mathematics

MDPI

Article

Competitive Coevolution-Based Improved Phasor Particle Swarm Optimization Algorithm for Solving Continuous Problems

Omer Ali [1], Qamar Abbas [1], Khalid Mahmood [2], Ernesto Bautista Thompson [3,4,5], Jon Arambarri [3,6,7] and Imran Ashraf [8,*]

1 Department of CS, International Islamic University Islamabad, Islamabad 44000, Pakistan; omer.phdcs200@iiu.edu.pk (O.A.); qamar.abbas@iiu.edu.pk (Q.A.)
2 Institute of Computing and Information Technology, Gomal University, Dera Ismail Khan 29220, Pakistan; khalid@gu.edu.pk
3 Higher Polytechnic School, Universidad Europea del Atlántico, Isabel Torres 21, 39011 Santander, Spain; ernesto.bautista@unini.edu.mx (E.B.T.); jon.arambarri@uneatlantico.es (J.A.)
4 Department of Project Management, Universidad Internacional Iberoamericana, Campeche 24560, Mexico
5 Fundación Universitaria Internacional de Colombia, Bogotá, Colombia
6 Universidad Internacional Iberoamericana, Arecibo, PR 00613, USA
7 Universidade Internacional do Cuanza, Cuito, Bié, Angola
8 Department of Information and Communication Engineering, Yeungnam University, Gyeongsan 38541, Republic of Korea
* Correspondence: imranashraf@ynu.ac.kr

Abstract: Particle swarm optimization (PSO) is a population-based heuristic algorithm that is widely used for optimization problems. Phasor PSO (PPSO), an extension of PSO, uses the phase angle θ to create a more balanced PSO due to its increased ability to adjust the environment without parameters like the inertia weight w. The PPSO algorithm performs well for small-sized populations but needs improvements for large populations in the case of rapidly growing complex problems and dimensions. This study introduces a competitive coevolution process to enhance the capability of PPSO for global optimization problems. Competitive coevolution disintegrates the problem into multiple sub-problems, and these sub-swarms coevolve for a better solution. The best solution is selected and replaced with the current sub-swarm for the next competition. This process increases population diversity, reduces premature convergence, and increases the memory efficiency of PPSO. Simulation results using PPSO, fuzzy-dominance-based many-objective particle swarm optimization (FMPSO), and improved competitive multi-swarm PPSO (ICPPSO) are generated to assess the convergence power of the proposed algorithm. The experimental results show that ICPPSO achieves a dominating performance. The ICPPSO results for the average fitness show average improvements of 15%, 20%, 30%, and 35% over PPSO and FMPSO. The Wilcoxon statistical significance test also confirms a significant difference in the performance of the ICPPSO, PPSO, and FMPSO algorithms at a 0.05 significance level.

Keywords: particle swarm optimization; phasor PSO; PSO coevolution; optimization; multi-swarm

MSC: 68W50

Citation: Ali, O.; Abbas, Q.; Mahmood, K.; Bautista Thompson, E.; Arambarri, J.; Ashraf, I. Competitive Coevolution-Based Improved Phasor Particle Swarm Optimization Algorithm for Solving Continuous Problems. *Mathematics* **2023**, *11*, 4406. https://doi.org/10.3390/math11214406

Academic Editor: Jian Dong

Received: 18 September 2023
Revised: 15 October 2023
Accepted: 20 October 2023
Published: 24 October 2023

1. Introduction

Particle swarm optimization (PSO) was initially introduced by Kennedy and Eberhart [1]. PSO is a simple stochastic searching technique for optimization that is motivated by the ordinary swarming comportment of bird clustering and fish schooling. The performance of PSO in finding virtuous solutions for optimization problems is very good. The PSO algorithm has the advantage of fast convergence, easy code implementation, less

computational complexity, and few parameter adjustments [2,3]. Instead of using only the fittest particle, PSO uses all particles of the population for computation due to its social behavior. All particles update their positions as determined by their individual best position (*pbest*) and overall best position (*gbest*).

The productivity of PSO is assessed by the minimum number of iterations needed to find an optimum solution with the indicated accuracy and minimal computation. The performance of the PSO algorithm highly depends on the selection of parameters. In PSO parametric modification, PSO parameters are adjusted to improve the convergence and exploration capabilities [4]. Parameter adjustments include modification of the inertia weight, cognitive factor, social factor, techniques for defining the own best *pbest* and overall best *gbest*, and different prototypes for the velocity update. The inertia weight w was introduced by Bansal et al. [5] to maintain and stabilize a broader scope of the search. To improve PSO using the inertia weight, different strategies like the growing inertia weight [5], falling inertia weight, and adaptive inertia weight are used.

Numerical optimization problems are important in computing and are commonly solved using evolutionary computing algorithms such as PSO and differential evolution [6]. When considering optimization problems, PSO has been successfully applied in both the continuous and discrete domains [7]. Among several discrete PSO variations, binary PSO [7] is possibly the most well-known model, and it has been applied to many problems, e.g., job-shop scheduling [8]. PSO is a heuristic algorithm like a genetic algorithm; however, it is computationally less expensive [9]. PSO parameters depend on specific applications and are adaptive according to the application. In multi-objective optimization problems, the PSO algorithm with multiple sub-populations has been used to achieve prominent results [10].

PSO has been efficiently applied in many real-world problems. Most real-world problems have increasing complexity, so the proficiency and effectiveness of PSO need to be continuously improved. Despite the many advantages of PSO, there are still certain research gaps that require attention. These research gaps include early convergence, memory efficiency, slow convergence toward the global optimum, PSO without parameters, and slow computational speed for large populations. Many variants of PSO have been successfully developed to handle large populations, where the population is divided into multiple sub-swarms, and each particle maintains the information of the local best. In the cooperative approach, the whole population is divided into many sub-swarms. Each sub-swarm coevolves with the others to form a complete solution. In competitive coevolution, the population is divided into many sub-swarms, and two sub-swarms are selected to compete for coevolution, with the swarm with the best fitness earning the right to represent itself.

Efficient and effective information sharing between sub-swarms is also an important research area, where each sub-swarm shares its individual best fitness with the others. To enhance performance and population diversity, a competitive coevolution process is applied to the sub-swarms. This improves the performance and population diversity of algorithms. Phasor PSO (PPSO) provides efficient results compared to other PSO variants in many multidimensional optimization problems. Introducing the phase angle in PPSO enhances the effectiveness and adaptability of the algorithm. However, there is still a need to modify PPSO to solve global optimization problems. In the case of a small population, PPSO achieves effective results but in a large population, PPSO needs to efficiently handle a large number of particles.

PPSO lacks population diversity due to a lack of competitive processes during evolution. This reduced diversity results in the degradation of the performance of PPSO due to stagnation in local optima. Diversity is incorporated by employing the competitive coevolution process, where the fitness of individuals is estimated following the exchange of information with individuals from other sub-populations.

1.1. Research Motivation

PPSO lacks population diversity due to a lack of competitive processes during the evolution of large-sized populations. Consequently, the results become trapped in local optima. This diversity issue reduces the convergence speed of the PPSO algorithm and ultimately reduces its performance. The inclusion of a competitive process for exchanging information during the evolutionary process of PPSO helps achieve convergence more quickly without compromising on performance.

1.2. Problem Statement

In the case of a large population, the PPSO algorithm suffers from a slow convergence speed due to the large number of particles. The selection of a suitable technique for dividing the population into multiple sub-swarms is crucial for PPSO. A large population size requires a large memory size to manage operations. PPSO spends more time and computational load in managing memory for a large population, leading to premature convergence within the population, which results in increased time complexity. PPSO requires interaction among all individual algorithms to reduce both the space and time complexity of the algorithm. The best individuals who share their values with the population help increase the performance of the algorithm.

1.3. Research Significance

The incorporation of a competitive coevolution process in PPSO helps improve the performance of the PPSO algorithm. In competitive coevolution, the process improves population diversity and the performance of the algorithm. The coevolutionary process makes PPSO more memory efficient and adaptable, resulting in a higher efficacy rate. The increased complexity in terms of dimensions, number of particles, and population range shows that the improved competitive multi-swarm phasor PSO (ICPPSO) performs better.

1.4. Research Contributions

This study makes the following contributions:

- A cooperative coevolution concept is incorporated into PPSO, which helps enhance diversity in the population and avoid local optima issues.
- The hybridization of PPSO with the competitive coevolution method helps enhance the performance of the PPSO algorithm.
- The experimental results of the average fitness performance metric and the results of the proposed algorithm are presented by using six standard benchmark functions.

