Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 4194263, 24 pages
https://doi.org/10.1155/2021/4194263

Research Article

An Expanded Heterogeneous Particle Swarm Optimization Based on Adaptive Inertia Weight

Sami Zdiri, Jaouher Chrouta, and Abderrahmen Zaafouri

Laboratory of Engineering of Industrial Systems and Renewable Energy (LISIER), National Higher Engineering School of Tunis (ENSIT), University of Tunis, Tunis, Tunisia

Correspondence should be addressed to Sami Zdiri; zdiri_sami@yahoo.fr

Received 30 April 2021; Revised 15 July 2021; Accepted 25 August 2021; Published 12 October 2021

Academic Editor: Dilbag Singh

In this study, a modified version of multiswarm particle swarm optimization algorithm (MsPSO) is proposed. However, the classical MsPSO algorithm causes premature stagnation due to the limitation of particle diversity; as a result, it is simple to slip into a local optimum. To overcome the above feebleness, this work presents a heterogeneous multiswarm PSO algorithm based on adaptive inertia weight strategies called (A-MsPSO). The MsPSO's main advantages are that it is simple to use and that there are few settings to alter. In the MsPSO method, the inertia weight is a key parameter affecting considerably convergence, exploration, and exploitation. In this manuscript, an adaptive inertia weight is adopted to ameliorate the global search ability of the classical MsPSO algorithm. Its performance is based on exploration, which is defined as an algorithm's capacity to search through a variety of search spaces. It also aids in determining the best ideal capability for searching a small region and determining the candidate answer. If a swarm discovers a global best location during iterations, the inertia weight is increased, and exploration in that direction is enhanced. The standard tests and indications provided in the specialized literature are used to show the efficiency of the proposed algorithm. Furthermore, findings of comparisons between A-MsPSO and six other common PSO algorithms show that our proposal has a highly promising performance for handling various types of optimization problems, leading to both greater solution accuracy and more efficient solution times.

1. Introduction

Particle swarm optimization (PSO) is a stochastic population-based algorithm developed by Eberhart and Kennedy [1]. PSO was derived from the intelligent collective behavior of animal species (e.g., flocks of birds), especially that of searching food. At the start of plowing, one bird discovers the food source. Then, it is followed very quickly by another and so on. In this process, the information regarding a potential feast is widely disseminated within the group of gulls that fly in search of food in a more or less orderly manner. The gathering takes place through a (voluntary or not) social exchange of information between individuals of the same species. One of them found a solution and the others adopt it. Kennedy and Eberhart [2] initially attempted to simulate the birds' ability to fly synchronously and to shift a path suddenly while staying in optimum shape. Then, they expanded their model into a simple and efficient optimization algorithm and the particles are individuals moving through the research hyperspace. PSO has been effectively deployed in a number of locations in recent years. Due to its multiple advantages, such as a wide search range, speedy convergence, and ease of implementation, it has been used in a variety of research fields and applications. On another side, since the PSO algorithm is easily stuck in a local optimum, it was greatly improved. The abovementioned enhancements are organized by the particular gap in the PSO they seek to fill.

The Binary PSO. To evolve the network architecture, Eberhart and Kennedy introduced a binary version of the PSO [3] to represent and solve problems of a binary nature as well as to compare genetic algorithms (GA) encoded binarily with PSO.

Rate of Convergence Improvements. Different strategies have been suggested to increase the PSO's convergence rate. They usually require improving the PSO update equations without altering the algorithm structure, which generally enhances the optimization of local performance, in several local minima functions. There are three parameters that change the PSO update equations: the inertia weight control [4–6] and the two acceleration coefficients. In fact, the inertia weight is among the earliest improvements of the original PSO made to further enhance the algorithm convergence rate. One of the most common enhancements is the implementation of Shi and Eberhart inertia weights [7]. In 1998, the authors suggested a technique to dynamically adapt the inertia weight using a fuzzy controller [8]. Then, researchers, in [9], demonstrated that the application of a constriction factor is required for the PSO algorithm to converge.

In the same framework and from another analytical view, PSO algorithms have been successfully used in a multitude of industrial applications and are readily trapped in local optima. To solve such issue, several improvements have been made: enhancing particle displacement adjustment parameters and improving topological structures, heterogeneous updating rules, and population grouping using multiple swarms. In [10], Liang and Sughantan have proposed a dynamic multiswarm particle swarm optimizer (DMSPSO) wherein the entire population is split in numerous swarms by applying clustering strategies to regroup these swarms and ensure the exchange of information between them. In [11], Yen and Leong have suggested two strategies, namely, swarm growth strategy and swarm decline strategy, which enable swarms to distribute information with the best individuals in the population. In addition, in [12], a cooperative method of PSO, which divides the population into four subswarms and affects significantly the convergence of the algorithm, has been introduced. In [13], Li et al. proposed an improved adaptive holonic PSO algorithm that did not consider connections between particles and used a clustering strategy. Obviously, the introduced algorithm enhanced the search efficiency. In [14], two optimization examples were solved using an improved optimization algorithm (OIPSO): optimization of resource constraints with the shortest project duration and optimization of resource leveling at fixed duration. Through these last two, it was proven that the developed algorithm allowed improving the optimization effect of PSO and accelerating the optimization speed. Similarly, in 2014, a new metaheuristic search algorithm has been proposed by Ngaam [15]; the population was partitioned into 4 subswarms: 2 basic subswarms for exploitation search as well as adaptive subswarm and the exploration subswarm. The main advantage of MsPSO is the possibility of using easily the tuning parameters, which improves the search performance of the algorithm. Despite the reported improvements, they cannot always meet the requirements of applications in various areas. Because of the importance of inertia weight as a tuning parameter used to ensure convergence and improve accuracy, Zdiri et al. [16] have performed a comparative study of the different inertia weight tuning strategies and they have also shown that the adaptive inertia weight is the best technique that allows obtaining greater precision. In this context and to assess the MsPSO algorithm performance, a new modified version, named multiswarm particle swarm optimization algorithm based on adaptive inertia weight (A-MsPSO), is proposed in this study. The main contribution of this article is resumed as follows: the population is split into 4 subswarms, each of which represents a single PSO. In each subswarm, we use an adaptive inertia weight strategy which oversees the search situation and adapts according to the evolutionary state of each particle. Adaptation is carried out, at each iteration, by a mechanism defined by the best overall aptitude (Gbest). The latter is determined from the best overall aptitude of each subswarm. This takes place with 2 constant acceleration coefficients applied to enhance the local search capacity of each subswarm and that of global exploration and speed of convergence. The remaining of this paper is structured as follows: Section 2 presents the PSO algorithm. MsPSO algorithm is defined in Section 3. Section 4 presents the introduced optimization algorithm (A-MsPSO). In Section 5, the relation between the position and the velocity of the particles in each subswarm of A-MsPSO is well explained. Section 6 presents the simulation results and shows the efficiency of the introduced algorithm by comparing it with other approaches. Section 7 suggests the future work possibilities. Finally, Section 8 concludes this work.

2. Particle Swarm Optimization

2.1. Definition. At the outset of the algorithm, the particles of the swarm are randomly distributed in the search space where each particle has a randomly defined movement speed and position. Thereafter, it is able, at each instance, to

(i) Evaluate its current position and memorize the best one it has reached so far and the fitness function value in this position.

(ii) Communicate with the neighboring particles, obtain from each of them its own best performance, modify its speed according to best solutions, and consequently, define the direction of the next displacement. The strategy of particle displacement is explained in Figure 1.

2.2. Formalization. The i^{th} particle displacement between iteration t and iteration $t+1$ is formulated analytically by the two following relations of velocity and position, respectively,

$$v_i(t+1) = wv_i(t) + c_1 r_1 \left[x_{\text{pbest}} - x_i(t)\right] + c_2 r_2 \left[x_{\text{gbest}} - x_i(t)\right], \tag{1}$$

$$x_i(t+1) = x_i(t) + v_i(t+1), \tag{2}$$

where w is a constant called inertia weight; x_{pbest} denotes the best historical personal position determined by the i^{th} particle; x_{gbest} corresponds to the best overall position found by the population; c_1 designates the cognitive parameter; c_2 is the social parameter; and r_1, r_2 are generated randomly by a uniform distribution in $[0, 1]$.

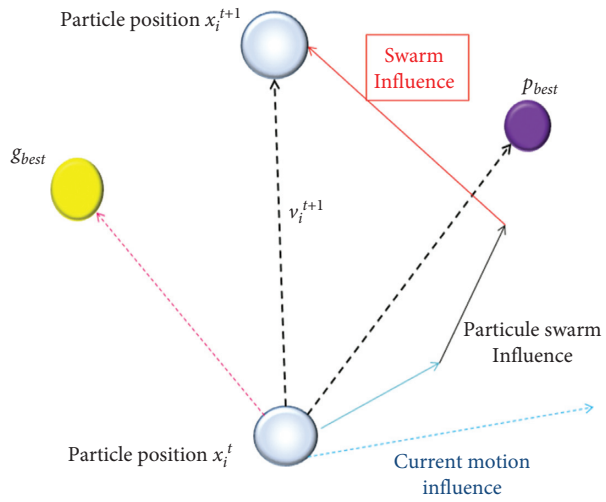


FIGURE 1: Schematic representation of the particle movement in a swarm.

2.3. PSO Improvement History. To improve the PSO algorithms, several modifications have been proposed. They can be grouped into three categories.

2.3.1. Association of PSO with Other Search Operators. In intelligent systems, hybridization consists in combining the desirable properties of different methods aimed at minimizing their weaknesses. Multiple hybrid PSO systems have been used in specific application contexts. In what follows, we briefly review some of these approaches. Among the techniques widely used to increase PSO performance, we can mention hybridization with evolutionary algorithms (AE) and particularly genetic algorithms (GA). Angeline [17] utilized a tournament selection mechanism whose purpose is to replace the position and speed of each poorly-performing particle with those of the best performing particle. This movement in space improves performance in 3 test functions used, yet it has moved away from the classical PSO. Moreover, Zhang and Xie used, in [18], different techniques in tandem in their PSO with differential evolution (DEPSO). The canonical PSO and DE (differential evolution) operators were utilized in alternate generations. Obviously, hybridization was efficiently applied in some test functions and the obtained results indicate that DEPSO can improve the PSO efficiency in solving larger dimensional problems. Liu and Abraham, in [19], hybridized a turbulent PSO (TPSO) with a fuzzy logic controller. The TPSO was based on the idea that particles stuck in a suboptimal solution, which led to the premature convergence of PSO. The velocity parameters were, then, adaptively regulated during optimization using a fuzzy logic extension. The efficiency of the suggested method in solving high and low dimensionality problems was tested, with positive results in both cases. In particular, when the performance of canonical PSO deteriorated considerably in the face of problems with strong dimensionality, the TPSO and FATPSO were slightly affected. Ying Tan described a PSO algorithm based on a cloning process (CPSO) inspired by the natural immune system in [20]. The method is characterized by its quick convergence and ease of implementation. The applied algorithm increased the area adjacent to the potential solution and accelerated the evolution of the swarm by cloning the best individual in the following generations, resulting in greater optimization capacity and convergence. To diversify the population, Xiao and Zuo [21] adopted a multipopulation technique, using each subpopulation, to a separate peak. Then, they utilized a hybrid DEPSO operator in order to determine the optima in each one. Moving peaks benchmark tests yielded much lower average offline error than competing solutions. Chen et al. integrated the PSO and CS (cuckoo search), in [22], to develop PSOCS, where cuckoos near excellent solutions interacted with each other, migrated slowly towards the ideal solutions and led by the global bests in PSO. In [23], a new crow swarm optimization (CSO) algorithm, that combines PSO with CSA (crow search algorithm) and allows people to explore unknown locations while being guided by a random individual, was suggested. CSO algorithm was tested on several benchmark functions. The suggested CSO's performance was measured in terms of optimization efficiency and global search capability and all of that are vastly enhanced over either PSO or CSA.

2.3.2. Improvement of the Topological Structure. There are two original versions of PSO: the classic local ring version (Lbest) and the global star version (Gbest). In addition to these two original models, several topologies were proposed. The majority of these enhanced the PSO algorithm's performance. A variety of topologies were introduced and improved from static topologies to dynamic ones. Some of the widely used and successful topologies are presented below.

(1) Static Topologies. The best known static versions are the global version and the classic local version. In the latter, the swarm converges more slowly than in Gbest, but can probably locate the global optimum. In general, using the Lbest model reduces the risks of premature convergence of the algorithm, which affects positively the performance of the algorithm, particularly for multimodal problems. We present some more recent examples as follows:

(2) Four Clusters [24]. The four-cluster topology uses four groups of particles linked together by several gateways. From a sociological point of view, this topology resembles four isolated communities. In each community, a few particles can communicate with a particle belonging to another community. The particles are divided into four groups, each of which consists of five particles and exchanges information with the other three groups. Each gateway is linked to only one group. This topology is characterized by the number of edges that indirectly connect two particles. The diameter of the graph is three, which means that information can be transmitted from one particle to another by borrowing a maximum of three edges.

