculating the parameter range for the next ten-level uniform layout. The initialization step includes selection of the parameters to be optimized, their ranges, and the solution accuracy. At this time, a suitable ten-level uniform layout, the output, a stepwise ratio, and the stop condition are also selected. The second main step can then be executed. Ten levels of the parameter range and ten levels of the solution accuracy are defined, and the ten levels are assigned to the ten-level uniform layout. The ten-level uniform layout experiments can then be performed, and the experimental results are recorded. After the ten-level uniform layout experiments in this stage are completed, the best combination obtained according to the output is an optimal or near-optimal value. The range for each parameter is then calculated according to the best combination and the stepwise ratio. The second and third main steps are repeated until the stop condition is met.

For a clear understanding of how the AUED method is applied in the proposed FPSO-NTE algorithm, the detailed steps of the method are given below.

A. Initialization of the AUED method in the proposed FPSO-NTE algorithm

Step 1. Define the experimental parameters as the four constant coefficients λ, α, β, and γ. For each parameter, set the range from 0 to 2, and set the solution accuracy to 0.0001.
Step 2. Set the experimental output as the fitness value.
Step 3. Set the stepwise ratio to 0.8.
Step 4. Select a suitable ten-level uniform layout of $U_{10}(10^4)$, as shown in Table 1.
Step 5. Repeat steps 1–4 until the objective value is reached or until the fitness value does not obtain a near-objective value in two consecutive ten-level uniform layout experiments.

Table 1. A ten-level uniform layout of $U_{10}(10^4)$.

Experiment Number	Column Numbers			
	1	**2**	**5**	**7**
1	1	2	5	7
2	2	4	10	3
3	3	6	4	10
4	4	8	9	6
5	5	10	3	2
6	6	1	8	9
7	7	3	2	5
8	8	5	7	1
9	9	7	1	8
10	10	9	6	4

B. Perform the ten-level uniform layout experiments

Step 1. The ranges for each parameter are divided into ten discrete values according to the chosen ten-level uniform layout of $U_{10}(10^4)$.
Step 2. Assign ten discrete values of each parameter into the chosen ten-level uniform layout of $U_{10}(10^4)$, shown as Table 2.
Step 3. Perform this process 15 times for each ten-level uniform layout experiment and record the average as the output.

Table 2. Ten-level uniform layout of $U_{10}(10^4)$ for the proposed adaptive-uniform-experimental-design-based fractional particle swarm optimizer with non-linear time-varying evolution (AUFPSO-NTE).

Experiment Number	Experimental Parameters			
	λ	α	β	γ
1	0	0.2222	0.8889	1.3333
2	0.2222	0.6667	2	0.4444
3	0.4444	1.1111	0.6667	2
4	0.6667	1.5556	1.7778	1.1111
5	0.8889	2	0.4444	0.2222
6	1.1111	0	1.5556	1.7778
7	1.3333	0.4444	0.2222	0.8889
8	1.5556	0.8889	1.3333	0
9	1.7778	1.3333	0	1.5556
10	2	1.7778	1.1111	0.6667

C. Update the search range for next ten-level uniform experiments

Step 1. For each parameter, calculate the search range according to the best combination in this stage and the stepwise ratio (0.8). The updated Algorithm 1 is shown below.
Step 2. Return to main step B and execute the experimental steps until the stop condition is met.

Algorithm 1

```
Start
    For K = 1 to PARA_NO
        LT ← LB(K);
        UT ← UB(K);
        LB(K) ← BEST(K) − (UT − LT) × SWR ÷ 2;
        UB(K) ← BEST(K) + (UT − LT) × SWR ÷ 2;
        If LB(K) < LT
          LB(K) = LT;
        End
        If UB(K) > UT
          UB(K) = UT;
        End
        For I = 1 to EXP_NO
            LEVEL(I, K) ← LB(K) + ((UB(K) − LB(K))/(EXP_NO − 1) × (I − 1));
        End
    End
End
```

where *PARA_NO* is the total number of parameters; *LB* and *UB* are the upper and lower bounds for each experimental parameter, respectively; *LT* and *UT* are temporary values of *LB* and *UB*, respectively; *LEVEL* is the level value; *EXP_NO* is the total number of experiments in the uniform layout; *BEST* is the best parameter value for the best combination obtained by the uniform layout experiments in this stage; and *SWR* is a stepwise ratio.

The following example demonstrates the use of the updated algorithm when the upper and lower bounds for each parameter are initially set to 2 and 0, respectively, the number of parameters was 4, and the number of experiments was 10. The first ten-level uniform layout experiments indicated that the best combination [P_1 P_2 P_3 P_4] was [0.2553 0.4514 0.5556 1.2455]. The stepwise ratio was set to 0.8. Here, the first parameter value was used to explain how to calculate a new range for the next ten-level uniform layout experiments stage.

When the first ten-level uniform layout experiments stage was completed, *LB* and *UB* were 0 and 2, respectively. At first, *LT* and *UT* were 0 and 2, respectively, due to *LB* and *UB*. The best parameter was 0.2553, and the stepwise ratio was 0.8. Therefore, $LB = 0.2553 - (2 - 0) \times 0.8 \div 2 = -0.5447$ and $UB = 0.2553 - (2 - 0) \times 0.8 \div 2 = 1.0553$. However, since *LB* was lower than *LT*, *LB* must be corrected to *LT*, and the new range for the first parameter was 0 to 1.0553. This was because the original range was set to 0 to 2. Therefore, we could know that *LB* for the next uniform layout experiments must be equal to or more than *LT*. In the same way, *UB* for the next uniform layout experiments must have been equal to or less than *UT*. Next, the range was divided into ten levels, and a discrete value was calculated for each level. The discrete values were calculated as follows: $0 + ((1.0553 - 0) \div (10 - 1) \times (1 - 1)) = 0$ for the first level; $0 + ((1.0553 - 0) \div (10 - 1) \times (2 - 1)) = 0.1173$ for the second level, and so on. The discrete value for the tenth level was calculated as $0 + ((1.0553 - 0) \div (10 - 1) \times (10 - 1)) = 1.0533$. Table 3 shows the levels obtained after the updated algorithm was executed in this instance.

Table 3. New levels obtained after the updated algorithm was executed in this instance.