The rest of this study is divided into six sections. The operation of PSO and its parameters are described in Section 2. Section 3 discusses the existing literature and PSO variants. The operation of PSSO and its related challenges are presented in Section 4. Section 5 elaborates on the proposed approach. The experimental results and discussions are provided in Section 6. Finally, the conclusions are presented in Section 7.

2. Operation of PSO and Its Parameters

PSO is regarded as a well-organized population and resistor parameter-based procedure for the universal optimization of different problems. PSO algorithms and GAs share several common aspects. The initialization of the population with a random solution and the subsequent generation updates to search for optima are almost the same. Genetic operators, like crossover and mutation operators, are not used in PSO because of its social behavior. So, in PSO, each particle progresses by cooperating and competing with other individuals [11]. Each particle modifies its movement according to its own best-reached position and the overall best position among all particles.

In PSO, every probable solution is characterized as a particle. Each particle has certain parameters that continuously adjust to update its position. These parameters include the current position, speed of the particle, and best position found by the particle thus far. At the beginning of PSO, all particles in the population are initialized with a random position, and their velocity is set to 0. In PSO, every particle in the dimensional space D is treated as a point. X_I represents the position of the i^{th} particle and calculates the existing quality of each particle. Each particle in the population updates its velocity V to evaluate the tracking of its movement at t iterations, where $X_I^T = (xi_1, xi_2, xi_3, \ldots, xi_D)$ signifies the position of particles and $V_I^T = (vi_1, vi_2, vi_3, \ldots, vi_D)$ represents the velocity of particles for the d^{th} dimension $(d = 1, 2, \ldots, D)$. These parameters are updated using Equations (1) and (2).

$$V_{id}^{k+1} = v_{id}^{k+1} + c_1 \times r_1 \times (pbest_{id}^k - x_{id}^k) + c_2 \times r_2 \times (gbest_{id}^k - x_{id}^k) \tag{1}$$

$$X_{id}^{k+1} = X_{id}^k + V_{id}^{k+1} \tag{2}$$

where c_1 and c_2 are acceleration controller coefficients, r_1, and r_2 are randomly generated coefficients within the range of (0, 1), X is the position of a particle, and k represents the current iteration. The range $[-v_{max}, v_{max}]$ is defined for the velocity V_i to stop the particle from moving outside the problem exploration area. *Pbest* is computed through a cognitive learning approach, in which the best solution thus far from the current particle is stored. *Gbest* is computed through a social learning approach, in which the best solution thus far from every particle in the population is stored. *Pbest* and *Gbest* are calculated using Equations (3) and (4).

$$\left.\begin{array}{l} pbest_{id}^{k+1} = pbest_{id}^k iff(pbest_{id}^k) \leq f(X_{id}^{k+1}) \\ pbest_{id}^{k+1} = X_{id}, Otherwise \end{array}\right\} \tag{3}$$

$$\left.\begin{array}{l} Gbest_{id}^{k+1} = pbest_{id}^k IFf(pbest_{id}^{k+1}) \leq f(Gbest_{id}^k) \\ Gbest_{id}^{k+1} = Gbest_{id}^k, Otherwise \end{array}\right\} \tag{4}$$

In Equation (3), *Pbest* is selected using the greedy criteria based on the fitness of the new position and the current position of the particle. Equation (3) is used to update the *Pbest* of each population member in each iteration. *Gbest* in Equation (4) represents the best particle overall.

2.1. Inertia Weight

The success of the optimization algorithm is analytically dependent on accurate stability among local and global searches during the progress of all iterations. To maintain this balance between local and overall searches, Shi et al. [12] proposed a fresh parameter w for the velocity update equation, called the inertia weight. The velocity update Equation (5) is expressed as

$$V_{id}^{k+1} = \omega \times v_{id}^k + c_1 \times r_1 \times (pbest_{id}^k - x_{id}^k) + c_2 \times r_2 \times (gbest_{id}^k - x_{id}^k) \tag{5}$$

where V_{id} is the velocity, and ω is the inertia weight.

Equation (5) is used to update the velocity for the updating of the particle position of the population of the swarm or sub-swarm. If the cost of inertia is high, particles have a higher probability of traveling to new search areas. However, if the cost of inertia is small, the probability of exploring the search area is reduced, resulting in fewer updates to the particles' velocity. At the initial value of inertia, w remains constant at 0.4 throughout the search process, but later, researchers employed different strategies and made changes to the inertia weight. These strategies can be classified into three types. To determine the optimum value in dynamic locations, Eberhart and Shi [13] effectively used the random cost of the inertia weight. The second type is a time-varying inertia weight approach, in which the inertia weight differs with the number of iterations over time. A linearly decreasing inertia approach was introduced by Lei et al. [14], which effectively contributed

to the positive modification of PSO characteristics. Similarly, other linear methods [15] and nonlinear approaches [16] have proven to be effective inertia weight approaches. Arumugam et al. [17] introduced a method where the inertia weight is estimated from the proportion of the *Gbest* fitness and mean of the best finesses in every iteration. The selection of the inertia weight strategy is always problem-dependent.

2.2. Cognitive and Social Learning Coefficients

The cognitive and social learning coefficients also play an important role in bringing stability to local and global searches. In Equation (5), $c1$ is the rational coefficient and c_2 is the social learning component. Equation (6) represents the cognitive part, and Equation (7) represents the communal learning part. Initially, the values of c_1 and c_2 were equal and set to 2.0 by Shi et al. [4]. The cognitive component c_1 controls the footstep size of particles taken toward the personal best solution, and the social coefficient c_2 controls the footstep size taken toward the global best solution. The social component c_2 has a greater impact on improving convergence compared to c_1. Time-varying cognitive coefficients were implemented by Cai et al. [18], which focus on convergence speed in the first phase, and the second phase focuses on global search competency. Large values of the cognitive coefficient c_1 and social coefficient c_2 equate to an enormous local search ability, and small values of c_1 and the social coefficient c_2 equate to an enormous global search space. In Equation (6), r_1 and r_2 are randomly generated numbers, CP is the cognitive part, and SP is the social part of the velocity.

$$CP = c_1 \times r_1 \times (pbest_{id}^k - x_{id}^k) \tag{6}$$

Equation (6) is used to calculate the relative contribution of the cognitive part by utilizing the *pbest* position of the current individual.

$$SP = c_1 \times r_1 \times (gbest_{id}^k - x_{id}^k) \tag{7}$$

The social part in the velocity is calculated using Equation (7), which focuses on the contribution of the *gbest* individual.

2.3. Pseudocode of PSO Algorithm

In the Algorithm 1 of PSO, all particles X of the population are initialized with random spots, and the fitness of each particle is computed. Then, the velocity V is updated using the velocity update equation, followed by the position update of particle X. *Pbest* and *Gbest* are updated for each particle in the population. This process is repeated unless the termination criteria are reached.

Algorithm 1 Pseudocode for the PSO algorithm.

Require: X is an individual, NP is the population size, V is the velocity, *Pbest* is the personal best position, *Gbest* is the personal best position, c_1 and c_2 are the social and local amplification factors, r_1 and r_2 are two random numbers, and k is the iteration number.

Ensure: Evolved population with *Gbest* at the optimal position to achieve optimal fitness values of the problem.

```
for whole population do
    Initialization of all particles
end for
while maximum iterations not reached do
    for each particle do
        Calculate the fitness of particles
        if fitness value is better than the best fitness value (Pbest) in history then
            Set current value as the new Pbest
        end if
```

Algorithm 1 *Cont.*

Select the particle with the best fitness value of all the particles as the *Gbest*
end for
for each particle **do**
Calculate particle velocity according to the given equation
$V_{id}^{k+1} = \omega \times v_{id}^{k} + c_1 \times r_1 \times (pbest_{id}^{k} - x_{id}^{k}) + c_2 \times r_2 \times (gbest_{id}^{k} - x_{id}^{k})$
Update particle position according to Equation (2)
$X_{id}^{k+1} = x_{id}^{k} + V_{id}^{k+1}$
end for
end while

3. Particle Swarm Optimization Variants and Literature Review

The particle swarm optimization algorithm is a well-known evolutionary computing optimization that is known to be very effective for solving real-world optimization problems [19]. Many variations of PSO algorithms have been presented, including various swarms, fresh effective learning policies, diversity-retaining strategies, and hybrid algorithms, to resolve several optimization complications. These enhancements of PSO include the population initialization process, adjustment of parameters (inertia weight, coefficients, *pbest*, *gebest*), sub-swarm techniques for large-scale optimization, and hybrid methods with other algorithms. For population enhancement, different techniques can be used.