(3) Wheel or Focal [25]. In this topology, all particles are isolated from each other and they are related only to the focal particle. Moreover, information is communicated by this particle that uses this data to adapt its trajectory.

In [26], the authors developed a new improved algorithm (FGLT-PSO) named fusion global-local topology particle swarm optimization that simultaneously utilizes local and global topologies in PSO to get out of local optima. This proposal enhanced the PSO algorithm performance in terms of solution precision and convergence velocity. The most important studies dealing with this network are represented in Figure 2.

(4) Dynamic Topologies. The authors tried to set up dynamic topologies having better performance than static topologies by changing their structures from one iteration to another. This change allows particles to alter their travel paths so that they can escape local optima. In the following part, we will discuss a set of dynamic topologies:

Fitness-distance ratio [27]: it is a variation of PSO based on a special social network; it was inspired by the observation of animal behavior. This social network does not use a specific geometric structure to effect communication between particles. In fact, the position and speed of a particle are updated by dimension. For each dimension, the particle has only one informant, which prevents several informants from communicating information that cancels each other out. The choice of this informant, named nbest (best nearest), is based on two criteria: (i) it has to be close to the current particle and (ii) it must have visited a position with better fitness.

Bastos-Filho [28]: in this structure, the swarm is divided into groups or clans of particles. Each clan, containing a fixed number of fully-connected particles, admits a leader. The authors added the possibility of the particles migration from one clan to another and, at each iteration, the particle with the best location was the leader.

2.3.3. Control of the PSO Algorithm Parameters. Many recent works focusing on optimization have shown that the best performance can be obtained only by selecting the adequate adjustment parameters [29, 30]. In PSO, two acceleration coefficients and the inertia weight are the three main factors affecting the algorithm performance. They affect also the orientation of the particle on its future displacement in order to guarantee a balance between local research and global research. Their control prevents the swarm from exploding and makes convergence easier.

The inertia weight allows defining the exploration capacity of each particle in order to improve its convergence. Great value equals great range of motion and, therefore, global exploration. A weak value is synonymous to low range of motion and, thus, local exploration. Therefore, fixing this factor amounts to finding a compromise between local exploration and global exploration. The size of the inertia weight influences the size of explored hyperspace and no value can guarantee convergence towards the optimal solution. Like the performance of other swarm algorithms, that of PSO can be improved by selecting the right parameters. Thus, Clerc [31] gave some general guidelines for choosing the right combination. The acceleration coefficients were first used by Kennedy and Eberhart. The social parameter they utilized is the same as the cognitive parameter with a constant value. Although several enhancements were made, the combination of the parameters w, c_1, and c_2 is the only solution that makes the adjustment of the balance between the phases of intensification and diversification of the research process possible. For example, in [32], a new optimization strategy for particle swarms (APSO) was formulated; the coefficients are equal to 2.0 and adjusted adaptively according to the evolutionary state (exploitation, exploration, convergence, and jumping out). It allows the automated adjustment of acceleration coefficients and inertia weight during runtime, resulting in faster convergence and better search efficiency. PSO presents a major problem called premature convergence. This problem can lead to the algorithm stagnation in local optimum. In fact, much effort has been made by scientists and researchers to enhance the performance of this algorithm. In 2014 [15], a new algorithm, called multiswarm particle swarm optimization based on the distribution of population into four subswarms cooperating through a specific strategy, has been proposed by Ngaam. In the following section, we will present MsPSO in detail.

3. Multiswarm Particle Swarm Optimization Algorithm (MsPSO)

3.1. General Description. This new optimization algorithm is an improved version of PSO. In this approach, the population is divided into four subswarms (an adaptive subswarm, an exploring subswarm, and two basic subswarms) to obtain the best management of the exploration and the exploitation of the research process in order to reach the best solution. The subswarm cooperation and communication model is shown in Figure 3.

S_1 and S_2 are the basic subswarms, whereas S_3 and S_4 are the adaptive and exploration subswarms, respectively. All the subswarms share information with each other and participate in the overall exploitation. Based on cooperation and information sharing, the particles in S_3 use the speed messages and fitness values of the particles in S_1 and S_2 to refresh their speeds and positions. However, the speed of the particles in S_4 is updated from the concentration of particles in S_1, S_2, and S_3. To improve the PSO convergence rate, several techniques, modifying the inertia weights and/or the acceleration coefficients in the algorithm update equations, have been recently developed.

3.2. The Inertia Weight in the Literature. The inertia weight, which was first developed by Shi and Eberhart to ensure a balance between exploitation and exploration, controls the effect of the particle's orientation on future displacement. The authors showed that, if $w \in [0.8, 1.2]$ convergence is faster with a reasonable number of iterations and a large value of w facilitates a global exploration, a small value will make it easier to explore locally. Several methods were

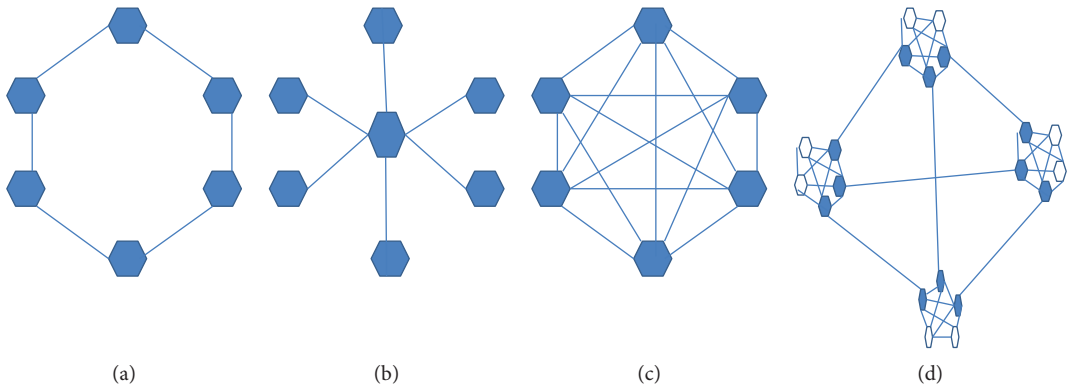


Figure 2: Different types of topologies of a particle swarm: (a) ring; (b) ray; (c) star; and (d) four clusters.

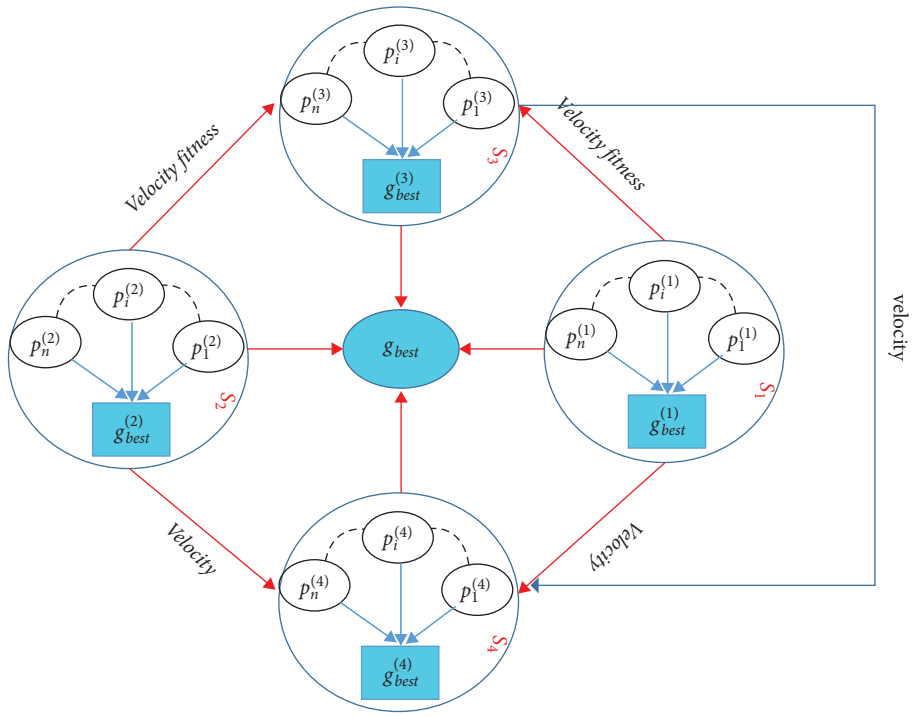


Figure 3: Model of communication between the subswarms in MsPSO.

suggested to solve PSO-based optimization problems by using an inertia weight in the particle velocity equation with some modifications. The forces changing the inertia weight can be classified into three groups.

3.2.1. Constant Inertia Weight. In many introduced methods, PSO-based optimization problems were solved by utilizing a constant inertia weight in a modified particle velocity equation. In [33], for example, Shi and Eberhard used a random value of w to follow the optima.

$$w = 0.5 + \frac{\text{rand}(\,)}{2}, \tag{3}$$

where rand( ) is generated randomly by a uniform distribution in $[0, 1]$.

3.2.2. Time-Varying Inertia Weight. In this class, the inertia weight value is determined as a function of time using a specific number of iterations. These strategies can be nonlinear or linear and decreasing or increasing. In what follows, we present some of these methods that have an impact factor in the literature. For example, in [34], Xin and Chen introduced a decreasing linear inertia weight efficiently used to enhance the fine tuning characteristics of the PSO. Besides, in [35], the authors added a chaotic term to the weight of inertia of linear growth. Moreover, Kordestani and Meybodi [36] developed an oscillating triangular inertia weight (OTIW) to follow the optima in a dynamic environment.

3.2.3. Adaptive Inertia Weight. Inertia weight techniques are employed in the third category to keep track of the search scenario and change the inertia weight value relying on the

feedback parameters [37]. In [38], Rao and Arumugam employed the ratio between the best overall shape and the best local mean shape of the particles in order to calculate the inertia weight in each iteration and avoid the premature convergence towards the local minimum. The strategy of the adaptive inertia weight has been made available to improve its research capacity and control the diversity of the population by an adaptive adjustment of the inertia weight.

3.3. Limitations and Discussion of MsPSO Algorithm. The MsPSO algorithm presents three main adjustment parameters: w, c_1, and c_2 that affect significantly the convergence of the algorithm. The four subswarms use two constant acceleration coefficients and time-varying inertia weight. The MsPSO algorithm contains a time-varying inertia weight function given by

$$w(i) = \frac{0.9 \times \text{numiter} - 0.5i}{\text{numiter}}. \tag{4}$$

Besides, c_1 and c_2, are adjusted to a constant value.

$$c_1 = c_2 = 1.494. \tag{5}$$

This category of functions can cause incorrect update of the particle velocity as no data about the evolving state showing the population diversity is identified or used. Therefore, the choice of the parameters can disturb the direction of the future displacement of each particle and degrade the algorithm performance in terms of computation time and convergence. To overcome the mentioned problems, we suggest, in this manuscript, a modified version of the MsPSO algorithm called adaptive multiswarm particle swarm optimization (A-MsPSO) where a mechanism is applied to adjust both the inertia weight and acceleration coefficient. The advantage of this mechanism is that it considers the evolutionary state of the particles of the four abovementioned subswarms. This enhancement made to introduce a process of exchange and cooperation between the subswarms to guarantee the best exploration and the most efficient exploitation of the search space.

4. Adaptive Multiswarm Particle Swarm Optimization Algorithm (A-MsPSO)

4.1. General Description of A-MsPSO Algorithm. A-MsPOS algorithm is used in this paper to enhance the exploration and exploitation of the basic MsPSO that incorporates four subswarms, an adaptive inertia weight method, and two constant acceleration coefficients. The populations are split into four subswarms containing N particles in a search space of D dimension where each subswarm runs a single PSO and searches a local optimum or, at the same time, a global optimum. Thus, from the best local individual of the four subswarms, the best global optimum can be obtained as follows: $g_{\text{best}} = \{f(g_{\text{best}}^{\text{sub}}/\text{sub} = 1, 2, 3, 4)\}$. The particle settings guarantee the update of speed and position using their subswarms equation (the details are presented in subsection 4.2). The four subswarms maintain specific binding to enhance the search capability. The cooperation and communication between the subswarms are presented in Figure 4. The pseudocode of the introduced A-MsPSO is provided in Algorithm 1 and the organization chart is exposed in Figure 5. The A-MsPSO algorithm contains 4 subswarms and uses various cooperative strategies to broaden the exploration and exploitation of the MsPSO.

Figure 6 describes the mechanism of exploring the new region where x_1, x_2, and x_3 positions of the particles of three subswarms S_1, S_2, and S_3 are randomly placed in the search space; they approach the new captured Gbest position. At time T, the velocities of each particle are defined randomly and their position, at time $T+1$, is determined according to the equations update. Position x_4 has the ability to explore a new area from which another space of unknown search is provided using the modification equation (12). The search solutions are obtained from a circle containing the overall optimum. Graphically, the old region and the new one are separated by an arc of a circle. Furthermore, the i^{th} particle in S_3 is modified according to the fitness values and the velocity of the particles in the basic subswarms. In S_4, the i^{th} particle changes its velocity depending on velocities of the particles in S_1, S_2, and S_3.