Levels	Parameters			
	*P*1	*P*2	*P*3	*P*4
1	0	0	0	0.4455
2	0.1173	0.1390	0.1506	0.6182
3	0.2345	0.2781	0.3012	0.7909
4	0.3518	0.4171	0.4519	0.9637

Table 3. *Cont.*

Levels	Parameters			
	***P*1**	***P*2**	***P*3**	***P*4**
5	0.4690	0.5562	0.6025	1.1364
6	0.5863	0.6952	0.7531	1.3091
7	0.7035	0.8343	0.9037	1.4818
8	0.8208	0.9733	1.0544	1.6546
9	0.9380	1.1124	1.2050	1.8273
10	1.0553	1.2514	1.3556	2

4. Simulation Results and Comparisons

This section presents the results for the 15 global numerical optimization problems in Table 4, which were used for the performance evaluations of the proposed AUFPSO-NTE algorithm. In Example (1), functions f_1–f_7 were used to compare the performance of the proposed AUFPSO-NTE, the PSO-FOV (PSO with the fractional-order velocity) proposed by Solteiro Pires et al. [34], the FPSO improved by Solteiro Pires et al. [35] from PSO-FOV, the modified PSO (MPSO) proposed by Shi and Eberhart [11], and the PSO proposed by Kennedy and Eberhart [1]. In Example (2), functions f_5 and f_8–f_{11} were used to compare the performance of the proposed AUFPSO-NTE, the FPSO-based algorithms proposed by Gao et al. [36], and the PSO proposed by Gao et al. [36]. In Example (3), functions f_5, f_8, f_{10}, f_{11}, and f_{12} were used to compare between the AUFPSO-NTE, the adaptive fractional-order Darwinian PSO (AFO-DPSO) proposed by Guo et al. [38], and the PSO-based algorithms described in Guo et al. [38]. In Example (4), functions f_5, f_8, f_{10}, f_{11}, and f_{13}–f_{15} were used to compare between the AUFPSO-NTE, the fractional-order Darwinian PSO (FDPSO) proposed by Hosseini et al. [39] and the PSO-based algorithms described in Hosseini et al. [39]. The simulations were run on a Windows 10 personal computer with a core i7-6700M, a 3.4 GHz CPU, and 8 GB RAM.

Table 4. Benchmark functions.

Name	Definition	Solution Space	Optimal Value
Bohachevsky 1	$f_1 = v_1^2 + 2v_2^2 - 0.3\cos(3\pi v_1) - 0.4\cos(4\pi v_2) + 0.7$	$[-50,50]^2$	0
Colville	$f_2 = 100\left(v_2 - v_1^2\right)^2 + (1 - v_1)^2 + 90\left(v_4 - v_3^2\right)^2 + (1 - v_3)^2 + 10.1\left[(v_2 - 1)^2 + (v_4 - 1)^2\right] + 19.8(v_2 - 1)(v_4 - 1)$	$[-10,10]^4$	0
Drop wave	$f_3 = -\frac{1+\cos\left(12\sqrt{v_1^2+v_2^2}\right)}{0.5\left(v_1^2+v_2^2\right)+2}$	$[-10,10]^2$	−1
Easom	$f_4 = -\cos(v_1)\cos(v_2)e^{-(v_1-\pi)^2-(v_2-\pi)^2}$	$[-100,100]^2$	−1
Rastrigin	$f_5 = \sum_{i=1}^{G}\left(v_i^2 - 10\cos(2\pi v_i) + 10\right)$	$[-5.12,5.12]^G$	0
Michalewiczs	$f_6 = \sum_{i=1}^{G} -\sin(v_i)\left[\sin\frac{(i+1)v_i^2}{\pi}\right]^{2t}$	$[0,\pi]^2$	−1.8409
Rosenbrock's valley	$f_7 = \sum_{i=1}^{G-1} 100\left(v_{i+1} - v_i^2\right)^2$	$[-2.048,2.048]^2$	0
Sphere	$f_8 = \sum_{i=1}^{G} v_i^2$	$[-100,100]^G$	0
Ackley	$f_9 = -20\exp\left(-0.2\sqrt{\frac{\sum_{i=1}^{G} v_i^2}{G}}\right) + \exp\left(\sqrt{\frac{\sum_{i=1}^{G}\cos 2\pi v_i}{G}}\right) + 20 + e$	$[-32,32]^G$	0
Rosenbrock	$f_{10} = \sum_{i=1}^{G-1}\left[100\left(v_{i+1} - v_i^2\right)^2 + (1 - v_i)^2\right]$	$[-30,30]^G$	0
Griewank	$f_{11} = \frac{1}{4000}\sum_{i=1}^{G} v_i^2 - \prod_{i=1}^{G}\cos\left(\frac{v_i}{\sqrt{i}}\right) + 1$	$[-600,600]^G$	0
DeJong F4	$f_{12} = \sum_{i=1}^{G} v_i^4$	$[-20,20]^G$	0
Schwefel's P1.2	$f_{13} = \sum_{i=1}^{G}\left(\sum_{j=1}^{i} v_j\right)^2$	$[-100,100]^G$	0

Table 4. *Cont.*

Name	Definition	Solution Space	Optimal Value
Quartic	$f_{14} = \sum_{i=1}^{G} iv_i^4 + random[0, 1)$	$[-1.28,1.28]^G$	0
Salomon	$f_{15} = 1 - \cos\left(2\pi\sqrt{\sum_{i=1}^{G} v_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{G} v_i^2}$	$[-100,100]^G$	0

4.1. Example (1): Proposed AUFPSO-NTE in Comparison with FPSO, PSO-FOV, MPSO, and PSO

The FPSO was the first developed, called PSO-FOV (PSO with the fractional-order velocity), and improved by Solteiro Pires and coworkers [34,35], the MPSO introduced a time-varying inertia weight [11], and PSO was first proposed by Kennedy and Eberhart [1]. All three algorithms were compared with the proposed AUFPSO-NTE.

Table 5 shows the number of dimensions (D_n) that functions f_1 to f_7 were set to with D_n = 2, 4, 2, 2, 30, 2, and 4, respectively; the number of particles (S) was set to S = 10; and the number of iterations (I) was set to 200. Table 6 shows the parameter settings for the proposed AUFPSO-NTE and for the FPSO, PSO-FOV, MPSO, and PSO. In the AUFPSO-NTE, the minimum weight ($\omega_{\min}$) and maximum weight ($\omega_{\max}$) were set to 0.4 and 0.9, respectively. The minimum cognitive coefficient ($c_{1\min}$) and maximum cognitive coefficient ($c_{1\max}$) were set to 0 and 2, respectively. The minimum social coefficient ($c_{2\min}$) and maximum social coefficient ($c_{2\max}$) were set to 0 and 2, respectively. In the MPSO, $\omega_{\min}$ and $\omega_{\max}$ were set to 0.4 and 0.9, respectively, and c_1 and c_2 were both set to 2. For FPSO, PSO-FOV and PSO, c_1 and c_2 were both set to 2. The maximum velocity ($V_{\max}$) was defined as the maximum position minus minimum position, and the minimum velocity ($V_{\min}$) was defined as negative $V_{\max}$. Table 7 shows the best combinations of the constant coefficients λ, α, β, and γ for each benchmark function of the proposed AUFPSO-NTE. For comparisons, the constant coefficient λ and results are obtained by the algorithms developed by Solteiro Pires and coworkers [34,35]. Tables 8 and 9 show the constant coefficient λ for each benchmark function of the FPSO [35] and PSO-FOV [34], respectively. In Tables 7–9, λ is the fractional order of the derivative. In Table 7, α, β, and γ are coefficients that influence ω, c_1, and c_2, respectively.