PSO-sono was introduced by Meng et al. [20] for numerical optimization problems. They presented a hybrid paradigm based on the sorted swarm, adaptation schemes for the constriction coefficient and the paradigm ratio, and fully informed variations of the PSO algorithm. The experimental results showed the competitiveness of the presented enhancement of the existing PSO for standard benchmark functions. In terms of modified PSO, the concept of distance-based enhancement was presented by Lazzus et al. in [21]. A random, guided new direction helped improve the search capability of the PSO algorithm. The presented method showed significant improvements compared to standard PSO using benchmark optimization functions.

Nabi et al. [22] introduced task scheduling-based adaptive PSO in their research work to balance loads in cloud computing. They adaptively updated the inertia weight to update the velocity of the PSO algorithm using an adaptive linear decreasing approach. The research results demonstrated significant improvements compared to five other inertia-weight techniques used in the PSO algorithm. The concept of an optimal control parameter in PSO was introduced by Eltamaly in his research work on energy systems [23]. The author used two nested PSOs to optimize the control parameters, where the inner PSO was used as a fitness function for the outer PSO. The experimental results showed that the presented approach helped optimize parameters for standard benchmark problems.

Quantum PSO is a new discrete PSO algorithm that utilizes the concept of a quantum individual. Quantum PSO utilizes the concept of a quantum bit within the quantum particle. The quantum bit can probabilistically take a value of 0 or 1 upon random observation [24]. The concept of soliton-based quantum-behaved PSO was presented by Fallahi and Taghadosi to solve optimization problems in their research work [25]. In non-linear situations, solitons can rearrange and reproduce themselves stably without becoming trapped. The experimental results of soliton quantum-behaved PSO showed significant improvements when considering probability density function-based motion scenarios.

Quantum-based PSO has also been applied to task scheduling in device–edge–cloud cooperative computing [26]. Many other variants of the PSO algorithm are available, such as bare-bone PSO [27], stochastic PSO [28], self-adaptive PSO [29], and multi-population PSO [30].

3.1. Phasor PSO

PPSO is a new and improved, simple, and adjustable PSO model proposed by Ghasemi et al. [31]. PPSO is built with the addition of the phase angle θ to particle update

equations, aiming to achieve optimal results in high-dimension optimization problems. In PPSO, each control parameter generated by the algorithm is merged with the phase angle. The increase in optimization efficiency is the most significant advantage of PPSO. Because of the phase angle (θ), PSO becomes a non-parametric algorithm with simpler calculations. In PPSO, episodic trigonometric functions, e.g., *sin* and *cos*, are used as control parameters.

3.2. Single Population Approach

Initially, PSO works best with small population sizes. However, modifications and upgrades by researchers have made PSO efficient for large populations and multimodal problems. PSO works better with different population sizes depending on the given problem [32]. In PSO, the social iteration within the population is one of the key factors of the algorithm. The population has a communication structure among particles that enables them to collectively share search space experiences, aiming to solve complex problems and improve population diversity. This social network for information exchange is called topology. PSO has three commonly used topologies, as shown in Figure 1 [33].

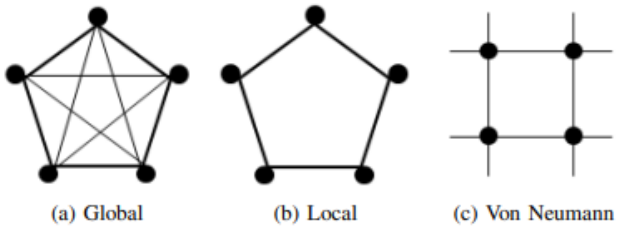


Figure 1. Single population topologies [33].

The global topology shown in Figure 1 is also known as the star topology, in which all the particles in the swarm are interconnected. Each particle in the population can share information directly. The local topology, also known as the ring topology, is where all particles in a swarm have only two neighbor particles to share information with. The von Neumann topology is the local best topology, in which each particle in a swarm has only four neighbor particles to share information with. In the von Neumann topology, particles are arranged in a grid-like structure. All three topologies increase PSO performance depending on the given problem.

3.3. Synchronous and Asynchronous Approaches to PSO

Original PSO uses a synchronous approach, in which the position and velocity of the element are reorganized after the entire swarm ends the current iteration. This kind of PSO is also known as S-PSO. S-PSO offers worthy information to all particles of the population. In S-PSO, each particle of the swarm has the benefit of picking a healthier neighbor and exploring the material delivered by this neighbor. However, premature convergence is a common shortcoming of S-PSO. In asynchronous PSO (A-PSO) [34], once the performance of the particle has been calculated, the *pbest* is updated immediately. In A-PSO, particle information is upgraded by the current iteration instead of information from the previous iteration. A-PSO has a shorter execution time but inadequate information due to the upgrading of information during current iterations. Due to reliable solution quality and robust exploitation, S-PSO performs better than A-PSO. So, the selection of the approach between S-PSO and A-PSO is problem-dependent. In some problems, S-PSO performs well, whereas in others, S-PSO shows good performance.

3.4. Multi-Population Approach

In PSO, population diversity is increased by dividing the entire population into multiple substitute swarms [35]. At the start of the algorithm, the entire population is divided into multiple sub-swarms. Then, the PSO algorithm is applied to each sub-swarm, and each sub-swarm computes the best results of its region. In the last part, all sub-

swarms share their best results and compute a single result for the whole population. Different evolutionary and coevolutionary methods are used to exchange information among sub-swarms.

3.5. Modified PSO for Multimodal Function

In the original PSO, the whole population is used for computation, but in the modified PSO (MPSO), the whole population is divided into multiple sub-swarms regarding the arrangement of the particles [36]. The best particle in each sub-swarm is stored as the local best *Lbest* of the current sub-swarm. Instead of using the global best in the velocity update, the *Lbest* is used as the best particle of each sub-swarm. Now, the velocity update equation is modified, as shown in Equation (8)

$$V_{id}^{k+1} = v_{id}^{k} + c_1 \times r_1 \times (pbest_{id}^{k} - x_{id}^{k}) + c_2 \times r_2 \times (Lbest_{id}^{k} - x_{id}^{k}) \quad (8)$$

Instead of using the *Gbest* in the velocity update equation, each sub-swarm uses its best particle thus far within its range. Because of the multiple sub-swarms, several optimal solutions, like the *Lbest* and *Pbest*, can be found, which can be useful in multimodal optimization problems. For a population p, with M as the total number of sub-populations and N as the total quantity of particles in the population, the number of particles in each sub-population is calculated using $\frac{N}{M}$.

3.6. Multi-Adaptive Strategy-Based PSO

In multi-adaptive strategy-based PSO (MAPSO), an entire population is divided into several small-size sub-swarms [37]. Two adaptive strategies, i.e., adaptive learning exemplars and adaptive population size, are introduced into the sub-swarms' mechanism to improve the comprehensive performance of MAPSO. According to the fitness value, the search particle in a sub-swarm can adaptively select its own learning particles. Throughout the entire optimization process, computational resources are rationally distributed based on the adaptation of the population. The adaptive learning strategy facilitates a favorable search behavior. In the adaptive population size strategy, the particle deletion process can speed up the convergence of MAPSO. Introducing more particles generated with the help of the differential evolution algorithm into the current population in the adaptive population size strategy can provide more helpful information. MAPSO is a relatively time-consuming variant of PSO, especially on simple uni-modal functions. It is more appropriate for complex problems rather than simple uni-modal problems.

3.7. PSO Based on Multi-Exemplar and Forgetting Capabilities

The multi-exemplar and forgetting capabilities of PSO are employed in a new version of PSO, called expanded PSO (XPSO) [11]. Initially, XPSO enhances the social learning aspect of each particle by using certain exemplars, learning from both the best particle in the local neighborhood and the best experience from the entire population, referred to as the global best. Then, diverse forgetting abilities are assigned to different particles by XPSO. In addition, the acceleration coefficients of each particle can be adjusted during the evolution process. It is very important to select a good neighbor for each particle to extract more useful information from the exemplar to provide positive guidance for the particles. In XPSO, a random order of numbers is assigned to particles, and neighbors are determined for the current particle based on this assigned random order. Although XPSO demonstrates reasonable performance compared to other PSO variants, two areas have room for further improvement and research. One is the efficiency of the newly introduced parameter in XPSO. The other involves finding ways to extract more valuable knowledge from the collective experiences of the entire population, and then applying this information to the parametric adjustment and learning models of PSO.

3.8. Multiple Archive-Based PSO

The idea of triple archive-based PSO was introduced by Xia et al. in their research work [38]. They effectively addressed the issues related to learning models and handled proper exemplar selection in their paper. They stored the proficient individuals in the archive and then reused the archive.