4.2. A-MsPSO Search Behavior. The four subswarms collaborate and search for the optimum Gbest solution by using the best solutions of each subswarm. As shown in Figure 4, the global optimum is defined as a function of the global optimum of the 4 subswarms. The expressions of the speed and position of the particles of S_1 and S_2 are given by the following equations:

$$\begin{aligned} v_i^{(1/2)}(t+1) &= w_a v_i^{(1/2)}(t) + c_1 r_1 \left[p_i^{(1/2)} - x_i^{(1/2)}(t)\right] \\ &\quad + c_2 r_2 \left[g_{\text{best}} - x_i^{(1/2)}(t)\right], \end{aligned} \tag{6}$$

where r_1 and r_2 are randomly generated in $[0, 1]$, the superscript of (1/2) represents the basic subswarms; and w_a is the adaptive inertia weight obtained by

$$w_a(i) = w_{\max} + (w_{\max} - w_{\min}) \times \left(\frac{e^{x(i)-1}}{e^{x(i)+1}}\right), \tag{7}$$

where

$$x(i) = \frac{g_{\text{best}} - i}{g_{\text{best}} + i}, \tag{8}$$

where g_{best} represents the best position shared by the entire population; $w_{\min}$ and $w_{\max}$ denote initial value and final one, respectively.

The particles in the adaptive subswarm S_3 adjust their flight directions to the base of best particles in S_1 and S_2. The speed in S_3 is revised in accordance with the following equation:

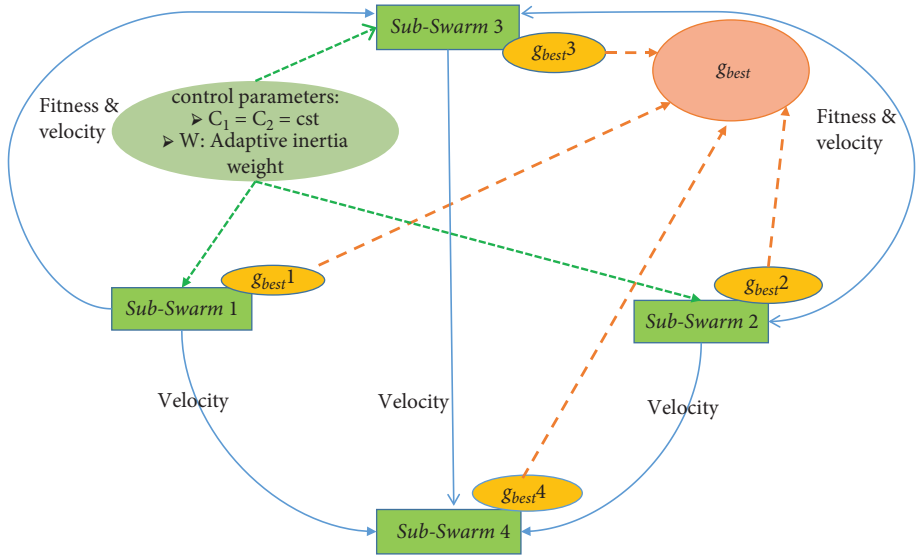


Figure 4: Communication model in A-MsPSO.

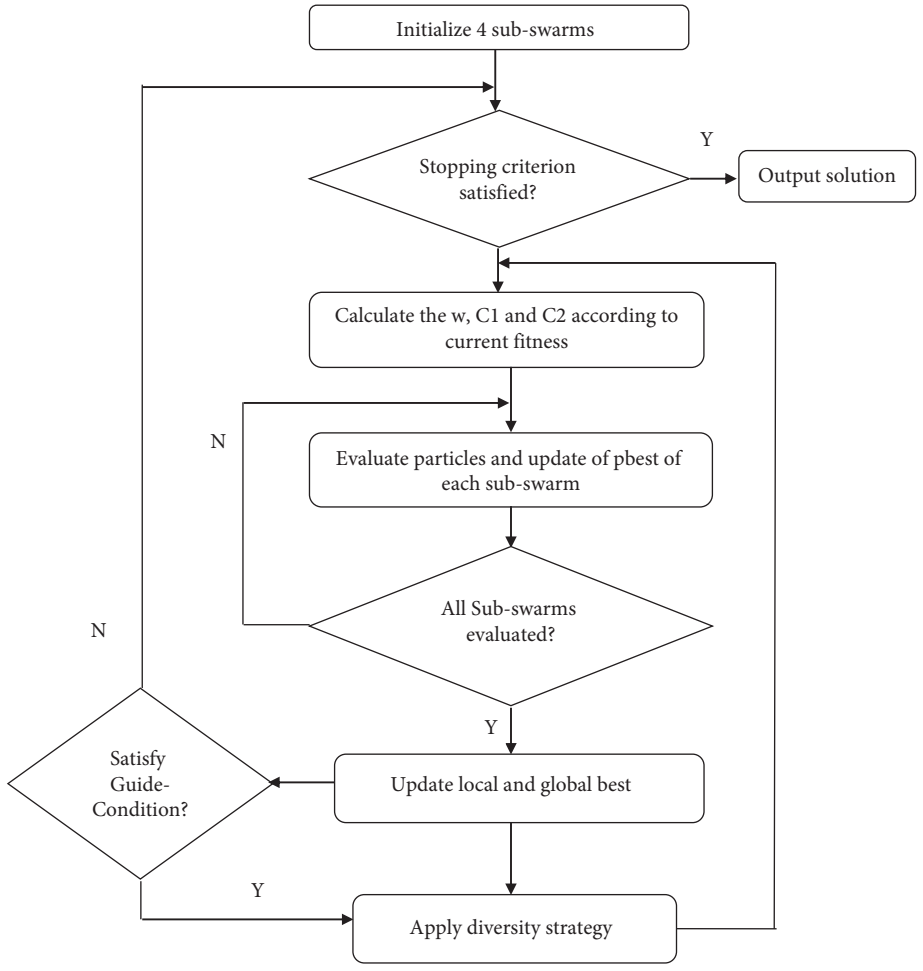


Figure 5: Organization chart of A-MsPSO.

```
Begin
Set the particle size of each subswarm
Initialize the positions, velocities, inertia weight, and acceleration factors
Determine each particle's shape worth
Find the Pbest in subswarms S1, S2, and S3 and the g_best in the population
Repeat until you hit the maximum number
  {
  Calculate the velocities v1, v2, v3, and v4 in subswarms S1, S2, S3, and S4
  Update the positions
  Evaluate the fitness of the i^th particle
  Update g_best
  If the guide condition is satisfied
  In each subswarm, use diversity guided convergence strategy.
  End if
  }
End do
Return g_best of the algorithm
```

Algorithm 1: The proposed algorithm (A-MsPSO).

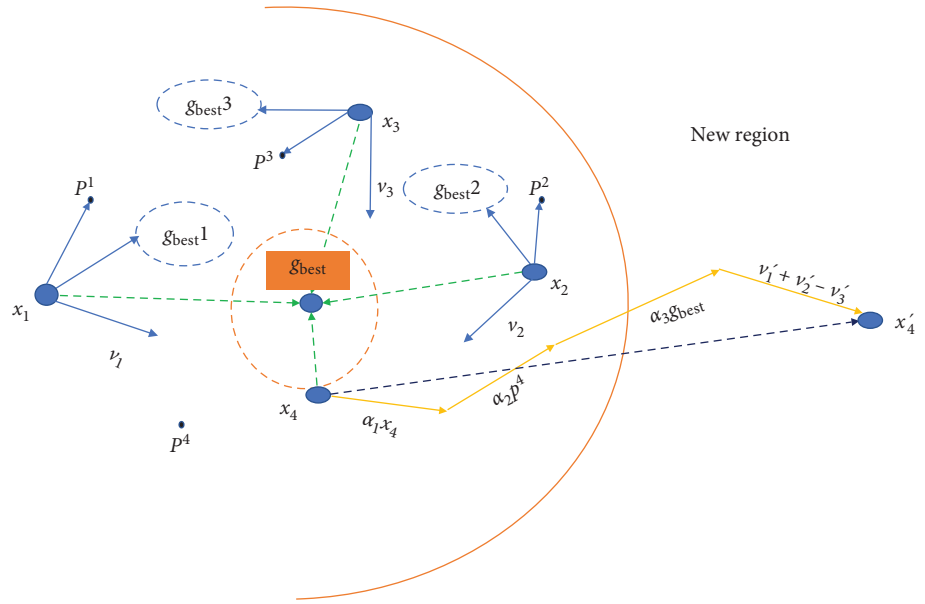


Figure 6: The cooperative evolutionary process in A-MsPSO.

$$v_i^{(3)}(t+1) = w_a\left[\frac{\gamma}{\gamma_1}v_i^{(1)}(t+1) + \frac{\gamma}{\gamma_2}v_i^{(2)}(t+1) + v_i^{(3)}(t)\right] + c_1 r_3\left[p_i^{(3)} - x_i^{(3)}(t)\right] + c_2 r_4\left[g_{best} - x_i^{(3)}(t)\right], \quad (9)$$

where r_3 and r_4 are randomly generated in $[0, 1]$.

The fitness values γ_1 and γ_2 of the particles in S_1 and S_2 are declared in the following equation:

$$\gamma = \gamma_1 + \gamma_2. \quad (10)$$

The velocity update equation in S_4 is written as follows:

$$v_i^{(4)}(t+1) = v_i^{(1)}(t+1) + v_i^{(2)}(t+1) - v_i^{(3)}(t+1). \quad (11)$$

The position of S_4 is updated by applying the following equation:

$$x_i^{(4)}(t+1) = \alpha_1 x_i^{(4)}(t) + \alpha_2 p_i^{(4)} + \alpha_3 g_{best} + v_i^{(4)}(t+1), \quad (12)$$

with

$$\alpha_1 + \alpha_2 + \alpha_3 = 1, \quad (13)$$

where

$$\alpha_1 = \frac{1}{6},$$
$$\alpha_2 = \frac{1}{3}, \quad (14)$$
$$\alpha_3 = \frac{1}{2}.$$

The impact factors α_1, α_2, and α_3 are employed to control the effects of the information on the position of the particle in S_4.

In A-MsPSO, information is shared by all particles in various areas. The basic subswarms (S_1 and S_2) influence the velocity and position information for other particles in other swarms. In S_3 with the current velocities and the fitness values, the particles follow the new search repertoire. Figure 4 depicts the cooperation between the subswarms.

In S_4, the update of the position and the speed are different from those in the other three subswarms and the particles adjust their speed according to the speeds and positions of S_1, S_2, and S_3. The rules of cooperation are defined to ensure the communication between the subswarms in the search process.

5. Particle Paths: Relation between the Particles Position and Velocity

We examine theoretically, in this section, the trajectories of the particles, the convergence of the A-MsPSO algorithm, and the speed of the particles of each subswarm. Convergence is defined based on the following limit:

$$\lim_{t \longrightarrow \infty} x_i(t) = P_i, \quad (15)$$

where x_i is the position at time t of the i^{th} particle and P_i corresponds to local or global optimum. The velocity and position update equations for the A-MsPSO with the adaptive inertia weight is applied as follows:

$$v(t+1) = w_a v(t) + c_1 r_1 \left(p_{\text{best}} - x(t)\right) + c_2 r_2 \left(g_{\text{best}} - x(t)\right),$$
$$x_i(t+1) = x_i(t) + v_i(t+1). \quad (16)$$

Authors, in [39], investigated the convergence of MsPSO and demonstrated that parameters setting affects the particle performance. The performed analysis shows that the relation between the particles in the 4 subswarms can be presented as a system to validate the convergence of the A-MsPSO algorithm.

Applying (2)–(11), we get the following A-MsPSO system:

$$x(t+1) = Fx(t) + \text{RE}, \quad (17)$$

where F is the system matrix and R is the constant matrix. $x(t) = [x_t^1, x_t^2, x_t^3, x_t^4, x_{t-1}^1, x_{t-1}^2, x_{t-1}^3]^T$ and $E = [p^{(1)}, p^{(2)}, p^{(3)}, p^{(4)}, g]^T$ is the vector consisting of individual optimums and the single global optimum.

The used symbols are written as follows:

$$\varphi_1 = v_{11} + v_{12} = c_1 r_{11} + c_2 r_{12},$$
$$\varphi_2 = v_{21} + v_{22} = c_1 r_{21} + c_2 r_{22}, \quad (18)$$
$$\varphi_3 = v_{31} + v_{32} = c_1 r_{31} + c_2 r_{32},$$

where r_{ij} is the random number in $[0, 1]$, $j = 1, 2$, and $i = 1, 2, 3$.