Table 5. Settings for number of dimensions, number of particles, and number of iterations in the proposed AUFPSO-NTE and for FPSO, PSO-FOV, MPSO, and PSO in Example (1).

	Function	Number of Dimension (D_n)	Number of Particles (S)	Number of Iterations (I)
f_1	Bohachevsky 1	2	10	200
f_2	Colville	4	10	200
f_3	Drop wave	2	10	200
f_4	Easom	2	10	200
f_5	Rastrigin	30	10	200
f_6	Michalewiczs	2	10	200
f_7	Rosenbrock's valley	4	10	200

Table 6. Parameter settings for the proposed AUFPSO-NTE and for FPSO, PSO-FOV, MPSO, and PSO in Example (1).

Terms		Algorithms				
		AUFPSO-NTE	FPSO	PSO-FOV	MPSO	PSO
ω	min max	0.4 0.9	N/A	N/A	0.4 0.9	N/A
c_1	min max	0 2	2	2	2	2

Table 6. *Cont.*

Terms		Algorithms				
		AUFPSO-NTE	**FPSO**	**PSO-FOV**	**MPSO**	**PSO**
c_2	min	0	2	2	2	2
	max	2				

Table 7. The best combinations of the constant coefficients for each benchmark function of the proposed AUFPSO-NTE in Example (1).

Function	λ	α	β	γ
Bohachevsky 1	0.6667	1.5556	1.7778	1.1111
Colville	0.8852	0.3010	1.2533	0.9829
Drop wave	0.0320	0.3562	0.5070	1.5989
Easom	1.0035	0.3911	1.5022	1.4322
Rastrigin	1.3333	0.4444	0.2222	0.8889
Michalewiczs	0.9867	0.2489	1.9140	1.7678
Rosenbrock's valley	1.3333	0.4444	0.2222	0.8889

Table 8. The λ parameter values for each benchmark function of FPSO in Example (1).

Function	λ
Bohachevsky 1	1.65
Colville	1.75
Drop wave	1.43
Easom	1.98
Rastrigin	1.99
Michalewiczs	1.99
Rosenbrock's valley	1.66

Table 9. The λ parameter values for each benchmark function of PSO-FOV in Example (1).

Function	λ
Bohachevsky 1	0.35
Colville	0.57
Drop wave	0.57
Easom	0.32
Rastrigin	0.54
Michalewiczs	0.19
Rosenbrock's valley	0.41

Table 10 shows the performance comparison results for the proposed AUFPSO-NTE and for FPSO [35], PSO-FOV [34], MPSO [11], and PSO [1]. The table shows the best solution, mean, and standard deviation (S.D.) that each algorithm obtained for f_1–f_7 in 30 independent trials. In Example (1), all algorithms except PSO obtained the best solutions for f_1, f_3, f_4, and f_6. Only the proposed AUFPSO-NTE and FPSO obtained the best solution for f_5 [35]. Additionally, only the proposed AUFPSO-NTE obtained the best solution for f_2 and f_7. In terms of the mean and S.D., the proposed AUFPSO-NTE outperformed FPSO [35], PSO-FOV [34], MPSO [11], and PSO [1].

Table 10. Superior results obtained by the proposed AUFPSO-NTE in comparison with FPSO, PSO-FOV, MPSO, and PSO in 30 independent trials in Example (1).

Function	Terms	AUFPSO-NTE	FPSO	PSO-FOV	MPSO	PSO
Bohachevsky 1	Best	0	0	0	0	0.1118
	Mean	0	5.6621×10^{-16}	0.0138	0.0275	5.1798
	S.D.	0	1.8519×10^{-15}	0.0754	0.1048	6.5905
Colville	Best	7.7920×10^{-5}	1.3908×10^{-3}	4.0173×10^{-3}	0.2002	11.7514
	Mean	1.6906	2.4786	2.8318	5.9281	223.7255
	S.D.	1.8018	2.0702	2.8068	9.8226	462.9368
Drop wave	Best	−1	−1	−1	−1	−0.9803
	Mean	−1	−0.9763	−0.9617	−0.9660	−0.7892
	S.D.	1.9387×10^{-11}	0.0310	0.0318	0.0324	0.1580
Easom	Best	−1	−1	−1	−1	−0.9103
	Mean	−1	−1	0.9998	−0.9998	−0.0721
	S.D.	7.6828×10^{-11}	6.8674×10^{-6}	9.0521×10^{-4}	1.0572×10^{-3}	0.1996
Rastrigin	Best	0	0	94.1343	0.0151	24.6608
	Mean	5.1159×10^{-14}	0.1044	153.9817	133.8380	223.4087
	S.D.	6.5665×10^{-14}	0.3025	27.1046	77.8643	75.7597
Michalewiczs	Best	−1.8409	−1.8409	−1.8409	−1.8409	−1.8388
	Mean	−1.8409	−1.8409	−1.8409	−1.8409	−1.8388
	S.D.	2.9272×10^{-10}	9.4381×10^{-8}	1.2377×10^{-7}	3.9510×10^{-7}	4.2741×10^{-7}
Rosenbrock's valley	Best	4.0000×10^{-34}	1.2791×10^{-21}	4.8507×10^{-16}	3.9505×10^{-14}	0.0177
	Mean	3.9129×10^{-22}	1.3965×10^{-4}	2.5900×10^{-3}	2.8691×10^{-3}	4.2239
	S.D.	1.6158×10^{-21}	6.4890×10^{-4}	0.0089	0.0157	7.7636

4.2. Example (2): Proposed AUFPSO-NTE in Comparison with FVFP-PSO, FP-PSO, FV-PSO, and PSO

Example (2) compared the performance of the proposed AUFPSO-NTE with the standard PSO and with three modifications of PSO proposed by Gao et al. [36]: particle swarm optimization with the fractional-order velocity and the fractional-order position (FVFP-PSO), particle swarm optimization with the fractional-order position (FP-PSO), and particle swarm optimization with the fractional-order velocity (FV-PSO).

Table 11 shows that the number of dimensions for functions f_5 and f_8–f_{11} was set to $D_n = 10$, the number of particles was set to $S = 30$, and the number of iterations was set to $N = 300$. Table 12 shows the parameter settings for the proposed AUFPSO-NTE, FVFP-PSO, FP-PSO, FV-PSO, and PSO. Parameters $\omega_{\min}$, $\omega_{\max}$, $c_{1\min}$, $c_{1\max}$, $c_{2\min}$, and $c_{2\max}$ of the proposed AUFPSO-NTE were set as in Example (1). In FVFP-PSO, FP-PSO, and FV-PSO, the fractional-order velocities and positions were affected by factors ε and ζ. Notably, factors ω, ε, and ζ underwent time-varying evolution in FVFP-PSO, FP-PSO, and FV-PSO. Table 12 also shows that c_1 and c_2 were 1. In Table 13, λ is the fractional order of the derivative and α, β, and γ are coefficients that influence ω, c_1, and c_2, respectively. Table 13 shows the best combinations of constant coefficients λ, α, β, and γ for each benchmark function of the proposed AUFPSO-NTE.