3.9. Coevolution Algorithm in Multi-Population Problems

Cooperative coevolution and competitive coevolution are two categories of coevolution algorithms. These approaches have been used by numerous researchers in their research on various versions of the PSO algorithm. The coevolution algorithm is a technique that provides an extension of the evolutionary algorithm for multi-objective and large population problems [10]. Coevolution improves diversity and reduces the risk of premature convergence of the whole population. It also aids in decomposing the problem into sub-parts [39]. In coevolution, the whole population is divided into two or more sub-populations, and the progress of all sub-populations is accelerated simultaneously [40]. In each sub-population, the fitness of particles is evaluated based on collaboration with individuals from other populations. In the evolutionary algorithm, each sub-population advances by manipulating its own values. So, the performance of the coevolutionary algorithm depends on direct communication between two or more particles from different sub-populations. In evolutionary algorithms, the fitness of particles is determined empirically and is independent of population circumstances. While the coevolutionary algorithm uses a biased approach, the fitness of an individual is estimated after an exchange of information with individuals from other sub-populations [38].

Niu et al. [41] introduced collaborative and competitive versions of multi-swarm cooperative PSO using the concept of master and slave swarms. Diversity in multiple slave swarms was maintained by independently running PSO on each slave swarm, which then communicated with the master swarm to evolve the knowledge of the master swarm. Wang et al. [42] presented a two-step cooperative PSO, where a two-swarm strategy is used in the first step to perform dimension partition and integration using a cooperative strategy. The velocity is controlled adaptively in the second step using an amplification factor of the velocity with a value of 0.9 to control the landscape of the considered problem. Hu et al. [43] tackled the problem of path failure in reconstructing the network topology by introducing an immune cooperative particle swarm optimization algorithm in the domain of heterogeneous wireless sensor networks. They considered macro-nodes and source sensors, creating sub-swarms in the form of k disjoint communication paths to provide alternative paths in case of broken paths. They used the immune cooperative mechanism to enhance the global search capability. The concept of adaptive cooperative PSO was introduced by Wang [44] to tackle the curse of dimensionality in cooperative PSO by dividing the swarm into sub-swarms of smaller dimensions. They controlled the exploration and exploitation capabilities of sub-swarms by exchanging information cooperatively using the adaptive inertia weight.

Li et al. [45] introduced a mixed mutation strategy-based multi-population cooperation PSO for higher-dimensional optimization problems. Multi-population cooperation PSO utilized the mean learning strategy based on dynamic segments in the coevolution process for information sharing. Covariance guidance-based multi-population cooperative PSO [35] divides the population into inferior, exploratory, and elite groups based on the Euclidean distance from the global leading particle. Cooperation between the inferior group, exploratory group, and inferior group is used in the cooperative process to maintain the balance between exploration and exploitation through information exchange. The concept of multiple populations for multiple objectives was introduced in multiple populations of coevolutionary particle swarm optimization [46]. The authors' presented algorithm was used for financial management in selecting specific values of stocks while adding cardinality constraints to balance return and risk to obtain a feasible solution. Health monitoring systems normally exchange data and information in closely cooperating medical applica-

tions. Tang et al. [47] introduced coevolution-based quantum-behaved PSO to optimize the allocation of resources in the cognitive radio sensor network domain. The experimental results showed excellent performance in the considered domain. Madni et al. [48] introduced the concept of cooperative coevolution-based multi-guide PSO by applying cooperative coevolution to each objective of the sub-swarm, aiming to reduce computational costs. The cooperative coevolution-based multi-guide PSO algorithm exhibited excellent performance in high-dimensional optimization problems.

3.9.1. Cooperative Coevolution Mechanism

Cooperative coevolution was introduced by Van den Bergh and Frans [49]. In cooperative coevolution, the initial population is divided into multiple sub-swarms. Then, the particle under assessment from the current sub-swarm is estimated by gathering the best particles from other sub-populations to form a complete solution. After the evaluation of each particle, the archive is updated. The archive stores the leading individuals throughout the evolution. A completely appropriate solution formed by the sub-swarms will be placed into the archive if no other leading solutions are found. If any leading solution is found during the iterations, the appropriate solution is replaced with the leading solution.

The sub-swarms are assessed in a repetitive mode in cooperative coevolution. The parameters of the current sub-swarm are updated before advancing to the next population. The arrangements of the current sub-swarm will be updated before moving to the succeeding sub-swarm. This method of updating arguments is based on a reflexive, anti-symmetric, and transitive order, such as using Pareto ranks and niche counts in commands to resolve rank ties. Pareto ranks use the following Equation (9) for ranking:

$$Rank(i) = 1 + n_i \tag{9}$$

where n_i is the number of dominant archive members of the i^{th} particle.

The lower-ranked particle is selected. In the event of a tie between two particles, the one with a lower niche count is chosen. The selection of a dominant particle helps increase the diversity of the overall solution. Cooperative coevolution has significantly improved the performance of team objectives.

At the start of the competitive coevolution mechanism, the whole population can be divided into multiple sub-swarms, and the initially selected variable for the probability of sub-swarms is initialized with the help of a uniform distribution. In a uniform distribution, the chance of selection of each sub-swarm is equal. So, *variable*1 is initialized using 1/D for uniform probability selection. After the first iteration, the variable of probability (*variable*1) can be upgraded depending on the process of competition between sub-swarms. Initially, the cycle probability variable is allocated to the i^{th} sub-swarm, and the competitor sub-swarm is designated using roulette wheel selection. After the selection of two sub-swarms, the solution of the competitor and current sub-swarm coevolve with all further sub-swarms to form two new sub-swarms. The sub-swarm with the improved solution is selected and included in the population with a probability selection variable in the next iteration. Now probability variable (*variable*1), which is first initialized with uniform probability, is updated using Equation (10).

$$P_{ij}(k) = P_{ij}(k-1) \pm \frac{1}{D}\alpha \tag{10}$$

where α denotes the level of learning. So, the value of p in the i^{th} sub-swarm is increased if the i^{th} sub-swarm is more adapted by decision *variable*1. It is used for the cooperative coevolution process in the next iteration of MOPSO.

3.9.2. Competitive Coevolution Mechanism

In competitive coevolution, variable 'p' is allocated to each sub-swarm, indicating the probability of signifying certain sub-swarms. Competitive coevolution used a different

strategy compared to cooperative coevolution, in which the population is divided into multiple swarms, but only two sub-swarms—the recent sub-swarm and contestant sub-swarm—can contest to be representative of variable1. Only one sub-swarm (current or contestant) can be representative at one time. In competitive coevolution, fitness implies only the robustness of the sub-swarm; improvement in one sub-swarm decreases the performance of the other sub-swarm. The continuous competition between two solutions, defeating each other, results in an increased solution quality. The evolution of the whole population in the form of sub-swarms helps prevent the solution from becoming trapped in local optima.

4. Phasor PSO Challenges and Limitations

PPSO is an improved version of PSO that uses episodic trigonometric functions, like *sin* and *cos*, as control parameters. The isolated nature of *sin* and *cos* is harnessed to characterize all control parameters of PSO. To meet this goal, each particle is linked with a one-dimensional phase angle θ. The initial value of the inertia weight is set to 0. So, the velocity update Equation (11) of PSO is now changed with the phase angle θ and an updated position calculation using Equation (12).

$$v^{lter} = p(\theta_i^{lter}) \times (pbest_i^{lter} - X_i^{iter}) + g(\theta_i^{lter}) \times (Gbest_i^{lter} - X_i^{iter}) \tag{11}$$

The phase angle for the social aspect of PPSO is represented by $p(\theta_i^{lter})$ in Equation (11), and the phase angle for the cognitive aspect is represented by $g(\theta_i^{lter})$

$$X_i^{lter+1} = X_i^{lter} + V_i^{lter} \tag{12}$$

The position of PPSO is updated by using the phasor velocity V_i in Equation (12).

In the velocity update of PPSO, $p(\theta_i^{lter})$ and $g(\theta_i^{lter})$ are calculated using trigonometric functions of sine and cosine with the angle θ in the following equations:

$$p(\theta_i^{lter}) = |\cos\theta_i^{lter}|^{2\times\sin\theta_i^{lter}} \tag{13}$$

$$g(\theta_i^{lter}) = |\cos\theta_i^{lter}|^{2\times\sin\theta_i^{lter}} \tag{14}$$

The social aspect of PPSO uses a phasor-based amplification factor denoted as $p(\theta_i^{lter})$ in Equation (13) utilizing sine and cosine functions, and the cognitive aspect uses a phasor-based amplification factor denoted as $g(\theta_i^{lter})$ in Equation (14).