The equation of a particle position in S_1 is formulated as follows:

$$\begin{aligned} x_{t+1}^1 &= x_t^{(1)} + v_{t+1}^{(1)} \\ &= x_t^{(1)} + w_a v_t + v_{11}\left(p_t^{(1)} - x_t^{(1)}\right) + v_{12}\left(g_t - x_t^{(1)}\right) \\ &= x_t^{(1)} - v_{11}x_t^{(1)} - v_{12}x_t^{(1)} + w_a\left(x_t^{(1)} - x_{t-1}^{(1)}\right) + v_{11}p_t^{(1)} + v_{12}g_t \\ &= x_t^{(1)} - v_{11}x_t^{(1)} - v_{12}x_t^{(1)} + w_a x_t^{(1)} - w_a x_{t-1}^{(1)} + v_{11}p_t^{(1)} + v_{12}g_t \\ &= x_t^{(1)}\left(1 + w_a - \varphi_1\right) - w_a x_{t-1}^{(1)} + v_{11}p_t^{(1)} + v_{12}g_t. \end{aligned} \quad (19)$$

The equation of a particle position in S_2 is written as follows:

$$\begin{aligned} x_{t+1}^{(2)} &= x_t^{(2)} + v_{t+1}^{(2)} \\ &= x_t^{(2)} + w_a v_t + v_{21}\left(p_t^{(2)} - x_t^{(2)}\right) + v_{22}\left(g_t - x_t^{(2)}\right) \\ &= x_t^{(2)} - v_{21}x_t^{(2)} - v_{22}x_t^{(2)} + w_a\left(x_t^{(2)} - x_{t-1}^{(2)}\right) + v_{21}p_t^{(2)} + v_{22}g_t \\ &= x_t^{(1)} - v_{21}x_t^{(2)} - v_{22}x_t^{(2)} + w_a x_t^{(2)} - w_a x_{t-1}^{(2)} + v_{21}p_t^{1} + v_{22}g_t \\ &= x_t^{(2)}\left(1 + w_a - \varphi_2\right) - w_a x_{t-1}^{(2)} + v_{21}p_t^{(2)} + v_{22}g_t. \end{aligned} \quad (20)$$

In subswarm S_3, the equation applied to obtain the particle position is written as follows:

$$
\begin{aligned}
x_{t+1}^{(3)} &= x_t^{(3)} + v_{t+1}^{(3)} \\
&= x_t^{(3)} + \left[\frac{\gamma}{\gamma_1} v_{(t+1)}^{(1)} + \frac{\gamma}{\gamma_2} v_{(t+1)}^{(2)} + v_{(t)}^{(3)} + v_{31}\left[p_t^{(3)} - x_t^{(3)}\right] + v_{32}\left[g_t - x_{(t)}^{(3)}\right]\right. \\
&= x_t^{(3)} + w_a\left[\frac{\gamma}{\gamma_1}\left(x_t^{(1)}\left(w_a - \varphi_1\right) - w_a x_{t-1}^{(1)} + v_{11} p_t^{(1)} + v_{12} g_t\right)\right. \\
&\quad + \frac{\gamma}{\gamma_2}\left(x_t^{(2)}\left(w_a - \varphi_2\right) - w_a x_{t-1}^{(2)} + v_{21} p_t^{(2)} + v_{22} g_t\right) + v_{(t)}^{(3)} \\
&\quad + v_{31}\left[p_t^{(3)} - x_t^{(3)}\right] + v_{32}\left[g_t - x_{(t)}^{(3)}\right] \\
&= x_t^{(3)} + w_a\left(x_t^{(3)} - x_{t-1}^{(3)}\right) \\
&\quad + w_a\left[\frac{\gamma}{\gamma_1}\left(x_t^{(1)}\left(w_a - \varphi_1\right) - w_a x_{t-1}^{(1)} + v_{11} p_t^{(1)} + v_{12} g_t\right)\right. \\
&\quad + \frac{\gamma}{\gamma_2}\left(x_t^{(2)}\left(w_a - \varphi_2\right) - w_a x_{t-1}^{(2)} + v_{21} p_t^{(2)} + v_{22} g_t\right) \\
&\quad + v_{31} p_t^{(3)} - \varphi_3 x_{(t)}^{(3)} + v_{32} g_t \\
&= x_t^{(3)}\left(1 + w_a - \varphi_3\right) - w_a x_{t-1}^{(3)} + v_{31} p_t^{(3)} \\
&\quad + \frac{w_a \gamma\left(w_a - \varphi_1\right)}{\gamma_1} x_t^{(1)} - \frac{\left(w_a\right)^2 \gamma}{\gamma_1} x_{t-1}^{(1)} + \frac{w_a \gamma v_{11}}{\gamma_1} p_t^{(1)} \\
&\quad + \frac{w_a \gamma\left(w_a - \varphi_2\right)}{\gamma_2} x_t^{(2)} - \frac{\left(w_a\right)^2 \gamma}{\gamma_2} x_{t-1}^{(2)} + \frac{w_a \gamma v_{21}}{\gamma_2} p_t^{(2)} \\
&\quad + \left(\frac{\gamma w_a v_{22}}{\gamma_2} + \frac{\gamma w_a v_{12}}{\gamma_1} + v_{32}\right) g_t.
\end{aligned}
\tag{21}
$$

The equation of a particle position in S_4 is formulated as follows:

$$\begin{aligned}
x_{(t+1)}^{(4)} &= \alpha_1 x_{(t)}^{(4)} + \alpha_2 p_t^{(4)} + \alpha_3 g_t + v_{(t+1)}^{(4)} \\
&= \alpha_1 x_{(t)}^{(4)} + \alpha_2 p_t^{(4)} + \alpha_3 g_t + v_{(t+1)}^{(1)} + v_{(t+1)}^{(2)} - v_{(t+1)}^{(3)} \\
&= \alpha_1 x_{(t)}^{(4)} + \alpha_2 p_t^{(4)} + \alpha_3 g_t \\
&\quad + \left(1 - \frac{w_a \gamma}{\gamma_1}\right) v_{(1+t)}^{(1)} + \left(1 - \frac{w_a \gamma}{\gamma_2}\right) v_{(1+t)}^{(2)} \\
&\quad - (w_a - \varphi_3) x_{(t)}^{(3)} + w_a x_{(t-1)}^{(3)} - v_{31} p_t^{(3)} - v_{32} g_t \\
&= \alpha_1 x_{(t)}^{(4)} + \alpha_2 p_t^{(4)} - (w_a - \varphi_3) x_{(t)}^{(3)} + w_a x_{(t-1)}^{(3)} \\
&\quad - v_{31} p_t^{(3)} + \left(1 - \frac{w_a \gamma}{\gamma_1}\right)\left((w_a - \varphi_1) x_{(t)}^{(1)} - w_a x_{(t-1)}^{(1)} + v_{11} p_t^{(1)}\right) \\
&\quad + \left(1 - \frac{w_a \gamma}{\gamma_2}\right)\left((w_a - \varphi_2) x_{(t)}^{(2)} - w_a x_{(t-1)}^{(2)} + v_{21} p_t^{(2)}\right) \\
&\quad + \left[\left(1 - \frac{w_a \gamma}{\gamma_1}\right) v_{11} + \left(1 - \frac{w_a \gamma}{\gamma_2}\right) v_{22} - v_{32} + \alpha_3\right] g_t.
\end{aligned} \tag{22}$$

Then, A-MsPSO system is obtained as shown in (B_1).

According to the convergence analysis performed in [40], the particles of each subswarm converge to stable points defined by the limits given in the following equations:

$$\begin{aligned}
\lim_{t \longrightarrow +\infty} x^{(1)} &= \frac{c_1}{2} p^{(1)} + \frac{c_2}{2} g_{\text{best}}, \\
\lim_{t \longrightarrow +\infty} x^{(2)} &= \frac{c_1}{2} p^{(2)} + \frac{c_2}{2} g_{\text{best}}, \\
\lim_{t \longrightarrow +\infty} x^{(3)} &= w_a c_1 \left(p^{(1)} + p^{(2)}\right) + \frac{c_1}{2} p^{(3)} + g_{\text{best}} \left(2 w_a c_2 + \frac{c_2}{2}\right), \\
\lim_{t \longrightarrow +\infty} x^{(4)} &= \frac{(1 - w_a) c_1}{2} \left(p^{(1)} + p^{(2)}\right) - \frac{c_1}{2} p^{(3)} + \alpha_2 p^{(4)} + g_{\text{best}} \left((1 - 2 w_a) c_2 - \frac{c_2}{2} + \alpha_3\right).
\end{aligned} \tag{23}$$

6. Experimental Study

This section includes three parts. In the first part, we present the test functions used in the experimental study. However, the second part shows the good performance of A-MsPSO algorithm through computational experiments, while a nonlinear system is used, in the last part, to validate the suggested method performance.

6.1. Test Function and Parameter Settings. To test the efficiency of A-MSPSO from the benchmark functions available in the literature, we chose 4 unimodal functions (F_1, F_2, F_3, and F_4) and 11 multimodal functions (from F_5 to F_{15}) [41]. The formulas and the properties employed in these functions are shown in Table 1. The utilized experimental machine has a third-generation i3 processor of 2.5 GHz and 128 GB of storage capacity and the applied programming language is MATLAB.

6.2. Results and Discussions. In this subsection, we compare the introduced method with the most widely used inertia weight strategies, on the one hand, and with other versions of PSO, on the other hand. In subsection 6.2, several inertia weight strategies, such as random-MsPSO, constant-

Table 1: The main test functions.

#	Test function	Range	$f_{\min}$
F_1	Sphere model	$[-100, 100]^D$	−450
F_2	Schwefel's problem 2.22	$[-100, 100]^D$	−450
F_3	Schwefel's problem 1.2	$[-100, 100]^D$	−450
F_4	Schwefel's problem 2.21	$[-100, 100]^D$	−450
F_5	Generalized Rosenbrock's function	$[-100, 100]^D$	−310
F_6	Step function	$[-100, 100]^D$	390
F_7	Quartic function	$[0, 600]^D$	−180
F_8	Generalized Schwefel's problem 2.26	$[-32, 32]^D$	−140
F_9	Generalized Rastrigin's function	$[-5, 5]^D$	−330
F_{10}	Ackley's function	$[-5, 5]^D$	−330
F_{11}	Generalized Griewank function	$[-0.5, 0.5]^D$	90
F_{12}	Generalized penalized functions	$[-\pi, \pi]^D$	−460
F_{13}	Schwefel's function	$[-5, 5]^D$	−130
F_{14}	Shekel's foxholes function	$[-100, 100]^D$	−300
F_{15}	Kowalik's function	$[-100, 100]^D$	100

MsPSO, linear time-varying-MsPSO, and sigmoid-MsPSO, are analyzed on 15 test functions. In subsection 6.2, the A-MsPSO algorithm is compared to a number of PSO versions and to the basic MsPSO algorithm. Then, in subsection 6.2, the computational cost of the A-MsPSO algorithm is compared to that of the existing PSO variations. Finally, in subsection 6.2.1, the performance of this algorithm on Box and Jenkins gas furnace data is validated.

6.2.1. Comparisons of Inertia Weight Diversity. The most intensively used inertia weight strategies and the A-MsPSO algorithm are generally applied to solve benchmark problems. Table 2 lists the parameters used in each approach. All of the functions were tested on 30 dimensions. The maximum number of iterations was set to 2000, while the number of executions for each test was equal to 30. The performance of the algorithm was measured by the standard deviation and the mean values, as illustrated in Table 3. The values in bold indicate the best solution with a significance level of 5%. In this context, the A-MsPSO uses an adaptive parameter adjustment strategy based on the evolving state of each particle improving performance in terms of overall optimization and convergence velocity. This finding shows that the A-MsPSO has a better search capacity compared to the other search algorithms (random-MsPSO, constant-MsPSO, linear-time-varying-MsPSO, and sigmoid MsPSO) applied on 13 functions from F_1 to F_{15}, except F_8 and F_{15}.

In Figure 7, the vertical axis and the horizontal axis represent the best fitness and the number of iterations, respectively. Each curve demonstrates a shift in the shape of each algorithm based on a predetermined number of iterations. The final point of each curve represents the global optimum.

The proposed algorithm has significantly faster convergence when applied on the functions F_1, F_2, F_3, F_4, F_5, F_6, F_9, and F_{11} than when applied on the functions F_7, F_8, F_{10}, F_{12}, F_{13}, F_{14}, and F_{15}. However, the ability to jump out of the local optimum is improved. The curves in the functions F_7, F_{10}, F_{12}, F_{14}, and F_{15} remain stable after 200 iterations; this shows that the swarm is trapped to a local minimum, and the A-MsPSO method loses its global search capability.