Table 11. Settings for the number of dimensions, number of particles, and number of iterations in the proposed AUFPSO-NTE, fractional-order position (FP-PSO), fractional-order velocity (FV-PSO), FVFP-PSO, and PSO in Example (2).

Function		Number of Dimension (D_n)	Number of Particles (S)	Number of Iterations (I)
f_5	Rastrigin	10	30	300
f_8	Sphere	10	30	300
f_9	Ackley	10	30	300
f_{10}	Rosenbrock	10	30	300
f_{11}	Griewank	10	30	300

Table 12. Parameters settings for the proposed AUFPSO-NTE, FVFP-PSO, FP-PSO, FV-PSO, and PSO in Example (2).

Terms		Algorithms				
		AUFPSO-NTE	**FVFP-PSO**	**FP-PSO**	**FV-PSO**	**PSO**
ω	min max	0.4 0.9	0.4 0.9	0.4 0.9	0.4 0.9	N/A
c_1	min max	0 2	1	1	1	1
c_2	min max	0 2	1	1	1	1
ε	min max	N/A	0.1 1.2	1	0.1 1.2	N/A
ζ	min max	N/A	0.1 1.2	0.1 1.2	1	N/A

Table 13. The best combinations of the constant coefficients for each benchmark function of the proposed AUFPSO-NTE in Example (2).

Function	λ	α	β	γ
Rastrigin	1.3333	0.4444	0.2222	0.8889
Sphere	1.4689	0.5306	0.2595	0.9879
Ackley	1.5538	0.3519	0.1248	0.7688
Rosenbrock	1.2	0	1.7235	1.8864
Griewank	1.5111	0.2765	0.1136	0.8

Table 14 compares the mean values obtained using the proposed AUFPSO-NTE, FVFP-PSO, FP-PSO, FV-PSO, and PSO developed by Gao et al. [36] for f_5 and f_8–f_{11} in 100 independent trials. Table 14 shows that the means obtained by the proposed AUFPSO-NTE were better than those obtained using FVFP-PSO, FP-PSO, FV-PSO, and PSO.

Table 14. Means obtained using the proposed AUFPSO-NTE and for FVFP-PSO, FP-PSO, FV-PSO, and PSO for 5 benchmark functions in 100 independent trials in Example (2).

Function	**AUFPSO-NTE**	**FVFP-PSO**	**FP-PSO**	**FV-PSO**	**PSO**
Rastrigin	0	0	3.4182	20.3351	18.3371
Sphere	2.8280×10^{-41}	8.9588×10^{-36}	1.9469×10^{-19}	941.4338	8.2506×10^{-12}
Ackley	8.8818×10^{-16}	8.4555×10^{-15}	0.0299	10.4455	0.0231
Rosenbrock	7.7881	8.0633	8.8267	2.5590×10^{5}	56.5664
Griewank	0	0.0013	0.3770	10.3937	0.1041

4.3. Example (3): Comparison of the Proposed AUFPSO-NTE with AFO-FPSO, NCPSO, FO-DPSO, FPSO, APSO, DPSO, HPSO, and PSO

Example (3) compares the performance of the proposed AUFPSO-NTE with the AFO-DPSO developed by Guo et al. [38], and the NCPSO (new chaos PSO), FO-DPSO (fractional-order Darwinian PSO), FPSO, APSO (Adaptive PSO), DPSO (Darwinian PSO), HPSO (hybrid PSO), and PSO developed by Guo et al. [38]. The AFO-DPSO introduces fractional-order velocity into a Darwinian PSO algorithm and includes a mutation mechanism to overcome premature convergence, NCPSO improves the chaos-PSO algorithm, FO-DPSO is a fractional-order Darwinian PSO, APSO provides adaptive PSO to enable automatic control of parameters, DPSO is Darwinian particle swarm optimization, and HPSO combines the concept of evolutionary computation with PSO.

In Table 15, the D_n for functions f_5, f_8, f_9, f_{11}, and f_{12} were set to 30, the number of particles was set to $S = 30$, and the number of iterations was set to $N = 1000$. Tables 16 and 17 show the parameter settings for the proposed AUFPSO-NTE and for the AFO-DPSO, NCPSO, FO-DPSO, FPSO, APSO, DPSO, HPSO, and PSO. Parameters $\omega_{\min}$, $\omega_{\max}$, $c_{1\min}$, $c_{1\max}$, $c_{2\min}$, and $c_{2\max}$ of the proposed AUFPSO-NTE were set as in Example (1). For AFO-DPSO, ω was 1; c_1 and c_2 were 1.5 to 2.5, respectively; and δ was 0.05 to 0.1. For NCPSO, ω was 0.7298, and c_1 and c_2 were both 1.4962. For FO-DPSO and FPSO, ω was 0.9, λ was 0.632, and c_1 and c_2 were both 1.5. The APSO parameters were automatically set. In DPSO, ω was set to 0.9. In HPSO, $\omega_{\min}$ and $\omega_{\max}$ were set to 0.2 and 0.8, respectively, and c_1 and c_2 were both set to 2.5. In PSO, $\omega_{\min}$ and $\omega_{\max}$ were set to 0.4 and 0.9, respectively, and c_1 and c_2 were both set to 2. Table 18 illustrates the best combinations of constant coefficients λ, α, β, and γ for each benchmark function of the proposed AUFPSO-NTE. In Table 18, λ is the fractional order of the derivative and α, β, and γ are coefficients which influence ω, c_1, and c_2, respectively.

Table 15. Number of dimensions, number of particles, and number of iterations for the proposed AUFPSO-NTE and for AFO-DPSO, NCPSO, FO-DPSO, FPSO, APSO, DPSO, HPSO, and PSO in Example (3).

Function		Number of Dimension (D_n)	Number of Particles (S)	Number of Iterations (I)
f_5	Rastrigin	30	30	1000
f_8	Sphere	30	30	1000
f_9	Ackley	30	30	1000
f_{11}	Griewank	30	30	1000
f_{12}	DeJong F4	30	30	1000

Table 16. Parameter settings for the proposed AUFPSO-NTE and for AFO-DPSO, NCPSO, FO-DPSO, FO-PSO, APSO, DPSO, HPSO, and PSO in Example (3).

Terms		Algorithms				
		AUFPSO-NTE	AFO-FPSO	NCPSO	FO-DPSO	FPSO
ω	min	0.4	1	0.7298	0.9	0.9
	max	0.9				
c_1	min	0	1.5	1.4962	1.5	1.5
	max	2	2.5			
c_2	min	0	1.5	1.4962	1.5	1.5
	max	2	2.5			

Table 17. Parameter settings for the proposed AUFPSO-NTE and for AFO-DPSO, NCPSO, FO-DPSO, FPSO, APSO, DPSO, HPSO, and PSO in Example (3).