In PPSO, initially, the population is randomly generated in the D-dimensional space, similar to the original PSO but with the addition of the phase angle θ for each particle, through a uniform distribution $\theta_i^{lter=1} = U(0, 2\pi)$ and an initial velocity limit $V_{max,i}^{lter=1}$. Then, the velocities of all particles are recalculated using Equation (6), and the positions of the particles are updated using Equation (16), which is the same as in the original PSO. For the next iteration phase, the angle θ and the maximum velocity of each particle are calculated using Equations (15) and (16), and the iterations are repeated until the maximum number of iterations is reached.

$$\theta_i^{lter+1} = \theta_i^{lter} + |\cos(\theta_i^{lter}) + \sin(\theta_i^{lter})| \times 2\pi \tag{15}$$

$$V_{I,max}^{lter+1} = |\cos\theta_i^{lter}|^2 \times (X_{max} - X_{min}) \tag{16}$$

The phase angle θ for the i^{th} particle for the next iteration is calculated using an amplified summation of trigonometric functions, as shown in Equation (15). The velocity is updated using a non-parametric Equation (16). One of the preeminent benefits of PPSO compared to some other PSO variants is its ability to enhance the optimization efficiency of PSO even when dealing with higher dimensions of problems. For shaping the control

parameters of PSO, planting the phasor angle is an effective, flexible, and trustworthy strategy. Despite these advantages over PSO, PPSO does have several challenges and limitations.

4.1. Slow Convergence Speed

PPSO is an efficient solution, particularly in the case of a small population size. But in the case of a large population size, it suffers slow convergence speed due to a vast number of particles. The selection of the suitable technique for dividing the population into multiple sub-swarms is required for phasor PSO.

4.2. Memory Efficiency

A large population size requires a large memory size to manage its operation. PPSO requires efficient techniques to manage memory for a large population size. PPSO consumes more time and computational resources in managing memory for a large population.

4.3. Multi-Swarm Search Ability

PPSO is effective in a single population but for large-scale optimization, its population needs to be divided into multiple swarms. A suitable strategy is required for dividing the whole population into multiple swarms.

4.4. Less Diversity

PPSO suffers from premature convergence in large populations, resulting in increased time complexity in the adjustment of the population. Insufficient population diversity is also a cause for premature convergence. Population diversity needs to be improved to enhance the exploration and exploitation abilities of PPSO.

4.5. Interaction Among Individuals

PPSO requires interaction among all individuals in an algorithm to reduce the space and time complexity of the algorithm. The best individuals sharing their values with the population help to enhance the performance of the algorithm.

5. Improved Competitive Coevolution-Based Multi-Swarm Phasor Particle Swarm Optimization

Different variants of PSO yield efficient and minimized results for large-scale optimization problems. PSO is adaptable, and its suitability for a particular problem depends on the algorithm, meaning that some variants suitable for large populations may not be as suitable for small populations. Conversely, variants that are suitable for small-scale problems may not necessarily be suitable for large-scale optimization problems. In recent years, researchers have focused on the development of PSO algorithms for large-scale optimization and multidimensional problems.

Premature convergence of PSO in global optimum problems is reduced by splitting the whole population into multiple sub-swarms. This practice also improves the population range of the exploration. Coevolutionary algorithms have been successfully applied to different PSO variants to enhance diversity and efficiency. Although enormous progress has been made in evolutionary and coevolutionary optimization like MPSO, efforts to enhance PPSO for multidimensional optimization problem-consuming coevolutionary algorithms have not been made thus far. There are two types of coevolution algorithms: 'competitive coevolution' and 'cooperative coevolution'. In cooperative coevolution, each individual coevolves with other individuals. The method proposed in the current research involves the enhancement of PPSO using a competitive coevolution technique. This technique empowers PPSO to contribute effective and efficient results in the case of a large population. This technique makes PPSO more memory efficient and reduces the risk of premature convergence.

In ICPPSO, a large population is distributed into multiple sub-swarms. Initially, the current sub-swarm is selected through a uniform distribution, and the competitor sub-swarm is nominated using the roulette wheel selection method. Then, in the competition procedure, the characteristics of both selected sub-swarms merge with the characteristics of all the other sub-swarms and produce two results. The sub-swarm that offers the finest result is the winner and represents the j^{th} decision variable. The winning sub-swarm replaces the values of the current sub-swarm. Thus, the selected sub-swarm becomes more adaptable because of coevolution, increasing its probability of representation in the next iteration. After the coevolution process, the sub-swarms are combined to form one single population. The pseudocode for ICPPSO is provided below as Algorithm 2.

Algorithm 2 Pseudocode for the ICPPSO algorithm.

Input: NP is the population size, V_{min} is the minimum velocity, V_{max} is the maximum velocity, X_{min} and X_{max} are the *minimum* and *maximum* values of the search space of a given problem, θ is the particle angle, f is the fitness function.
Output Evolved population with gbest at the optimal position to achieve optimal fitness values of the problem.

1: Parametric initialization (NP, V_max, V_min, X_max, X_min)
2: Randomly and uniformly initialize particle position $X_i(1, 2 \ldots NP)$
3: Randomly and uniformly initialize particle angle $\theta_i(1, 2 \ldots NP)$
4: Evaluate initial fitness $f(X_i^1)$ and initial *Pbest* and *Gbest*
5: Divide the whole population into multiple sub-swarms.
6: Initial selection of current and competitor swarms using a uniform distribution
7: $P(x) = \frac{1}{b-a}$
8: In subsequent cycles, roulette wheel selection is used for the selection process.
9: $p_i = \frac{f_i}{\sum_{j=1}^{N} f_i}$
10: Coevolve current and competitor sub-swarms with other sub-swarms.
11: **if** sub-swarm particle <current sub-swarm **then**
12: Current sub-swarm =sub-swarm particle.
13: **end if**
14: Competition of current and competitor sub-swarms.
15: Select best (current sub-swarm, competitor sub-swarm).
16: Adapt relevant sub-swarm according to competition results.
17: Merge all sub-swarms into a single population.
18: Update velocity and position of particles.
19: Update *pbest* and *gbest* of the whole population.
20: Update value of θ and Vmax.
21: Jump to step 5 until termination criteria reached

5.1. Summary of Contributions and Implications of Improved Competitive Multi-Swarm Phasor Particle Swarm Optimization

The contributions and implications of the improved competitive multi-swarm phasor particle swarm optimization are as follows:

- Initially, PPSO was developed for small population sizes. But in the case of a large population size, it suffers from slow convergence speed due to the vast number of particles. The incorporation of competitive coevolution into PPSO helps divide the population into multiple sub-swarms, which helps improve convergence speed.
- Simple PPSO is effective in a single population, but for large-scale optimization, its population needs to be divided into multiple swarms. A suitable strategy is required for dividing the whole population into multiple swarms. The incorporation of competitive coevolution into PPSO helps improve multi-swarm search capability.

5.2. Benefits of Improved Competitive Multi-Swarm Phasor Particle Swarm Optimization

- Simple PPSO suffers from premature convergence in populations, resulting in more time complexity in the adjustment of the population. Less population diversity is also a cause for premature convergence. Population diversity needs to be improved to enhance the exploration and exploitation ability of PPSO. The incorporation of competitive coevolution into PPSO can help improve population diversity.
- PPSO requires interaction among all individuals of the algorithm to reduce the space and time complexity of the algorithm. The best individuals sharing their values with the population help to increase the performance of the algorithm. The incorporation of competitive coevolution into PPSO helps improve the interaction among individuals.
- A large population size requires a large memory size to manage its operation. Phasor PSO spends more time and computational load on managing memory for a large population. In the case of a large population size, it suffers from slow convergence speed due to the vast number of particles. The proposed ICPPSO divides the main swam into multiple sub-swarms so the algorithm processes one sub-swarm at a time, which helps save memory.

5.2.1. Parameter Settings

In the proposed ICPPSO algorithm, the parametric adjustments differ from the original PSO. In ICPPSO, the inertia weight is set to '0', similar to PPSO. Initially, the control parameters are set based on PPSO, and then some parameters are adjusted for the coevolution process in ICPPSO. A population size of $NP = 100$ and a dimension size of $D = 10$ are selected. Six fitness functions ($f1$, $f2$, $f3$, $f4$, $f5$, $f6$) are used, along with 1000 iterations.

The whole population is divided into five sub-swarms, and the number of individuals in each sub-swarm is assigned randomly from the main population. All sub-swarms are evolved one by one. The pool contains two sub-swarms selected probabilistically, where one sub-swarm is a competitor sub-swarm and the other is a current sub-swarm. Then, two sub-swarms are used in coevolution with all other sub-swarms to form two new sub-swarms. The results generated using these parameters are reported in Table 1.