6.2.2. Comparison of the Introduced Algorithm with Other PSO Algorithms. The proposed A-MsPSO was compared to six other variants of PSO including LW-PSO, CLWPSO, SIW-APSO, MSCPSO, NPSO, and standard MsPSO. The parameters of the algorithms are listed in Table 4 according to their references. The versions of PSO were applied to the 6 test functions (F_1, F_2, F_3, F_4, F_5, and F_6) with 30 dimensions. The maximum number of iterations was set to 1000 and 30 independent tests of each algorithm were performed. The results of standard deviation (STD) and mean values (Mean) are listed in Table 5 and the best are in bold. From this table, we notice that, on the 4 reference functions (F_1, F_2, F_3, and F_4), the results obtained by A-MsPSO are better than those provided by the other PSO variants. They show that the search method with an adaptive parameter adjustment strategy based on the evolutionary state of each and every particle is the most efficient in improving the PSO algorithm performance in solving optimization functions compared to the other.

6.2.3. Computational Cost. The CPU time was used to compare the computational efficiency of the PSO variants; the computational efficiency of each algorithm on the test function f was computed using the following formula:

$$H = \frac{T_e(f)}{T_{\text{tot}}(f)} \times 100\%, \tag{24}$$

where T_e is the time of computing the benchmark function algorithm f, and T_{tot} denotes the cumulative time of all the function algorithms f. Figure 8 shows the percentage of the total computational time by four PSO variants (A-MsPSO, S-MsPSO, R-MsPSO, and L-MsPSO) on fifteen benchmark functions. R-MsPSO is the most time consuming algorithm out of twelve reference functions and L-MsPSO is the fastest. The A-MsPSO algorithm ranks second out of twelve

Table 2: Description of the various inertia weight adjustment mechanisms.

Label	Reference	Inertia weight strategy	Adaptation mechanism
w_1	[18]	$w = 0.5 + \text{rand}/2$	Random inertia weight
w_2	[7]	$w = c$	Constant inertia weight
w_3	[41]	$w(i) = (\text{numiter} - i/\text{numiter})(w_{\min} - w_{\max}) + w_{\max}$	Linear time-varying
w_4	[42]	$w(i) = (w_{\min} - w_{\max})/1 + e^{-u\times(k-n\times\text{gen})} + w_{\max}$ $u = 10^{(\log(\text{gen})-2)}$	Sigmoid inertia weight
w_5	[43]	$w(i) = w_{\max} + (w_{\max} - w_{\min}) \times (e^{x(i)-1}/e^{x(i)+1})$	Adaptive inertia weight

Table 3: The results obtained for 15 test functions.

	Mean ± STD	Mean ± STD	Mean ± STD
Functions	F_1	F_2	F_3
Constant	$1.421e-81 \pm 3.966e-81$	$4.665e-43 \pm 8.523e-43$	$9.964e-84 \pm 2.315e-83$
Random	$2.477e-175 \pm 0.00e+00$	$6.038e-86 \pm 1.877e-85$	$1.141e-168 \pm 0.00e+00$
LTV	$1.683e-234 \pm 0.00e+00$	$9.813e-119 \pm 2.032e-118$	$6.820e-234 \pm 0.00e+00$
Sigmoid	$2.592e-82 \pm 8.077e-82$	$1.789e-42 \pm 4.530e-42$	$1.890e-83 \pm 4.247e-83$
A-MsPSO	**1.687e − 298 ± 0.0e + 00**	**4.697e − 179 ± 0.00e + 00**	**2.127e − 297 ± 0.00e + 00**
Functions	F_4	F_5	F_6
Constant	$3.119e-43 \pm 7.072e-43$	$2.892e+01 \pm 4.115e-02$	0 ± 0
Random	$6.813e-87 \pm 1.749e-86$	$2.890e+01 \pm 2.967e-02$	0 ± 0
LTV	$4.507e-117 \pm 1.265e-116$	$2.890e+01 \pm 4.076e-02$	0 ± 0
Sigmoid	$7.593e-44 \pm 7.062e-44$	$2.890e+01 \pm 3.698e-02$	0 ± 0
A-MsPSO	**1.010e − 177 ± 0.00e + 00**	**2.890e + 01 ± 1.932e − 02**	**0 ± 0**
Functions	F_7	F_8	F_9
Constant	$3.073e-04 \pm 1.779e-04$	$-11.521e-04 \pm 8.382e+04$	0 ± 0
Random	$1.265e-04 \pm 9.7369e-05$	$-10.030e-04 \pm 5.857e+04$	0 ± 0
LTV	$1.240e-04 \pm 9.4534e-05$	$-9.351e-04 \pm 4.951e+04$	0 ± 0
Sigmoid	$2.683e-04 \pm 1.9508-04$	$-11.474e-04 \pm 6.063e+04$	0 ± 0
A-MsPSO	$1.569e-04 \pm 1.4101-04$	$-7.952e-04 \pm 3.244e+04$	0 ± 0
Functions	F_{10}	F_{11}	F_{12}
Constant	$8.8818e-16 \pm 0.00e+00$	0 ± 0	$5.713e-01 \pm 1.417e-01$
Random	$8.8818e-16 \pm 0.00e+00$	$0 \pm 0.$	$3.990e-01 \pm 1.319e-01$
LTV	$8.8818e-16 \pm 0.00e+00$	0 ± 0	$3.051e-01 \pm 4.993e-02$
Sigmoid	$8.8818e-16 \pm 0.00e+00$	0 ± 0	$5.318e-01 \pm 2.313e-01$
A-MsPSO	**8.8818e − 16 ± 0**	**0 ± 0**	**5.573e − 01 ± 1.152e − 01**
Functions	F_{13}	F_{14}	F_{15}
Constant	$2.816e+00 \pm 2.195e-01$	$9.989e-01 \pm 2.220e-16$	$10.003e-04 \pm 3.200e-05$
Random	$2.219e+00 \pm 1.307e-01$	$9.989e-01 \pm 1.282e-16$	$8.105e-04 \pm 3.374e-04$
LTV	$2.761e+00 \pm 3.044e-01$	$9.989e-01 \pm 0.00e+00$	$8.103e-04 \pm 3.373e-04$
Sigmoid	$2.903e+00 \pm 1.866e-01$	$9.989e-01 \pm 2.866e-16$	$8.304e-04 \pm 3.497e-04$
A-MsPSO	**2.093e + 00 ± 2.251e − 03**	**1.494e + 00 ± 1.282ee − 16**	**10.006e − 04 ± 5.151e − 05**

functions, first in F_1 and third in F_{12} and F_{14}. The comparison of S-MsPSO and A-MsPSO demonstrates that the latter is the longest, on 13 reference functions, and the standard MsPSO is the fastest. This observation means that the computational complexity of introduced algorithm increases by using an adaptive inertia weight.

6.3. Box–Jenkins Gas Furnace Data. Box and Jenkins gas oven dataset is a benchmark widely used as a standard test in fuzzy modeling and identification techniques. It consists of 296 pairs of observations in/outs [u (k); y (k)] k from 1 to 296. Methane and air were combined to form a mixture of CO_2-containing gases. The input u (k) of the furnace corresponds to the flow of methane and the y (k) output is the concentration of CO_2 percentage in the outgoing gases.

Step 1: the experiment was conducted on 296 pairs of data. The population size in the four subswarms was set to 6, the number of iterations was 50, the rule number was 3, the acceleration coefficients were fixed at 1.494 (5), and the inertia weight was chosen as declared previously according to equations (7) and (8). The model obtained for the candidate characteristic variables is visualized in Figure 9. The A-MsPSO algorithm had a performance index of 0.106, which is the best compared to those of the other algorithms indicated in Table 6.

Step 2: the primary 148 data pairs were employed as training data to verify the robustness of A-MsPSO, while the remaining 148 data pairs were utilized as test data to predict the performance. The comparison of the performance of A-MsPSO and that of the real system is

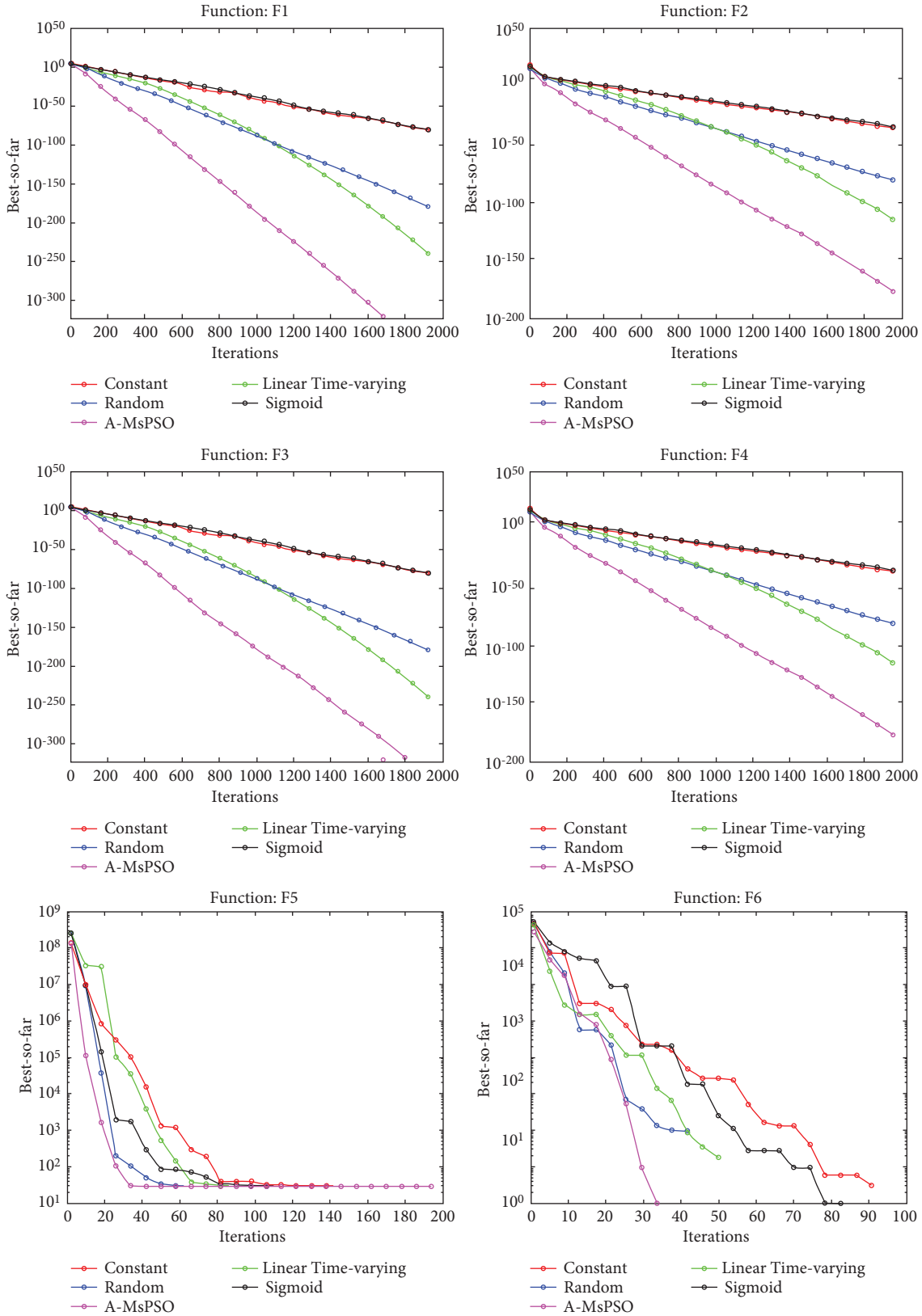


Figure 7: Continued.