Terms		Algorithms				
		AUFPSO-NTE	APSO	DPSO	HPSO	PSO
ω	min	0.4	Auto-control	0.9	0.2	0.4
	max	0.9			0.8	0.9
c_1	min	0	Auto-control	N/A	2.5	2
	max	2				
c_2	min	0	Auto-control	N/A	2.5	2
	max	2				

Table 18. Best combinations of the constant coefficients for each benchmark function of the proposed AUFPSO-NTE in Example (3).

Function	λ	α	β	γ
Rastrigin	1.1111	0	1.5556	1.7778
Sphere	1.3333	0.4444	0.2222	0.8889
Ackley	1.3813	0.9368	0.9227	1.7236
Griewank	0.4444	1.1111	0.6667	2
DeJong F4	1.3333	0.4444	0.2222	0.8889

In Tables 19 and 20, the mean values obtained by the AUFPSO-NTE are compared with those obtained using AFO-DPSO, NCPSO, FO-DPSO, FPSO, APSO, DPSO, HPSO, and PSO given by Guo et al. [38]. Mean values obtained using each PSO-based algorithm for f_5, f_8, f_9, f_{11}, and f_{12} in 30 independent trials were recorded. Tables 19 and 20 show that the means obtained using the proposed AUFPSO-NTE were better than those obtained using AFO-DPSO, NCPSO, FO-DPSO, FPSO, APSO, DPSO, HPSO, and PSO. In Guo et al. [38], the performance was evaluated in terms of variance in the optimum (12):

$$\text{variances in optimum} = \sum_{i=1}^{30} \left| \frac{f_i - f_{avg}}{f_{max}} \right|^2, \tag{12}$$

where f_i is the ith fitness value and f_{avg} is the mean fitness value for 30 independent trials. The f_{max} is a normalization factor. When $|f_i - f_{avg}| > 1$, f_{max} is the maximum ($|f_i - f_{avg}|$); otherwise, $f_{max} = 1$.

Table 19. Means obtained using the proposed AUFPSO-NTE and using AFO-FPSO, NCPSO, FO-DPSO, and FO-PSO for 5 benchmark functions in 30 independent trials.

Function	AUFPSO-NTE	AFO-FPSO	NCPSO	FO-DPSO	FO-PSO
Rastrigin	0	1.8956×10^{-10}	4.3741×10^{-4}	4.2305×10^{-5}	3.5000×10^{-3}
Sphere	1.5042×10^{-124}	2.3420×10^{-14}	2.0279×10^{-9}	3.4728×10^{-7}	1.5340×10^{-5}
Ackley	8.8818×10^{-16}	3.6610×10^{-11}	9.0869×10^{-7}	1.3774×10^{-6}	1.4000×10^{-6}
Griewank	0	0	9.9050×10^{-11}	8.1377×10^{-9}	1.4184×10^{-7}
DeJong F4	1.3852×10^{-255}	6.3364×10^{-23}	2.0809×10^{-17}	8.8098×10^{-16}	9.2521×10^{-12}

Table 20. Means obtained using the proposed AUFPSO-NTE and using APSO, DPSO, HPSO, and PSO for 5 benchmark functions in 30 independent trials.

Function	AUFPSO-NTE	APSO	DPSO	HPSO	PSO
Rastrigin	0	1.0100	1.9899	4.8642	106.55
Sphere	1.5042×10^{-124}	1.4500×10^{-10}	0.0328	0.3876	370.04
Ackley	8.8818×10^{-16}	0.3550	2.4083	5.6972	11.4953
Griewank	0	0.0167	7.400×10^{-3}	0.0237	2.6100×10^{7}
DeJong F4	1.3852×10^{-255}	2.1300×10^{-10}	1.3752×10^{-5}	0.0635	4.3467×10^{3}

The variance in the optimum was also used to compare the performance of the proposed AUFPSO-NTE with algorithms given by Guo et al. [38]. Tables 21 and 22 compare the variance in the optimum obtained by the proposed AUFPSO-NTE and by AFO-DPSO, NCPSO, FO-DPSO, FPSO, APSO, DPSO, HPSO, and PSO. A variance in the optimum closer to 0 indicates a better performance. Tables 21 and 22 indicate that the proposed AUFPSO-NTE had a better variance in the optimum compared to AFO-DPSO, NCPSO, FO-DPSO, FPSO, APSO, DPSO, HPSO, and PSO.

Table 21. Variances in the optimum obtained using the proposed AUFPSO-NTE, AFO-FPSO, NCPSO, FO-DPSO, and FO-PSO for 5 benchmark functions in 30 independent trials.

Function	AUFPSO-NTE	AFO-FPSO	NCPSO	FO-DPSO	FO-PSO
Rastrigin	0	0.0017	0.0043	0.0137	0.0232
Sphere	1.6599×10^{-245}	0.0031	0.0597	0.4091	0.7505
Ackley	2.9170×10^{-61}	1.4637×10^{-5}	3.5416×10^{-4}	5.9058×10^{-4}	0.0025
Griewank	0	0.0116	0.1912	0.6574	0.8151
DeJong F4	0	0.0201	0.2765	0.7344	0.8836

Table 22. Variances in the optimum obtained using the proposed AUFPSO-NTE, APSO, DPSO, HPSO, and PSO for 5 benchmark functions in 30 independent trials.

Function	AUFPSO-NTE	APSO	DPSO	HPSO	PSO
Rastrigin	0	0.0173	0.0774	0.2162	0.3488
Sphere	1.6599×10^{-245}	0.5126	1.0068	1.6022	2.0978
Ackley	2.9170×10^{-61}	0.0011	0.0162	0.0200	0.9074
Griewank	0	0.6819	0.9371	1.3658	1.6408
DeJong F4	0	0.8381	0.9611	1.0130	1.7960

4.4. Example (4): Comparison of the Proposed AUFPSO-NTE with HAFPSO, GAPSO, HFPSO, FPSO, and PSO

Example (4) compares the performance of the proposed AUFPSO-NTE with the HAFPSO (hunter-attack fractional-order PSO) developed by Hosseini et al. [38], GAPSO (genetic algorithm-PSO) [59], HFPSO (hybrid firefly algorithm and PSO) [60], FPSO, and PSO developed by Hosseini et al. [38]. The HAFPSO introduces the concept of hunter-attack into the FODPSO and the GAPSO is a compound optimizer that introduces the crossover and mutation strategy of GA into PSO. The HFPSO combines the firefly optimization algorithm and PSO.

Table 23 shows that the number of dimensions for functions f_5, f_8, f_{10}, f_{11}, and f_{13}–f_{15} was set to $D_n = 50$, the number of particles was set to $S = 30$, and the number of iterations was set to $N = 1000$. Parameters $\omega_{\min}$, $\omega_{\max}$, $c_{1\min}$, $c_{1\max}$, $c_{2\min}$, and $c_{2\max}$ of the proposed AUFPSO-NTE were set as in Example (1). Except for the fractional order value (λ) in HAFPSO being 0.6, the parameter settings for the HAFPSO, GAPSO, HFPSO, FPSO, and PSO were not mentioned. In Table 24, λ is the fractional order of the derivative and α, β, and γ are coefficients that influence ω, c_1, and c_2, respectively. Table 24 shows the best combinations of the constant coefficients λ, α, β, and γ for each benchmark function of the proposed AUFPSO-NTE.