Table 1. Average fitness values for f_1 to f_6 using varying dimensions and population sizes.

Dimension	Population Size	Algorithm	f1	f2	f3	f4	f5	f6
10	100	PPSO	1.00×10^{1}	5.10×10^{-1}	9.65×10^{0}	2.22×10^{2}	2.29×10^{1}	1.26×10^{1}
		ICPPSO	$\mathbf{2.69 \times 10^{-2}}$	$\mathbf{2.69 \times 10^{-1}}$	$\mathbf{5.90 \times 10^{0}}$	$\mathbf{2.01 \times 10^{0}}$	$\mathbf{3.90 \times 10^{-1}}$	$\mathbf{5.36 \times 10^{-1}}$
		FMPSO	3.00×10^{-2}	4.00×10^{-1}	8.07×10^{0}	1.12×10^{2}	1.25×10^{0}	6.47×10^{0}
30	150	PPSO	8.54×10^{3}	6.05×10^{7}	1.70×10^{1}	8.98×10^{2}	5.03×10^{1}	3.83×10^{1}
		ICPPSO	$\mathbf{4.81 \times 10^{3}}$	$\mathbf{2.44 \times 10^{6}}$	3.70×10^{1}	2.65×10^{3}	$\mathbf{2.33 \times 10^{1}}$	$\mathbf{1.06 \times 10^{1}}$
		FMPSO	5.94×10^{3}	2.91×10^{7}	$\mathbf{1.10 \times 10^{1}}$	$\mathbf{6.84 \times 10^{2}}$	2.56×10^{1}	1.86×10^{1}
50	200	PPSO	1.01×10^{5}	9.93×10^{7}	7.10×10^{1}	1.92×10^{4}	4.75×10^{2}	4.50×10^{2}
		ICPPSO	$\mathbf{8.12 \times 10^{4}}$	$\mathbf{3.36 \times 10^{7}}$	$\mathbf{6.30 \times 10^{1}}$	8.30×10^{3}	$\mathbf{3.95 \times 10^{2}}$	$\mathbf{2.33 \times 10^{2}}$
		FMPSO	8.33×10^{4}	3.74×10^{7}	6.50×10^{1}	$\mathbf{7.90 \times 10^{3}}$	4.00×10^{2}	2.72×10^{2}

5.2.2. Phasor Angle and Initial Velocity

In the proposed ICPSO, the inertia weight is set to zero, and the phasor angle θ is used similarly to PPSO. Initially, θ is initialized with a uniform distribution of $\theta_i = U(0, 2\pi)$. In later iterations, θ is adjusted according to Equation (17), and the maximum velocity limit is upgraded according to Equation (18).

$$\theta_i^{Iter+1} = \theta_i^{Iter} + |\cos(\theta_i^{Iter}) + \sin(\theta_i^{Iter})| \times 2\pi \quad (17)$$

$$V_{I,max}^{Iter+1} = |\cos\theta_i^{Iter}|^2 \times (X_{max} - X_{min}) \tag{18}$$

The phase angle θ for the i^{th} particle for the subsequent iteration is calculated using the amplified summation of trigonometric functions, as shown in Equation (17). The velocity is updated using a non-parametric Equation (18).

5.2.3. Uniform Distribution

In the proposed ICPPSO, a uniform distribution is used for the initial values of θ, similar to the original PPSO. ICPPSO uses a uniform distribution in the initial selection of the competitor sub-swarms. In the uniform distribution, each particle has an equal probability in the selection process. The probability density function of the uniform distribution is expressed in Equation (19).

$$f(x) = \begin{cases} \frac{1}{b-a} & , a \leq x \leq b \\ 0 & , x < a \ \ or \ \ x > b \end{cases} \tag{19}$$

where a is the lower limit, b is the upper limit, and x must belong to the range $[a, b]$; otherwise, 0 is used.

5.2.4. Competition and Coevolution

PPSO lacks population diversity due to the lack of a competitive process during evolution, which results in becoming trapped in local optima. The diversity issue reduces the convergence speed of the PPSO algorithm, ultimately reducing performance. The inclusion of any competitive process during the evolutionary process of PPSO helps achieve convergence more quickly.

The competitive coevolution process increases diversity in the whole population, which helps avoid premature convergence. Diversity is incorporated because the fitness of an individual is estimated following the exchange of information with individuals from other sub-populations in the competitive coevolution process.

ICPSSO uses competition between the competitor and the current sub-swarms while coevolving with the best particle. The competitor sub-swarm is initially selected using a uniform distribution, and in subsequent iterations, it is selected using a roulette wheel selection process. The selection process relies on the probability of each sub-swarm, as expressed in Equation (20)

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_i} \tag{20}$$

where f_i is the probability of each sub-swarm, and $\sum_{j=1}^{N} f_i$ is the summation of probabilities of all swarms. After competitor sub-swarm selection, the coevolution process begins. Both the selected sub-swarms coevolve with all other sub-swarms, and the sub-swarm with the best solution emerges as the winner, representing itself in the subsequent iterations. Solutions with higher probabilities have a more significant area for selection, making them more adaptable in the selection process.

5.3. Benchmark Functions

Details of the benchmark functions used to generate the experimental results are provided in Table 2, including the name of the function, search space, and equation.

Although a large number of benchmark functions can be found in the existing literature, the choice of these functions depends on the nature of the evaluation, the modality of the model, etc. A large number of existing studies have used benchmark functions f6, f7, and f8 in their research works to reduce the complexity time of performance evaluation. For example, [50–54] employed functions f6, f7, and f8 to evaluate the performance of their proposed models. These studies suggest that functions f5, f6, or f7 suffice for the evaluation and comparison of evolutionary computing algorithms [50,54–56]. While there are several fixed-dimension functions available in the literature, our algorithm mainly addresses multi-

dimensional problems. Therefore, we have considered six standard benchmark multimodal functions, as suggested in [50–53]. The experimental results show that these functions can effectively assess performance and facilitate significance tests of the proposed algorithm in comparison to existing algorithms.

Table 2. List of benchmark functions.

Function	Function Name (Type)	Search Range	Equation
f1	Sphere model Function	[−100,100]	$[f(x) = \sum_{i=1}^{n} i.x_i^2]$
f2	Schwefel's problem function 1	[−100,100]	$[f(x) = \sum_{i=0}^{n} \left(\sum_{i=0}^{n} i.x^2\right)^2]$
f3	Schwefel's problem function 2	[−100,100]	$[f(x) = \max_{0 \le x \le D} \lvert x_i \rvert]$
f4	Rosenbrock function	[−2.048,2.048]	$[f(x) = \sum_{i=0}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)]$
f5	Rastrigin function	[−5.12,5.12]	$[f(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x) + 10)]$
f6	Griewank function	[−5.12,5.12]	$[f(x) = \sum_{i=1}^{D} (y_i^2 - 10\cos(2\pi y) + 10)]$ Where $yi = x_i^2$ if $\lvert X \rvert < 1/2$ round$(2x_i^2)$ if $\lvert X \rvert > 1/2$

6. Experimental Results and Discussion

In this section, we implement the proposed ICPPSO method using variants of the PSO algorithm, known as PPSO, including the competitive coevolution process in PPSO, to demonstrate the coevolution and sub-swarm effect on the PPSO algorithm. The experimental results of the PPSO, ICPPSO, and FMPSO algorithms are generated using populations of 100, 150, and 200 with dimensions of 10, 30, and 50, respectively. The number of training iterations considered is 1000 for all algorithms during the experiments. The experimental results for the fitness values of the PPSO, ICPPSO, and FMPSO algorithms are reported in Tables 3–8. Tables 3 and 4 show the results using a population size of 100 and a dimension size of 10.

Table 3. Average fitness values for f_1 to f_3, with NP = 100 and D = 10.