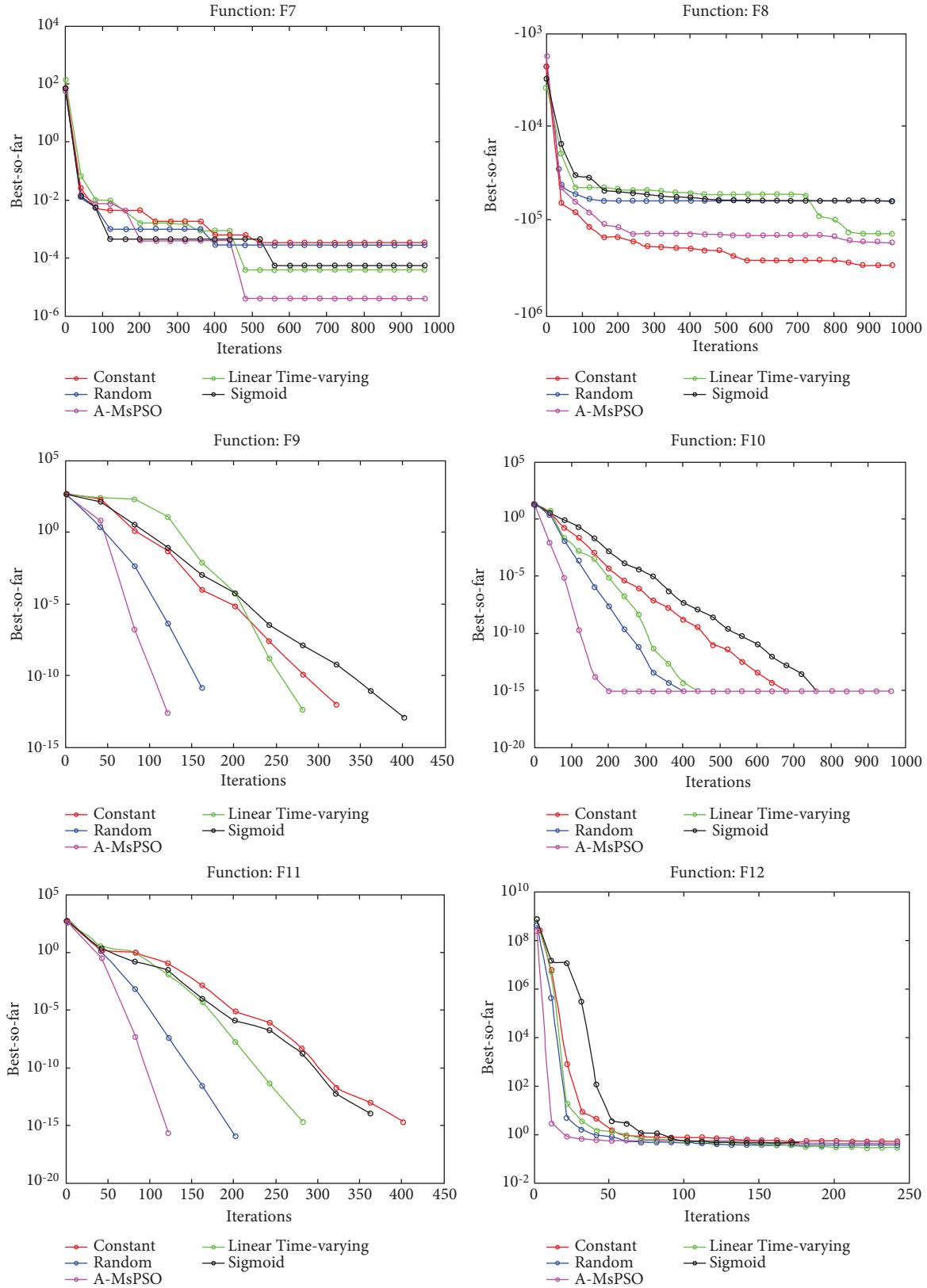


Figure 7: Continued.

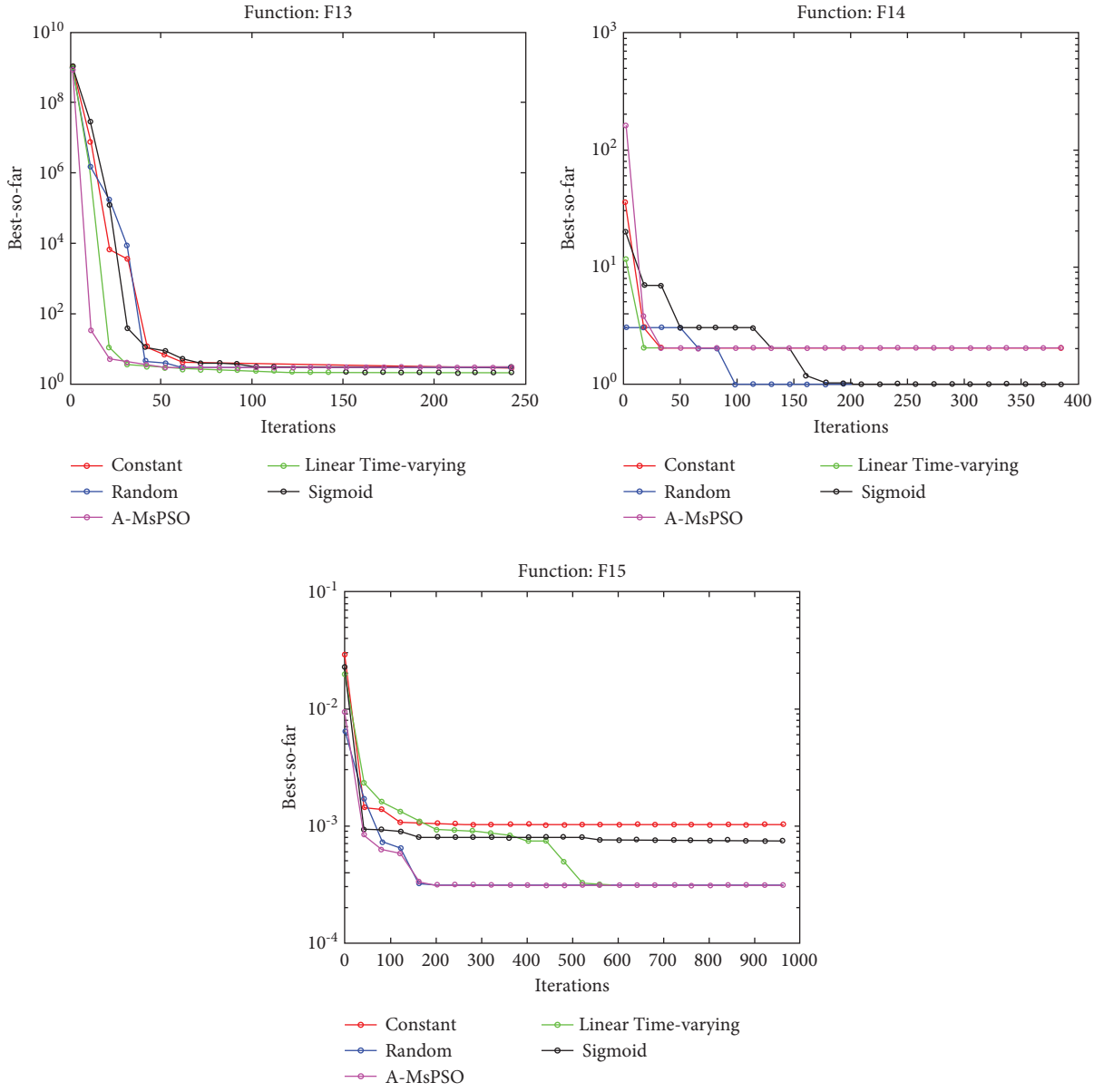


Figure 7: The algorithm's convergence curves for some problems.

Table 4: List of reference algorithms used in our comparison.

Algorithm	Name	Reference
LW-PSO	"PSO with linear inertia weight"	[7]
MSCPSO	"Multiswarm self-adaptive and cooperative particle swarm optimization"	[12]
MsPSO	"Multiswarm PSO"	[15]
SIW-APSO	"Self-inertia weight adaptive particle swarm optimization"	[44]
NPSO	"Novel particle swarm optimization algorithm for global optimization"	[45]
CLWPSO	"Comprehensive learning particle swarm optimizer"	[46]

presented in Figures 10 and 11, which display the proposed system approximation and generalization potential (A-MsPSO). The MSE values for the training and testing processes are 0.058 and 0.146, respectively, as shown in Table 7. Experimental results reveal that A-MsPSO has good precision and high powerful

Table 5: Comparison of mean and STD obtained by eight algorithms.

	Mean ± STD	Mean ± STD	Mean ± STD
Functions	F_1	F_2	F_3
LW-PSO	$7.67e-06 \pm 2.97e-06$	$7.30e-02 \pm 6.02e-02$	$6.83e-01 \pm 5.43e-01$
MSCPSO	$5.29e-23 \pm 3.69e-23$	$1.74e-012 \pm 3.76e-12$	$7.51e+00 \pm 1.26e+01$
MsPSO	$3.85e-104 \pm 8.07e-104$	$2.02e-54 \pm 6.01e-54$	$2.03e-102 \pm 6.43e-102$
SIW-APSO	$1.18e-146 \pm 2.65e-145$	$-3.87e+04 \pm 2.56e+01$	$2.56e-05 \pm 6.98e-04$
NPSO	$5.66e-141 \pm 9.8e-141$	$2.41e-129 \pm 4.170e-129$	$1.890e-83 \pm 4.247e-83$
CLWPSO	$4.46e-14 \pm 1.73e-14$	$2.10e+01 \pm 2.98e+00$	0 ± 0
A-MsPSO	**$2.59e-166 \pm 0.00e+00$**	**$4.39e-87 + 1.36e-86$**	**$8.086e-168 \pm 0.00e+00$**
Functions	F_4	F_5	F_6
LW-PSO	$1.93e-03 \pm 4.42e-04$	$1.69e-03 \pm 6.21e-04$	$1.19e-06 \pm 5.57e-07$
MSCPSO	$1.19e-10 \pm 4.79e-10$	0 ± 0	0 ± 0
MsPSO	$2.21e-54 \pm 5.76e-54$	$2.89e+01 \pm 3.63e-02$	0 ± 0
SIW-APSO	$1.16e-17 \pm 4.67e-13$	$2.96e-01 \pm 4.57e-01$	$2.32e-01 \pm 6.41e-01$
NPSO	0 ± 0	0 ± 0	$4.49e104 \pm 7.78e-104$
CLWPSO	$3.14e-10 \pm 4.64e-14$	$3.45e-07 \pm 1.94e-07$	$4.85e-10 \pm 3.63e-10$
A-MsPSO	**$5.96e-85 \pm 1.88e-84$**	$2.89e+01 \pm 1.80e-02$	$0.00e+00 \pm 0.00e+00$

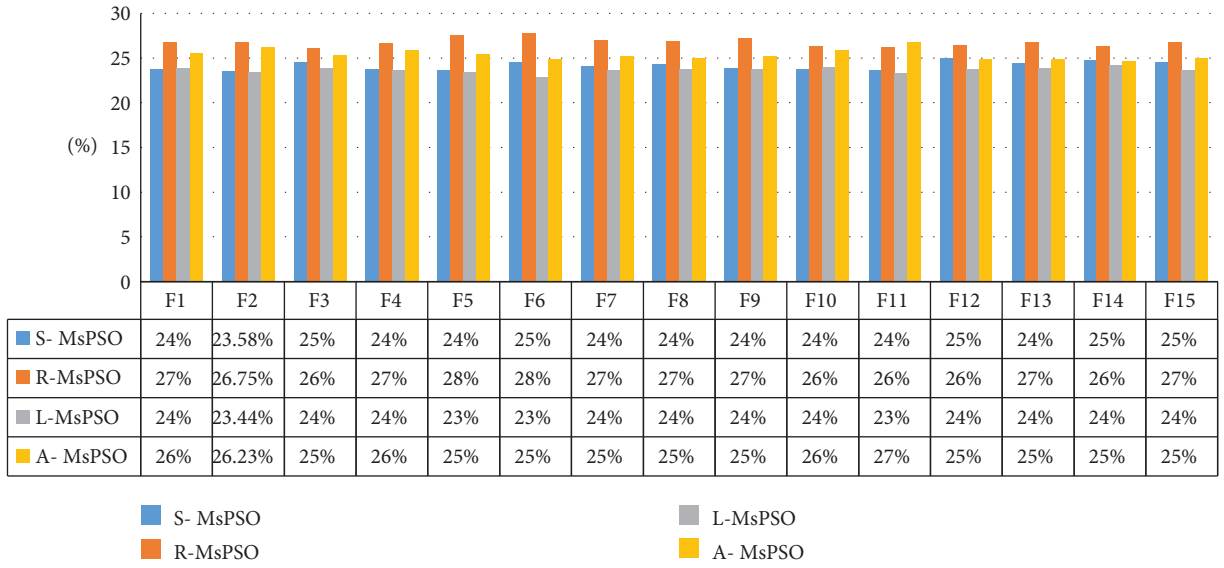
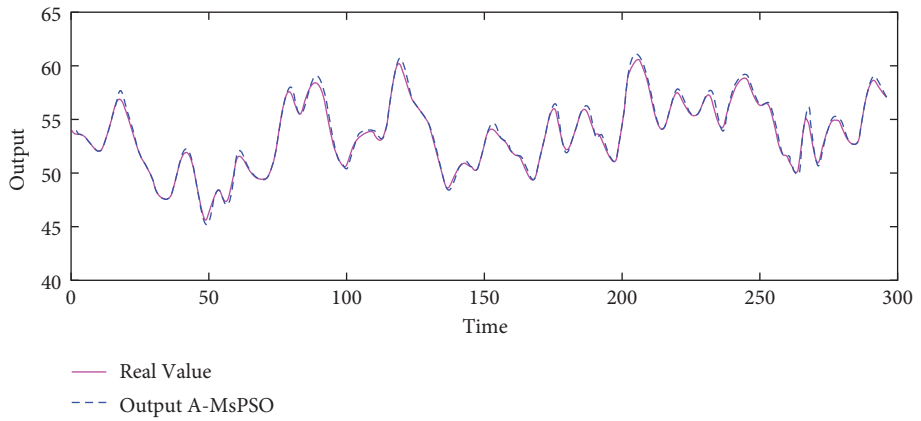
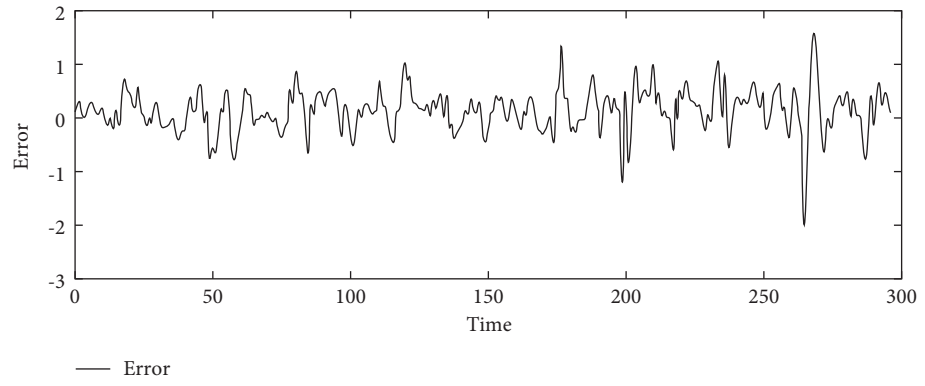


Figure 8: Comparison of the computational time with the fifteen benchmark functions from (F_1) to (F_{15}).