Table 23. Settings for the number of dimensions, number of particles, and number of iterations in the proposed AUFPSO-NTE and for HAFPSO, GAPSO, HFPSO, FPSO, and PSO in Example (4).

	Function	Number of Dimension (D_n)	Number of Particles (S)	Number of Iterations (I)
f_5	Rastrigin	50	30	1000
f_8	Sphere	50	30	1000
f_{10}	Rosenbrock	50	30	1000
f_{11}	Griewank	50	30	1000
f_{13}	Schwefel P1.2	50	30	1000
f_{14}	Quartic	50	30	1000
f_{15}	Salomon	50	30	1000

Table 24. Best combinations of the constant coefficients for each benchmark function of the proposed AUFPSO-NTE in Example (4).

Function	λ	α	β	γ
Rastrigin	1.2494	0	1.6361	1.8209
Sphere	1.3333	0.4444	0.2222	0.8889
Rosenbrock	1.7778	1.3333	0	1.5556
Griewank	0.2765	1.2	0.4889	2
Schwefel P1.2	1.3333	0.4444	0.2222	0.8889
Quartic	1.5111	0.2765	0.1136	0.8
Salomon	1.3087	1.6889	0.8592	0.0889

Table 25 shows the performance comparison results obtained using the proposed AUFPSO-NTE, and obtained using HAFPSO [38], GAPSO [59], HFPSO [60], FPSO, and PSO developed by Hosseini et al. [38] for f_5, f_8, f_{10}, f_{11}, and f_{13}–f_{15} in 100 independent trials. The table shows that the means and S.D. obtained using the proposed AUFPSO-NTE were better than those obtained using HAFPSO [38], GAPSO [59], HFPSO [60], FPSO, and PSO. Overall, the performance of the proposed AUFPSO-NTE was better than others in Example (4).

Table 25. Performance results obtained using the proposed AUFPSO-NTE in comparison with HAFPSO, GAPSO, HFPSO, FPSO, and PSO for 7 benchmark functions in 100 independent trials in Example (4).

Function	Terms	AUFPSO-NTE	HAFPSO	GAPSO	HFPSO	FPSO	PSO
Rastrigin	Mean	0	2.18×10^{-2}	6.76×10^{1}	8.83×10^{1}	7.40×10^{1}	7.90×10^{1}
	S.D.	0	2.83×10^{-2}	1.84×10^{1}	3.08×10^{1}	2.04×10^{1}	1.86×10^{1}
Sphere	Mean	1.89×10^{-121}	2.15×10^{-9}	3.67×10^{-3}	1.43×10^{-5}	7.51×10^{-3}	4.57×10^{-5}
	S.D.	1.75×10^{-120}	3.54×10^{-9}	6.11×10^{-3}	8.05×10^{-6}	3.20×10^{-2}	1.61×10^{-4}
Rosenbrock	Mean	4.88×10^{1}	1.00×10^{2}	8.14×10^{1}	1.53×10^{2}	1.16×10^{2}	1.06×10^{2}
	S.D.	1.06×10^{-1}	5.57×10^{1}	4.16×10^{1}	5.93×10^{1}	5.56×10^{1}	4.86×10^{1}
Griewank	Mean	0	1.27×10^{-2}	1.60×10^{-2}	9.88×10^{-1}	3.64×10^{-2}	5.81×10^{-2}
	S.D.	0	1.54×10^{-2}	2.06×10^{-2}	6.49×10^{-2}	6.61×10^{-2}	8.96×10^{-2}
Schwefel P1.2	Mean	6.98×10^{-123}	1.03×10^{3}	3.99×10^{1}	1.15×10^{3}	6.32×10^{2}	2.22×10^{3}
	S.D.	6.49×10^{-122}	4.83×10^{2}	3.53×10^{1}	1.09×10^{3}	4.32×10^{2}	8.84×10^{2}
Quartic	Mean	1.34×10^{-4}	1.34×10^{-2}	4.46×10^{-2}	3.70×10^{-2}	6.45×10^{-2}	5.23×10^{-2}
	S.D.	9.80×10^{-5}	3.91×10^{-3}	1.17×10^{-2}	1.28×10^{-2}	2.12×10^{-2}	1.90×10^{-2}
Salomon	Mean	7.99×10^{-3}	6.40×10^{-1}	7.85×10^{-1}	1.15	1.27	1.13
	S.D.	2.72×10^{-2}	7.91×10^{-2}	1.38×10^{-1}	1.99×10^{-1}	3.60×10^{-1}	2.74×10^{-1}

5. Conclusions

This study applied the AUED method to enhance the performance and effectiveness of the FPSO-NTE algorithm. Use of the AUED method in the proposed FPSO-NTE algorithm enabled a rapid automatic search for the best combination of four constant coefficients, namely λ, α, β, and γ. The major contribution of this paper was the use of AUED to improve the performance of the algorithm and to obtain a robust output. The above experimental and simulation results indicate that the proposed AUFPSO-NTE algorithm achieved a higher solution accuracy compared to the FPSO and PSO-FOV proposed by Solteiro Pires and coworkers [34,35], the FPSO proposed by Gao et al. [36], the AFO-DPSO proposed by Guo et al. [38], the HAFPSO proposed by Hosseini et al. [39], and the PSO-based algorithm described by them [34–36,38,39]. Examples demonstrated that the solutions obtained using the proposed AUFPSO-NTE algorithm were more consistent, i.e., more robust. Therefore, we conclude that the proposed AUFPSO-NTE algorithm had a superior effectiveness and performance.

Author Contributions: Formal analysis, P.-Y.Y. and F.-I.C.; funding acquisition, J.-T.T. and J.-H.C.; methodology, J.-T.T. and J.-H.C.; software, P.-Y.Y. and F.-I.C.; supervision, J.-T.T. and J.-H.C.; validation, P.-Y.Y. and F.-I.C.; writing—original draft, P.-Y.Y. and F.-I.C.; writing—review and editing, J.-T.T. and J.-H.C.

Funding: This research was funded in part by the Ministry of Science and Technology, Taiwan, R.O.C., grant numbers MOST 105-2221-E-992-304-MY3, MOST107-2221-E-992-086-MY3, and MOST107-2221-E-153-005-MY2, and in part by the "Intelligent Manufacturing Research Center" (iMRC) from the Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan, R.O.C. And The APC was funded by MOST107-2221-E-992-086-MY3.