Iterations	*f1*			*f2*			*f3*		
	PPSO	**ICPPSO**	**FMPSO**	**PPSO**	**ICPPSO**	**FMPSO**	**PPSO**	**ICPPSO**	**FMPSO**
0	6412.00	6510.74	6969.90	7706.78	7546.74	8213.20	64.00	64.23	81.13
50	6412.00	6510.74	6512.90	7078.28	7546.74	7314.70	46.01	40.05	44.04
100	4861.61	4768.63	5389.70	6902.23	6374.85	7011.90	44.22	36.33	44.42
150	4861.61	4768.63	5035.30	6374.85	6312.85	6507.30	36.78	36.33	37.30
200	4861.61	4768.63	4821.50	6374.85	6312.85	6349.70	36.78	36.33	37.24
250	3454.63	3022.91	4167.80	5132.50	4144.89	4675.30	36.47	33.80	36.92
300	3454.63	3022.91	3258.80	4316.79	4144.89	4656.70	36.47	33.80	36.54
350	3454.63	2169.43	3416.60	4316.79	4144.89	4635.60	36.02	33.80	35.27
400	2802.98	2169.43	2522.20	3840.17	1365.25	2603.90	36.02	29.77	32.90
450	2802.98	2169.43	2508.50	3523.24	1365.25	2450.80	34.50	29.77	32.95
500	2678.23	2169.43	2424.70	2207.05	1365.25	1965.20	34.50	29.77	32.27
550	2189.49	2004.65	2396.30	2207.05	863.43	1543.60	31.52	29.77	31.23
600	2189.49	2004.65	2109.80	1181.02	863.43	1028.30	31.52	27.44	29.57

Table 3. *Cont.*

Iterations	*f1*			*f2*			*f3*		
	PPSO	**ICPPSO**	**FMPSO**	**PPSO**	**ICPPSO**	**FMPSO**	**PPSO**	**ICPPSO**	**FMPSO**
650	2189.49	2004.65	2176.30	867.82	304.16	586.00	31.52	27.44	30.27
700	1865.27	1829.06	1999.20	867.82	304.16	465.30	31.52	27.44	29.73
750	1865.27	1102.65	1497.80	547.05	8.48	278.60	27.65	21.70	25.08
800	1587.23	1102.65	1494.90	226.50	8.48	121.40	23.49	19.05	21.49
850	556.63	540.60	548.80	78.65	4.92	42.10	23.49	17.14	20.43
900	358.96	50.22	47.20	10.52	2.89	7.40	17.98	5.90	12.47
950	50.25	1.39	1.01	6.53	1.39	4.00	11.56	5.90	8.75
1000	10.02	0.03	0.03	0.51	0.27	0.40	9.65	5.90	8.07

Table 4. Average fitness values for f_4 to f_6, with NP = 100 and D = 10.

Iterations	*f4*			*f5*			*f6*		
	PPSO	**ICPPSO**	**FMPSO**	**PPSO**	**ICPPSO**	**FMPSO**	**PPSO**	**ICPPSO**	**FMPSO**
0	1795.17	1766.33	1801.60	48.32	48.59	50.74	46.59	45.63	46.10
50	1067.91	1084.63	1195.79	48.32	46.47	53.78	46.59	40.95	43.82
100	1067.91	1084.63	1167.23	48.32	43.91	48.90	38.58	35.93	42.57
150	1005.21	1060.89	1146.08	48.32	33.45	41.60	38.58	35.93	37.29
200	1005.21	1020.89	1013.60	48.32	33.45	41.35	28.36	25.76	28.16
250	943.46	1020.43	1010.78	45.23	33.45	40.34	28.36	20.56	26.59
300	943.46	820.74	993.14	45.23	33.45	39.90	28.36	18.91	24.70
350	884.86	820.74	966.80	45.23	33.45	37.13	23.79	12.81	18.45
400	884.86	664.94	820.72	43.72	20.50	32.80	23.79	10.18	17.96
450	884.86	664.94	783.10	43.72	20.50	32.24	21.11	8.06	16.65
500	884.86	336.92	612.51	43.72	20.50	32.13	21.11	7.87	15.00
550	661.53	336.92	510.10	43.72	20.50	32.15	21.11	7.87	14.85
600	661.53	110.37	387.17	30.12	6.33	18.30	15.88	4.06	13.10
650	661.53	110.37	386.47	22.87	6.33	14.71	15.88	3.29	12.99
700	336.56	104.85	223.32	22.87	5.69	14.38	14.29	3.21	12.93
750	336.56	104.85	231.65	22.87	4.79	13.98	14.29	0.93	7.71
800	336.56	100.55	224.53	22.87	4.24	13.68	14.29	0.93	7.63
850	336.56	41.56	194.50	22.87	0.90	11.95	12.56	0.93	6.76
900	222.36	26.77	137.70	22.87	0.90	6.50	12.56	0.54	6.71
950	222.36	8.56	116.46	22.87	0.55	3.70	12.56	0.54	6.55
1000	222.36	2.01	112.20	22.87	0.39	1.25	12.56	0.54	6.47

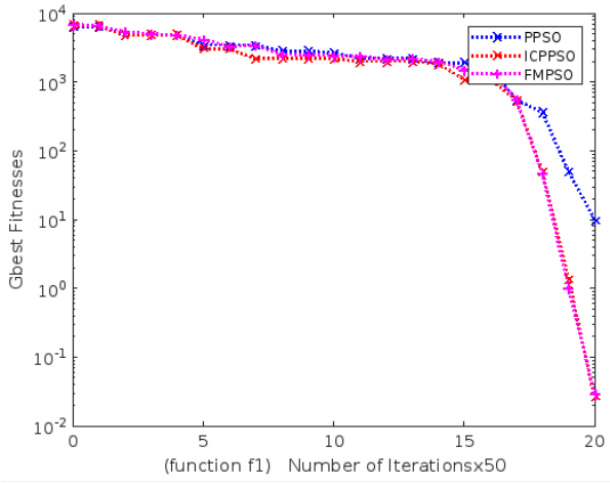
Although this cannot be deemed the all-time best implementation, the results are more optimized compared to the original PPSO, especially concerning large population diversity and size. The results generally show that initially, the effectiveness of the proposed method was almost the same, but as the population size and admissions size increased, so did the effectiveness of the proposed method. The results columns represent the same six

objective functions used for PPSO, FMPSO [46], and ICPPSO (proposed algorithm) to assess the effectiveness of the proposed solution. For all objective functions, the proposed solution performed well and became more effective with the increasing size and diversity of the population.

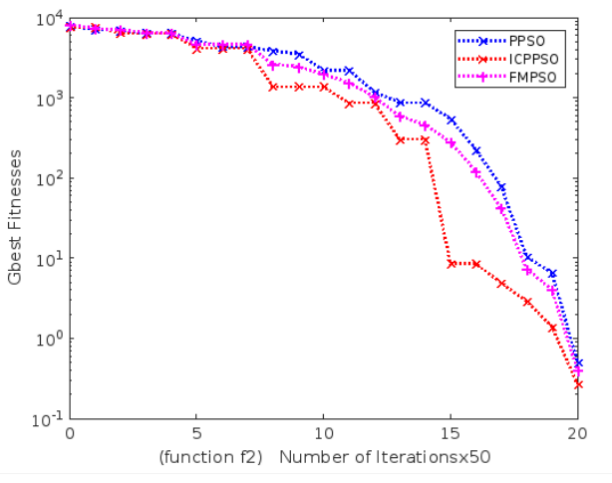
Convergence graphs for all the employed models are shown in Figure 2a–f. The competitive coevolution process within the sub-swarms played a major role in improving the performance of the proposed solution. If any sub-swarm encountered a premature convergence issue, the coevolution process provided assistance to escape from it. It was observed that PPSO achieved a good convergence speed for smaller population sizes, but for large population sizes and diversity, the chance of premature convergence increased. In the proposed solution, premature convergence was reduced because of the coevolution process. Based on the results obtained, it can be generally concluded that for all six objective functions, the proposed method is more effective and successful in cases of larger population sizes and diversity.

Figure 2 shows the logarithmic convergence graphs for PPSO, ICPPSO, and FMPSO for functions f_1 to f_6, respectively. The x-axis shows the number of training iterations in multiples of 50, and the y-axis shows the fitness values of *Gbest* for the considered algorithms. These convergence graphs were generated using a population size of 100 and a dimension of 10. Figure 2a shows that the performance of all three algorithms was similar in the initial iterations, but as the number of iterations increased, the performance of ICPPSO and FMPSO improved compared to the PPSO algorithm until the final iteration, at which point the performance of ICPPSO and FMPSO became similar. Figure 2b–d clearly show that as the number of iterations increased, the performance of the proposed algorithm gradually improved until the final iteration. Moreover, the performance of FMPSO surpassed that of the PPSO algorithm, as indicated by the results.

Tables 5 and 6 present the results of the fitness values for the PPSO, ICPPSO, and FMPSO algorithms for the test suite of six functions. A population size of 100 and a dimension size of 30 were considered in generating these results. For all three algorithms, Table 5 shows the results of functions f_1 to f_3, whereas Table 6 shows the results of functions f_4 to f_6. It is clear from the results that the performance of ICPPSO was approximately similar for function f_1, but the proposed algorithm outperformed PPSO and ICPPSO for functions f_2, f_3, f_4, f_5, and f_6. So, we can say that the incorporation of competitive coevolution significantly enhances the performance of the PPSO algorithm. On average, ICPPSO achieved improvements of 15%, 20%, 30%, and 35% compared to PPSO and FMPSO in terms of fitness results.