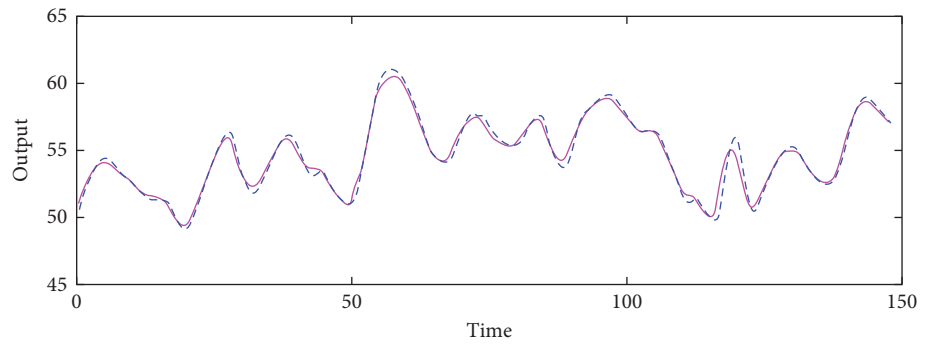
(a)

Figure 9: Continued.

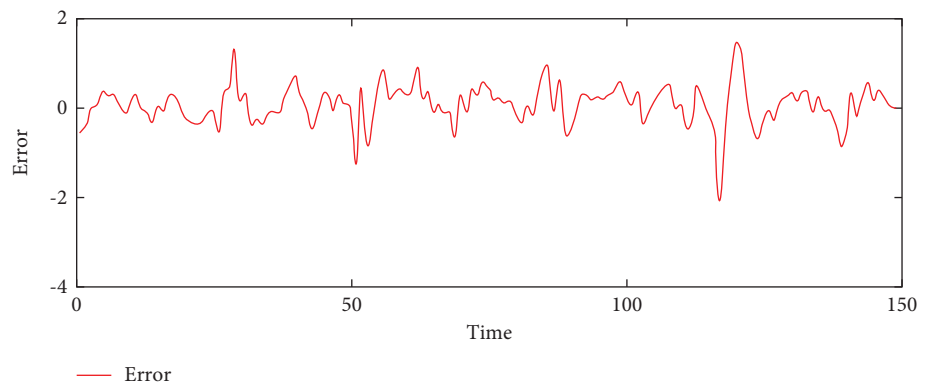


(b)

Figure 9: Modeling using 296 pairs of data from the Box–Jenkins gas furnace: (a) actual value and output from A-MsPSO and (b) error between A-MsPSO output and actual value.



(a)



(b)

Figure 10: Comparison of A-MsPSO and the real system over 148 testing data. (a) The A-MsPSO output and the real values. (b) Error between the A-MsPSO output and the real values.

Table 6: Step 1: different results on the Box–Jenkins gas furnace dataset.

Model	Reference	Number of inputs	Number of rules	MSE
Cheung et al. (2014)	[13]	2	3	0.110
Box and Jenkins (1990)	[47]	6	—	0.202
Sugeno and Yasukawa (1993)	[48]	3	6	0.190
Evsukoff et al. (2002)	[49]	2	90	0.090
Sugeno and Yasukawa (1993)	[50]	6	6	0.075
Tsekouras et al. (2005)	[51]	2	4	0.148
Bagis et al. (2008)	[52]	2	2	0.161
Zdiri et al. (2020)	[16]	**2**	**3**	**0.106**

The values in bold indicate the parameters and the results of our algorithm.

Table 7: Step 2: different results on the Box–Jenkins gas furnace dataset.

Model	Reference	Number of rules	MSE	
			Training data	Testing data
Cheung et al. (2014)	[13]	3	0.059	0.152
Lin and Cunningham (1995)	[53]	5	0.071	0.261
Kim et al. (1997)	[54]	6	0.034	0.244
Li et al. (2012)	[55]	3	0.0159	0.126
Tsekouras (2005)	[56]	6	0.022	0.236
Li et al. (2013)	[57]	3	0.015	0.147
Zdiri et al. (2020)	[16]	**3**	**0.058**	**0.146**

The values in bold indicate the parameter and the results of training data and testing data of our algorithm.

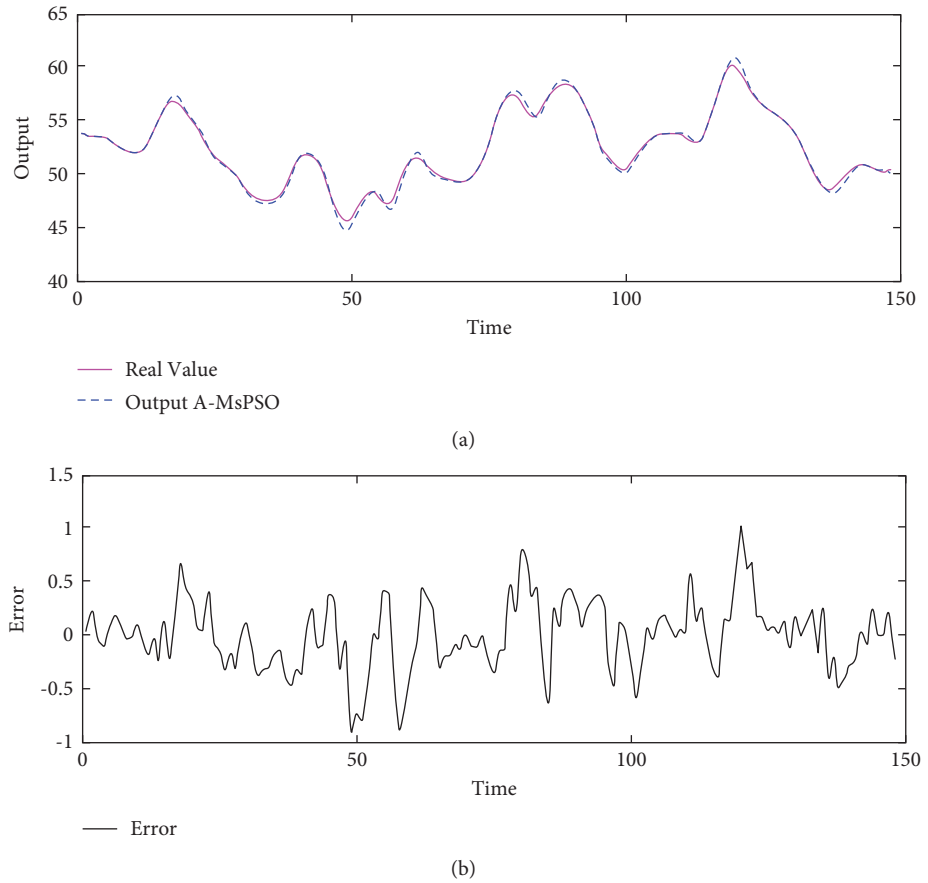


Figure 10: Comparison of A-MsPSO and the real system over 148 training data. (a) The A-MsPSO output and the real values. (b) Error between the A-MsPSO output and the real values.

generalization ability in system modeling of Box–Jenkins gas furnace dataset.

7. Future Work Possibilities

In future work, the proposed approach can be applied in several applications to (i) encrypt images using a 7D hyperchaotic map [30]; (ii) adjust the hyperparameters of a 4D chaotic map [58]; and (iii) optimize the parameters of 5D hyperchaotic map using A-MsPSO to perform encryption and decryption processes [29]; (iv) it may also be used in photovoltaic water pumping applications [59]. In addition, future research will focus on a thorough examination of A-MsPSO's applications in increasingly complicated practical optimization situations in order to deeply analyze the attributes and evaluate its performance.

8. Conclusion

An adaptive inertia weight approach was utilized in this paper to present a modified version of the multiswarm particle swarm optimization algorithm. Based on the comparison of four methods for establishing inertia weight strategies in the MsPSO algorithm, the ability of the A-MsPSO algorithm to optimize the issues with a bigger search space may be demonstrated by comparing experimental results in higher attributes. It was concluded that the MsPSO algorithm with adaptive inertia weight strategy is the best for greater accuracy, according to the results obtained by the performed tests. Theoretically, the four subswarms of A-MsPSO can also converge towards their own position of stable equilibrium. Furthermore, the experimental findings show that the suggested algorithm is capable of producing a robust model with improved generalization ability. We expect that this study will be extremely useful to researchers and will inspire them to develop good solutions to solve optimization problems using the A-MsPSO algorithm.

$$F = \begin{pmatrix} (1 + w_a - \varphi_1) & 0 & 0 & 0 & -w_a & 0 & 0 \\ 0 & (1 + w_a - \varphi_2) & 0 & 0 & 0 & -w_a & 0 \\ \dfrac{w_a\gamma(w_a - \varphi_1)}{\gamma_1} & \dfrac{w_a\gamma(w_a - \varphi_2)}{\gamma_2} & (1 + w_a - \varphi_3) & 0 & -\dfrac{(w_a)^2\gamma}{\gamma_1} & -\dfrac{(w_a)^2\gamma}{\gamma_2} & -w_a \\ \Gamma_a & \Gamma_b & (\varphi_3 - w_a) & \alpha_1 & \Gamma_c & \Gamma_d & -w_a \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{25}$$

with $\Gamma_a = (1 - (w_a\gamma/\gamma_1))(w_a - \varphi_1)$, $\Gamma_b = (1 - (w_a\gamma/\gamma_2))(w_a - \varphi_2)$, $\Gamma_c = -w_a(1 - (w_a\gamma/\gamma_1))$, and $\Gamma_d = -w_a(1 - (w_a\gamma/\gamma_2))$.

$$R = \begin{pmatrix} \nu_{11} & 0 & 0 & 0 & \nu_{12} \\ 0 & \nu_{21} & 0 & 0 & \nu_{22} \\ \frac{w_a\gamma}{\gamma_1}\nu_{11} & \frac{w_a\gamma}{\gamma_2}\nu_{21} & \nu_{31} & 0 & \frac{w_a\gamma}{\gamma_1}\nu_{12} + \frac{w_a\gamma}{\gamma_2}\nu_{22} + \nu_{32} \\ \frac{\gamma_1 - w_a\gamma}{\gamma_1}\nu_{11} & \frac{\gamma_2 - w_a\gamma}{\gamma_2}\nu_{21} & -\nu_{31} & \alpha_2 & \frac{\gamma_1 - w_a\gamma}{\gamma_1}\nu_{12} + \frac{\gamma_2 - w_a\gamma}{\gamma_2}\nu_{22} - \nu_{32} + \alpha_3 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{26}$$

$$\begin{pmatrix} x_{(t+1)}^{(1)} \\ x_{(t+1)}^{(2)} \\ x_{(t+1)}^{(3)} \\ x_{(t+1)}^{(4)} \\ x_{(t)}^{(1)} \\ x_{(t)}^{(2)} \\ x_{(t)}^{(3)} \end{pmatrix} = \begin{pmatrix} (1 + w_a - \varphi_1) & 0 & 0 & 0 & -w_a & 0 & 0 \\ 0 & (1 + w_a - \varphi_2) & 0 & 0 & 0 & -w_a & 0 \\ \frac{w_a\gamma(w_a - \varphi_1)}{\gamma_1} & \frac{w_a\gamma(w_a - \varphi_2)}{\gamma_2} & (1 + w_a - \varphi_3) & 0 & -\frac{(w_a)^2\gamma}{\gamma_1} & -\frac{(w_a)^2\gamma}{\gamma_2} & -w_a \\ \Gamma_a & \Gamma_b & (\varphi_3 - w_a) & \alpha_1 & \Gamma_a & \Gamma_d & -w_a \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_{(t)}^{(1)} \\ x_{(t)}^{(2)} \\ x_{(t)}^{(3)} \\ x_{(t)}^{(4)} \\ x_{(t-1)}^{(1)} \\ x_{(t-2)}^{(1)} \\ x_{(t-3)}^{(1)} \end{pmatrix} + \tag{27}$$

$$\begin{pmatrix} v_{11} & 0 & 0 & 0 & v_{12} \\ 0 & v_{21} & 0 & 0 & v_{22} \\ \frac{w_a\gamma}{\gamma_1}v_{11} & \frac{w_a\gamma}{\gamma_2}v_{21} & v_{31} & 0 & \frac{w_a\gamma}{\gamma_1}v_{12}+\frac{w_a\gamma}{\gamma_2}v_{22}+v_{32} \\ \frac{\gamma_1-w_a\gamma}{\gamma_1}v_{11} & \frac{\gamma_2-w_a\gamma}{\gamma_2}v_{21} & -v_{31} & \alpha_2 & \frac{\gamma_1-w_a\gamma}{\gamma_1}v_{12}+\frac{\gamma_2-w_a\gamma}{\gamma_2}v_{22}-v_{32}+\alpha_3 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} p_t^{(1)} \\ p_t^{(2)} \\ p_t^{(3)} \\ p_t^{(4)} \\ g_t \end{pmatrix} \text{(B1).} \tag{28}$$

Data Availability

The data used in this article are freely available upon request from the authors and for citing this paper in your manuscripts.