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

References

1. Eberhart, R.; Kennedy, J. A New Optimizer Using Particle Swarm Theory. In Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 4–6 October 1995.
2. Wang, Y.; Feng, X.Y.; Huang, Y.X.; Pu, D.B.; Zhou, W.G.; Liang, Y.C. A Novel Quantum Swarm Evolutionary Algorithm and Its Applications. *Neurocomputing* **2007**, *70*, 633–640. [CrossRef]
3. Rezaee Jordehi, A.; Jasni, J. Parameter Selection in Particle Swarm Optimisation: A Survey. *J. Exp. Theor. Artif. Intell.* **2013**, *25*, 527–542. [CrossRef]
4. Chen, Q.; Yang, J.G.; Gou, J. Image Compression Method Using Improved PSO Vector Quantization. In Proceedings of the Advances in Natural Computation: First International Conference, ICNC, Changsha, China, 27–29 August 2005.
5. Navimipour, N.J.; Eslamic, F. Service Allocation in the Cloud Environments Using Multi-Objective Particle Swarm Optimization Algorithm based on Crowding Distance. *Swarm Evol. Comput.* **2017**, *35*, 56–64.
6. Kerdphol, T.; Fuji, K.; Mitani, Y.; Watanabe, M.; Qudaih, Y. Optimization of a Battery Energy Storage System Using Particle Swarm Optimization for Stand-Alone Microgrids. *Electr. Power Energy Syst.* **2016**, *81*, 32–39. [CrossRef]
7. Naderi, E.; Narimani, H.; Fathi, M.; Narimani, M.R. A Novel Fuzzy Adaptive Configuration of Particle Swarm Optimization to Solve Large-Scale Optimal Reactive Power Dispatch. *Appl. Soft Comput.* **2017**, *53*, 441–456. [CrossRef]
8. Chou, P.Y.; Tsai, J.T.; Chou, J.H. Modeling and Optimizing Tensile Strength and Yield Point on a Steel Bar Using an Artificial Neural Network with Taguchi Particle Swarm Optimizer. *IEEE Access.* **2016**, *4*, 585–593. [CrossRef]
9. Girish, B.S. An Efficient Hybrid Particle Swarm Optimization Algorithm in a Rolling Horizon Framework for the Aircraft Landing Problem. *Appl. Soft Comput.* **2016**, *44*, 200–221.
10. Shi, Y.; Eberhart, R.C. A Modified Particle Swarm Optimizer. In Proceedings of the IEEE International Conference on Evolutionary Computation World Congress on Computational Intelligence, Anchorage, AK, USA, 4–9 May 1998.
11. Shi, Y.; Eberhart, R.C. Empirical Study of Particle Swarm Optimization. In Proceedings of the Congress on Evolutionary Computation-CEC99, Washington, DC, USA, 6–9 July 1999.
12. Eberhart, R.C.; Shi, Y. Tracking and Optimizing Dynamic Systems with Particle Swarm. In Proceedings of the Congress on Evolutionary Computation, Seoul, Korea, 27–30 May 2001.
13. Ratnaweera, A.; Halgamuge, S.K.; Watson, H.C. Self-Organizing Hierarchical Particle Swarm Optimizer with Time-Varying Acceleration Coefficients. *IEEE Trans. Evol. Comput.* **2004**, *8*, 240–255. [CrossRef]
14. Chatterjee, A.; Siarry, P. Nonlinear Inertia Weight Variation for Dynamic Adaptation in Particle Swarm Optimization. *Comput. Oper. Res.* **2006**, *33*, 859–871. [CrossRef]
15. Yang, X.; Yuan, J.; Yuan, J. A Modified Particle Swarm Optimizer with Dynamic Adaptation. *Appl. Math. Comput.* **2007**, *189*, 1205–1213. [CrossRef]
16. Ko, C.N.; Chang, Y.P.; Wu, C.J. An Orthogonal-Array-based Particle Swarm Optimizer with Nonlinear Time-Varying Evolution. *Appl. Math. Comput.* **2007**, *191*, 272–279. [CrossRef]
17. Ali, M.M.; Kaelo, P. Improved Particle Swarm Algorithms for Global Optimization. *Appl. Math. Comput.* **2008**, *196*, 578–593. [CrossRef]
18. Chen, X.; Li, Y. On Convergence and Parameter Selection of an Improved Particle Swarm Optimization. *Int. J. Control Autom. Syst.* **2008**, *6*, 559–570.
19. Huang, S.R. Survey of Particle Swarm Optimization Algorithm. *Comput. Eng. Des.* **2009**, *30*, 1977–1980.

20. Li, X. Niching without Niching Parameters: Particle Swarm Optimization Using a Ring Topology. *IEEE Trans. Evol. Comput.* **2010**, *14*, 150–169. [CrossRef]
21. Tsai, H.C.; Tyan, Y.Y.; Wu, Y.W.; Lin, Y.H. Isolated Particle Swarm Optimization with Particle Migration and Global Best Adoption. *Eng. Optim.* **2012**, *44*, 1405–1424. [CrossRef]
22. Chen, W.N.; Zhang, J.; Lin, Y.; Chen, N.; Zhan, Z.H.; Chung, H.S.H.; Li, Y.; Shi, Y.H. Particle Swarm Optimization with an Aging Leader and Challengers. *IEEE Trans. Evol. Comput.* **2013**, *17*, 241–258. [CrossRef]
23. Pehlivanoglu, Y.V. A New Particle Swarm Optimization Method Enhanced with a Periodic Mutation Strategy and Neural Networks. *IEEE Trans. Evol. Comput.* **2013**, *17*, 436–452. [CrossRef]
24. Li, N.J.; Wang, W.J.; Hsu, C.C.; Chang, J.W.; Chang, J.W. Enhanced Particle Swarm Optimizer Incorporating a Weighted Particle. *Neurocomputing* **2014**, *124*, 218–227. [CrossRef]
25. Wang, L.; Yang, B.; Li, Y.; Zhang, N. A Novel Improvement of Particle Swarm Optimization Using Dual Factors Strategy. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC), Beijing, China, 6–11 July 2014.
26. Cheng, R.; Jin, Y. A Social Learning Particle Swarm Optimization Algorithm for Scalable Optimization. *Inf. Sci.* **2015**, *291*, 43–60. [CrossRef]
27. Lynn, N.; Suganthan, P.N. Heterogeneous Comprehensive Learning Particle Swarm Optimization with Enhanced Exploration and Exploitation. *Swarm Evol. Comput.* **2015**, *24*, 11–24. [CrossRef]
28. Tsai, J.T.; Chou, P.Y.; Chou, J.H. Color Filter Polishing Optimization Using ANFIS with Sliding-Level Particle Swarm Optimizer. *IEEE Trans. Syst. Man Cybern. Syst.* **2017**. [CrossRef]
29. Kao, Y.T.; Zahara, E. A Hybrid Genetic Algorithm and Particle Swarm Optimization for Multimodal Functions. *Appl. Soft Comput.* **2008**, *8*, 849–857. [CrossRef]
30. Xin, B.; Chen, J. A Survey and Taxonomy on Hybrid Algorithms based on Particle Swarm Optimization and Differential Evolution. *J. Syst. Sci. Math. Sci.* **2011**, *31*, 1130–1150.
31. Noel, M.M. A New Gradient Based Particle Swarm Optimization Algorithm for Accurate Computation of Global Minimum. *Appl. Soft Comput.* **2012**, *12*, 353–359. [CrossRef]
32. Sun, Y.; Zhang, L.; Gu, X. A Hybrid Co-Evolutionary Cultural Algorithm based on Particle Swarm Optimization for Solving Global Optimization Problems. *Neurocomputing* **2012**, *98*, 76–89. [CrossRef]
33. Zhao, C.N.; Li, Y.S.; Lu, T. *Analysis and Design of Fractional Order System*; National Defense Industry Press: Beijing, China, 2011.
34. Solteiro Pires, E.J.; Tenreiro Machado, J.A.; Moura Oliveira, P.B.; Boaventura Cunha, J.; Mendes, L. Particle Swarm Optimization with Fractional-Order Velocity. *Nonlinear Dyn.* **2010**, *61*, 295–301. [CrossRef]
35. Solteiro Pires, E.J.; Tenreiro Machado, J.A.; Moura Oliveira, P.B. Fractional Particle Swarm Optimization. In *Mathematical Methods in Engineering*; Fonseca, N.M., Tenreiro Machado, J.A., Eds.; Springer: London, UK, 2014; pp. 47–56.
36. Gao, Z.; Wei, J.; Liang, C.; Yan, M. Fractional-Order Particle Swarm Optimization. In Proceedings of the 26th Chinese Control and Decision Conference (CCDC), Changsha, China, 31 May–2 June 2014.
37. Couceiro, M.; Ghamisi, P. *Fractional Order Darwinian Particle Swarm Optimization: Applications and Evaluation of an Evolutionary Algorithm*; Springer: Cham, Switzerland, 2016; pp. 11–20.
38. Guo, T.; Lan, J.L.; Li, Y.F.; Chen, S.W. Adaptive Fractional-Order Darwinian Particle Swarm Optimization Algorithm. *J. Commun.* **2014**, *35*, 130–140.
39. Akbar, S.; Zaman, F.; Asif, M.; Rehman, A.U.; Raja, M.A.Z. Novel application of FO-DPSO for 2-D parameter estimation of electromagnetic plane waves. *Neural Comput. Appl.* **2019**, *31*, 3681–3690. [CrossRef]
40. Ates, A.; Alagoz, B.B.; Kavuran, G.; Yeroglu, C. Implementation of fractional order filters discretized by modified fractional order darwinian particle swarm optimization. *Measurement* **2017**, *107*, 153–164. [CrossRef]
41. Shahri, E.S.A.; Alfi, A.; Machado, J.T. Fractional fixed-structure H_∞ controller design using augmented lagrangian particle swarm optimization with fractional order velocity. *Appl. Soft Comput.* **2019**, *77*, 688–695. [CrossRef]
42. Wei, J.R.; Ma, Y.; Xia, R.; Jiang, H.B.; Zhou, T.T. Image segmentation algorithm based on Otsu optimized by fractional-order particle swarm optimization. *Comput. Eng. Des.* **2017**, *38*, 3284–3290.
43. Guo, F.; Peng, H.; Zou, B.; Zhao, R.; Liu, X. Localisation and segmentation of optic disc with the fractionalorder Darwinian particle swarm optimization algorithm. *IET Image Process.* **2018**, *12*, 1303–1312. [CrossRef]

44. Ahilan, A.; Manogaran, G.; Raja, C.; Kadry, S.; Kumar, S.N.; Agees Kumar, C.; Jarin, T.; Krishnamoorthy, S.; Kumar, P.M.; Babu, G.C.; et al. Segmentation by Fractional Order Darwinian Particle Swarm Optimization Based Multilevel Thresholding and Improved Lossless Prediction Based Compression Algorithm for Medical Images. *IEEE Access.* **2019**, *7*, 89570–89580. [CrossRef]
45. Tang, Q.; Gao, S.; Liu, Y.; Yu, F. Infrared image segmentation algorithm for defect detection based on FODPSO. *Infrared Phys. Technol.* **2019**, *102*, 103051. [CrossRef]
46. Wang, Y.Y.; Peng, W.X.; Qiu, C.H.; Jiang, J.; Xia, S.R. Fractional-order Darwinian PSO-based feature selection for media-adventitia border detection in intravascular ultrasound images. *Ultrasonics* **2019**, *92*, 1–7. [CrossRef]
47. Hosseini, S.A.; Hajipour, A.; Tavakoli, H. Design and optimization of a CMOS power amplifier using innovative fractional-order particle swarm optimization. *Appl. Soft Comput.* **2019**, *85*, 105831. [CrossRef]
48. Akdağ, O.; Okumuş, F.; Kocamaz, A.F.; Yeroglu, C. Fractional Order Darwinian PSO with Constraint Threshold for Load Flow Optimization of Energy Transmission System. *Gazi Univ. J. Sci.* **2018**, *31*, 831–844.
49. Zameer, A.; Muneeb, M.; Mirza, S.M.; Raja, M.A.Z. Fractional-order particle swarm based multi-objective PWR core loading pattern optimization. *Ann. Nucl. Energy* **2020**, *135*, 106982. [CrossRef]
50. Wang, Y.Y.; Zhang, H.; Qiu, C.H.; Xia, S.R. A Novel Feature Selection Method Based on Extreme Learning Machine and Fractional-Order Darwinian PSO. *Comput. Intell. Neurosci.* **2018**, *2018*, 5078268. [CrossRef]
51. Taguchi, G.; Chowdhury, S.; Taguchi, S. *Robust Engineering*; McGraw-Hill: New York, NY, USA, 2000.
52. Wang, Y.; Fang, K.T. A Note on Uniform Distribution and Experimental Design. *Chin. Sci. Bull.* **1981**, *26*, 485–489.
53. Tsao, H.; Lee, L. Uniform Layout Implement on Matlab. *Stat. Decis.* **2008**, *6*, 144–146.
54. Tsai, J.T.; Yang, P.Y.; Chou, J.H. Data-Driven Approach to Using Uniform Experimental Design to Optimize System Compensation Parameters for an Auto-Alignment Machine. *IEEE Access.* **2018**, *6*, 40365–40378. [CrossRef]
55. Diethelm, K. *The Analysis of Fractional Differential Equations—An Application-Oriented Exposition Using Differential Operators of Caputo Type*; Springer: London, UK, 2010.
56. Podlubny, I. *Fractional Differential Equations*; Academic Press: San Diego, CA, USA, 1999.
57. Guo, B.; Pu, X.; Huang, F. *Fractional Partial Differential Equations and Their Numerical Solutions*; World Scientific: Singapore, 2015.
58. Cui, Z.H.; Zeng, J.C. *Particle Swarm Optimization*; Science Press: Beijing, China, 2011.
59. Liu, H.; Zhai, R.; Fu, J.; Wang, Y.; Yang, Y. Optimization study of thermal-storage PV-CSP integrated system based on GA-PSO algorithm. *Sol. Energy* **2019**, *184*, 391–409. [CrossRef]
60. Aydilek, İ.B. A hybrid firefly and particle swarm optimization algorithm for computationally expensive numerical problems. *Appl. Soft Comput.* **2018**, *66*, 232–249. [CrossRef]