Conflicts of Interest

The authors declare no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Supplementary Materials

Our A-MsPSO algorithm and the full simulation results codes are available as supplementary file. The utilized machine has a third-generation i3 processor of 2.5 GHz and 128 GB of storage capacity and the applied programming language is MATLAB. (*Supplementary Materials*)

References

[1] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39–43, 1995.

[2] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the ICNN'95-international Conference on Neural Networks: 1942–1948*, Perth, WA, Australia, 1995.

[3] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," *IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, vol. 5, pp. 4104–4108, 1997.

[4] M.-S. Leu and M.-F. Yeh, "Grey particle swarm optimization," *Applied Soft Computing*, vol. 12, no. 9, pp. 2985–2996, 2012.

[5] Z. H. Zhan, J. Zhang, L. Yun, and H. S.-H. Chung, "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 6, pp. 1362–1381, 2009.

[6] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 240–255, 2004.

[7] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence*, pp. 69–73, Anchorage, AK, USA, 1998.

[8] Y. Shi and R. Eberhart, "Fuzzy adaptive particle swarm optimization," *Congress on Evolutionary Computation*, vol. 1, pp. 101–106, 2001.

[9] Y. Shi and R. Eberhart, "Particle swarm optimization: developments, applications and resources," *IEEE Congress on Evolutionary Computation*, vol. 1, pp. 81–86, 2001.

[10] J. Liang and P. Suganthan, "Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism," *IEEE International Conference on Evolutionary Computation*, pp. 9–16, 2006.

[11] G. G. Yen and W. F. Leong, "Dynamic multiple swarms in multiobjective particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 39, no. 4, pp. 890–911, 2009.

[12] J. Zhang and X. Ding, "A multi-swarm self-adaptive and cooperative particle swarm optimization," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 6, pp. 958–967, 2011.

[13] H. Li, H. Jin, H. Wang, and Y. Ma, "Improved adaptive Holonic particle swarm optimization," *Mathematical Problems in Engineering*, vol. 2019, Article ID 8164083, 2019.

[14] H. Zhang and Z. Yang, "Large-scale network plan optimization using improved particle swarm optimization

algorithm," *Mathematical Problems in Engineering*, vol. 2017, Article ID 3271969, 2017.

[15] J. Ngaam and Xue-Ming, "Optifel: a convergent heterogeneous particle swarm optimization algorithm for takagisugeno fuzzy modeling," *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 4, pp. 919–933, 2014.

[16] S. Zdiri and J. Chrouta, "Inertia weight strategies in multiswarm particle swarm optimization," in *Proceedings of the IEEE 4th International Conference on Advanced Systems and Emergent Technologies (IC_ASET)*, pp. 266–271, Hammamet, Tunisia, 2020.

[17] P. J. Angeline, "Using selection to improve particle swarm optimization," *IEEE congress on evolutionary computation*, vol. 84, no. 89, 1998.

[18] W. J. Zhang and X. F. Xie, "DEPSO: hybrid particle swarm with differentiai evolution operator," in *Proceedings of the IEEE Interenational Conference on Systems, Man and Cybernetics (SMCC) 3816–3821*, Washington, DC, USA, 2003.

[19] H. Liu and A. Abraham, "Fuzzy adaptive turbulent particle swarm optimization," in *Proceedings of the International Conference on Hybrid Intelligent Systems (HIS'05)*, Rio de Janeiro, Brazil, 2005.

[20] Y. Tan, "Particle swarm optimization algorithms inspired by immunity-clonal mechanism and their applications to spam detection," *International Journal of Swarm Intelligence Research*, vol. 1, no. 1, pp. 64–86, 2010.

[21] L. Xiao and X. Zuo, "Multi-DEPSO: a DE and PSO based hybrid algorithm in dynamic environments," *IEEE World Congress on Computational Intelligence*, 2012.

[22] J.-F. Chen, Q. Do, and H.-N. Hsieh, "Training artificial neural networks by a hybrid PSO-CS algorithm," *Algorithms*, vol. 8, no. 2, pp. 292–308, 2015.

[23] Y. H. Jia, J. Qiu, Z. Z. Ma, and F. F. Li, "A novel crow swarm optimization algorithm (CSO) coupling particle swarm optimization (PSO) and crow search algorithm (CSA)," *Computational Intelligence and Neuroscience*, vol. 2021, Article ID 6686826, 2021.

[24] J. Kennedy and R. Mendes, "The fully informed particle swarm: simpler, maybe better," *IEEE Congress on Evolutionary Computation*, vol. 8, no. 3, pp. 204–210, 2004.

[25] R. Mendes, "Population structure and particle swarm performance," *IEEE Congress on Evolutionary Computation*, pp. 1671–1676, 2002.

[26] S. S. Beheshti and R. Mendes, "Fusion global-local-topology particle swarm optimization for global optimization problems," *Mathematical Problems in Engineering*, vol. 2014, Article ID 907386, 2014.

[27] T. Peram, K. Veeramachaneni, and C. K. Mohan, "Fitness-distance-ratio based particle swarm optimization," *IEEE Swarm Intelligence Symposium*, vol. 1, no. 4, pp. 174–181, 2003.

[28] C. J. A. Bastos-Filho, D. F. Carvalho, E. M. N. Figueiredo, and P. B. C. de Miranda, "Dynamic clan particle swarm optimization," *Intelligent Systems Design and Applications*, pp. 249–254, 2009.

[29] M. Kaur and D. Singh, "Multiobjective evolutionary optimization techniques based hyperchaotic map and their applications in image encryption," *Multidimensional Systems and Signal Processing*, vol. 32, no. 1, pp. 281–301, 2021.

[30] M. Kaur, D. Singh, and V. Kumar, "Color image encryption using minimax differential evolution-based 7D hyper-chaotic map," *Applied Physics*, vol. 126, pp. B1–B19, 2020.

[31] M. Clerc, *L'optimisation Par Essaim Particulaire: Versions Paramétriques et Adaptatives*, Hermes science publications, 2005.

[32] Z.-H. Zhan, Z. Jun, L. Yun, and H. S.-H. Chung, "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 6, pp. 1362–1381, 2009.

[33] R. Eberhart and Y. Shi, "Tracking and optimizing dynamic systems with particle swarms," *IEEE Congress on Evolutionary Computation*, vol. 1, pp. 94–100, 2001.

[34] J. Xin, G. Chen, and Y. Hai, "A particle swarm optimizer with multi-stage linearly-decreasing inertia weight," *IEEE International Joint Conference on Computational Sciences and Optimization*, vol. 1, pp. 505–508, 2009.

[35] Y. Feng, G. F. Teng, A. Wang, and Y. M. Yao, "Chaotic inertia weight in particle swarm optimization," in *Proceedings of the Second International Conference on Innovative Computing, Information and Control*, p. 475, Chennai, India, 2007.

[36] J. Kordestani, A. Rezvanian, and Meybodi, "An efficient oscillating inertia weight of particle swarm optimisation for tracking optima in dynamic environments," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 28, no. 1–2, pp. 137–149, 2016.

[37] C. H. S. Zhan, Z. Jun, L. Yun, and H. S.-H. Chung, "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 6, pp. 1362–1381, 2009.

[38] M. S. Arumugam and M. V. C. Rao, "On the improved performances of the particle swarm optimization algorithms with adaptive parameters, cross-over operators and root mean square (RMS) variants for computing optimal control of a class of hybrid systems," *Applied Soft Computing*, vol. 8, no. 1, pp. 324–336, 2008.

[39] Y. L. Zheng, L. H. Ma, L. Y. Zhang, and J. X. Qian, "Empirical study of particle swarm optimizer with an increasing inertia weight," *Congress on Evolutionary Computation*, pp. 221–226, 2003.

[40] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[41] R. F. Malik, T. A. Rahman, S. Z. M. Hashim, and R. Ngah, "New particle swarm optimizer with sigmoid increasing inertia weight," *International Journal of Computer Science and Security*, vol. 1, no. 2, pp. 35–44, 2007.

[42] F. Vandenbergh and A. Engelbrecht, "A study of particle swarm optimization particle trajectories," *Information Sciences*, vol. 176, no. 8, pp. 937–971, 2006.

[43] H. Anandakumar and K. Umamaheswari, "A bio-inspired swarm intelligence technique for social aware cognitive radio handovers," *Computers and Electrical Engineering*, vol. 71, pp. 925–937, 2018.

[44] A. A. Nagra, F. Han, and Q. H. Ling, "An improved hybrid self-inertia weight adaptive particle swarm optimization algorithm with local search," *Engineering Optimization*, vol. 51, no. 7, pp. 1115–1132, 2019.

[45] C. F. Wang and K. Liu, "A novel particle swarm optimization algorithm for global optimization," *Computational Intelligence and Neuroscience*, vol. 2016, Article ID 9482073, 9 pages, 2016.

[46] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[47] J. Box, *Box and Jenkins: Time Series Analysis, Forecasting and Control*, Holden Day, San Francisco, CA, USA, 1990.
[48] M. Sugeno and T. Yasukawa, "A fuzzy-logic-based approach to qualitative modeling," *IEEE Transactions on Fuzzy Systems*, vol. 1, no. 1, pp. 7–31, 1993.
[49] A. Evsukoff, A. Branco, and S. Galichet, "Structure identification and parameter optimization for non-linear fuzzy modeling," *Fuzzy Sets and Systems*, vol. 132, no. 2, pp. 173–188, 2002.
[50] M. Sugeno and T. Yasukawa, "A fuzzy-logic-based approach to qualitative modeling," *IEEE Transactions on Fuzzy Systems*, vol. 1, no. 1, pp. 7–31, 1993.
[51] G. E. Tsekouras, "On the use of the weighted fuzzy c-means in fuzzy modeling," *Advances in Engineering Software*, vol. 36, no. 5, pp. 287–300, 2005.
[52] A. Bagis, "Fuzzy rule base design using tabu search algorithm for nonlinear system modeling," *ISA Transactions*, vol. 47, no. 1, pp. 32–44, 2008.
[53] Y. Lin and G. A. Cunningham, "A new approach to fuzzy-neural system modeling," *IEEE Transactions on Fuzzy Systems*, vol. 3, no. 2, pp. 190–198, 1995.
[54] E. Kim, M. Park, S. Kim, and M. Mignon Park, "A transformed input-domain approach to fuzzy modeling," *IEEE Transactions on Fuzzy Systems*, vol. 6, no. 4, pp. 596–604, 1998.
[55] C. Li, J. Zhou, B. Fu, P. Kou, and J. Xiao, "T–S fuzzy model identification with a gravitational search-based hyperplane clustering algorithm," *IEEE Transactions on Fuzzy Systems*, vol. 20, no. 2, pp. 305–317, 2011.
[56] G. E. Tsekouras, "On the use of the weighted fuzzy c-means in fuzzy modeling," *Advances in Engineering Software*, vol. 36, no. 5, pp. 287–300, 2005.
[57] C. Li, J. Zhou, J. Xiao, and H. Xiao, "Hydraulic turbine governing system identification using T-S fuzzy model optimized by chaotic gravitational search algorithm," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 9, pp. 2073–2082, 2013.
[58] M. Kaur, D. Singh, and R. Singh Uppal, "Parallel strength Pareto evolutionary algorithm-II based image encryption," *IET Image Processing*, vol. 14, no. 6, pp. 1015–1026, 2020.
[59] N. Priyadarshi, M. S. Bhaskar, S. Padmanaban, F. Blaabjerg, and F. Azam, "New CUK-SEPIC converter based photovoltaic power system with hybrid GSA-PSO algorithm employing MPPT for water pumping applications," *IET Power Electronics*, vol. 13, no. 13, pp. 2824–2830, 2020.