(**a**)



(**b**)

Figure 2. *Cont.*

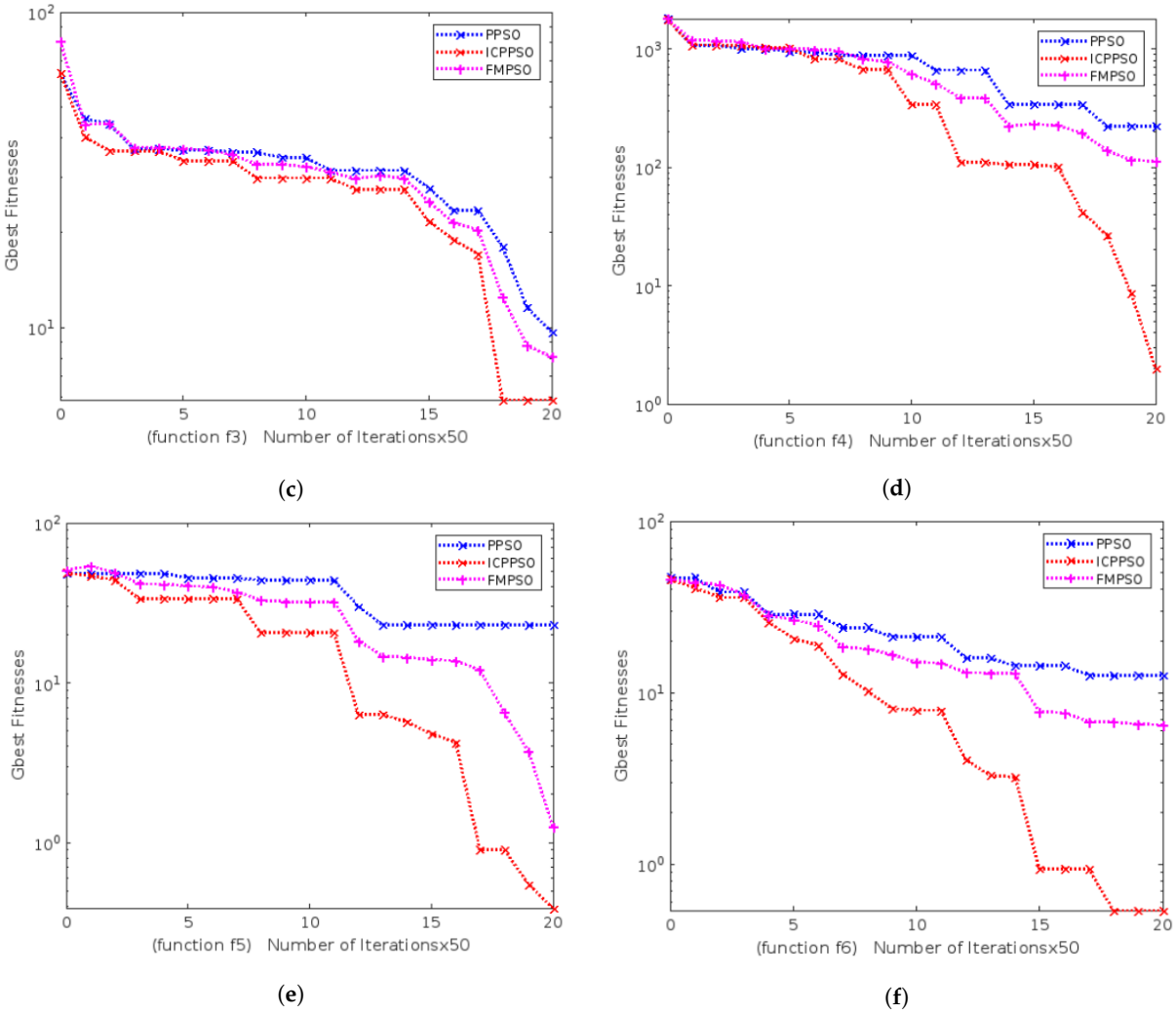


Figure 2. Logarithmic convergence of PPSO, ICPPSO, and FMPSO algorithms, with NP = 100 and D = 10.

Table 5. Average fitness values for f_1 to f_3, with NP = 150 and D = 30.

Iterations	*f1*			*f2*			*f3*		
	PPSO	**ICPPSO**	**FMPSO**	**PPSO**	**ICPPSO**	**FMPSO**	**PPSO**	**ICPPSO**	**FMPSO**
0	14,050	14,024	15,099	175,000,000	164,000,000	241,248,283	55	56	46
50	12,540	10,360	11,965	102,000,000	38,400,000	87,877,763	51	53	44
100	11,540	9301	11,133	80,000,000	10,200,000	60,702,189	45	53	42
150	11,540	9046	10,546	80,000,000	2,440,000	56,845,973	43	49	40
200	11,540	8593	10,502	60,500,000	2,440,000	47,646,675	31	49	20
250	8540	8593	8581	60,500,000	2,440,000	4,484,8163	30	37	17
300	8540	4811	6780	60,500,000	2,440,000	43,630,988	26	37	17
350	8540	4811	6697	60,500,000	2,440,000	39,809,808	18	37	14
400	8540	4811	6695	60,500,000	2,440,000	38,290,070	17	37	14
450	8540	4811	6691	60,500,000	2,440,000	37,754,894	17	37	14

Table 5. *Cont.*

Iterations	*f1*			*f2*			*f3*		
	PPSO	**ICPPSO**	**FMPSO**	**PPSO**	**ICPPSO**	**FMPSO**	**PPSO**	**ICPPSO**	**FMPSO**
500	8540	4811	6559	60,500,000	2,440,000	36,716,098	17	37	14
550	8540	4811	6461	60,500,000	2,440,000	35,860,231	17	37	14
600	8540	4811	6284	60,500,000	2,440,000	34,600,925	17	37	14
650	8540	4811	6194	60,500,000	2,440,000	33,636,356	17	37	13
700	8540	4811	6206	60,500,000	2,440,000	32,727,560	17	37	13
750	8540	4811	6125	60,500,000	2,440,000	31,774,173	17	37	13
800	8540	4811	6114	60,500,000	2,440,000	31,351,568	17	37	13
850	8540	4811	6101	60,500,000	2,440,000	31,123,045	17	37	13
900	8540	4811	6083	60,500,000	2,440,000	30,557,021	17	37	12
950	8540	4811	6049	60,500,000	2,440,000	30,098,774	17	37	12
1000	8540	4811	5941	60,500,000	2,440,000	29,068,581	17	37	11

Table 6. Average fitness values for f_4 to f_6, with NP = 150 and D = 30.

Iterations	*f4*			*f5*			*f6*		
	PPSO	**ICPPSO**	**FMPSO**	**PPSO**	**ICPPSO**	**FMPSO**	**PPSO**	**ICPPSO**	**FMPSO**
0	4595	4656	2494	106	100	104	58	57	69
50	3385	4656	1920	80	40	61	56	51	54
100	2952	3466	1522	60	38	56	56	51	54
150	2290	2650	1087	50	38	45	56	50	53
200	2001	2650	983	50	29	43	56	45	52
250	1738	2650	900	50	29	41	48	41	45
300	966	2650	891	50	23	37	48	32	42
350	966	2650	835	50	23	37	48	32	40
400	966	2650	828	50	23	35	48	32	40
450	966	2650	742	50	23	35	48	27	40
500	966	2650	740	50	23	34	48	21	35
550	966	2650	727	50	23	33	38	11	34
600	898	2650	722	50	23	32	38	11	28
650	898	2650	715	50	23	32	38	11	24
700	898	2650	713	50	23	30	38	11	24
750	898	2650	710	50	23	29	38	11	24
800	898	2650	707	50	23	28	38	11	23
850	898	2650	703	50	23	28	38	11	21
900	898	2650	701	50	23	26	38	11	21
950	898	2650	699	50	23	26	38	11	20
1000	898	2650	684	50	23	26	38	11	19

Figure 3 shows the convergence graphs of all algorithms for functions f_1 to f_6, respectively. These graphs were generated using a population size of 100 and a dimension of 30. Figure 3a shows that for the initial iterations, the performance of all the algorithms was similar; however, the performance of FMPSO and the proposed ICPPSO improved as the number of iterations increased. The performance of the proposed ICPPSO was better for functions $f1$, $f2$, $f5$, and $f6$ compared to both FMPSO and PPSO. Moreover, the performance of FMPSO was superior to that of PPSO, as indicated by the results.

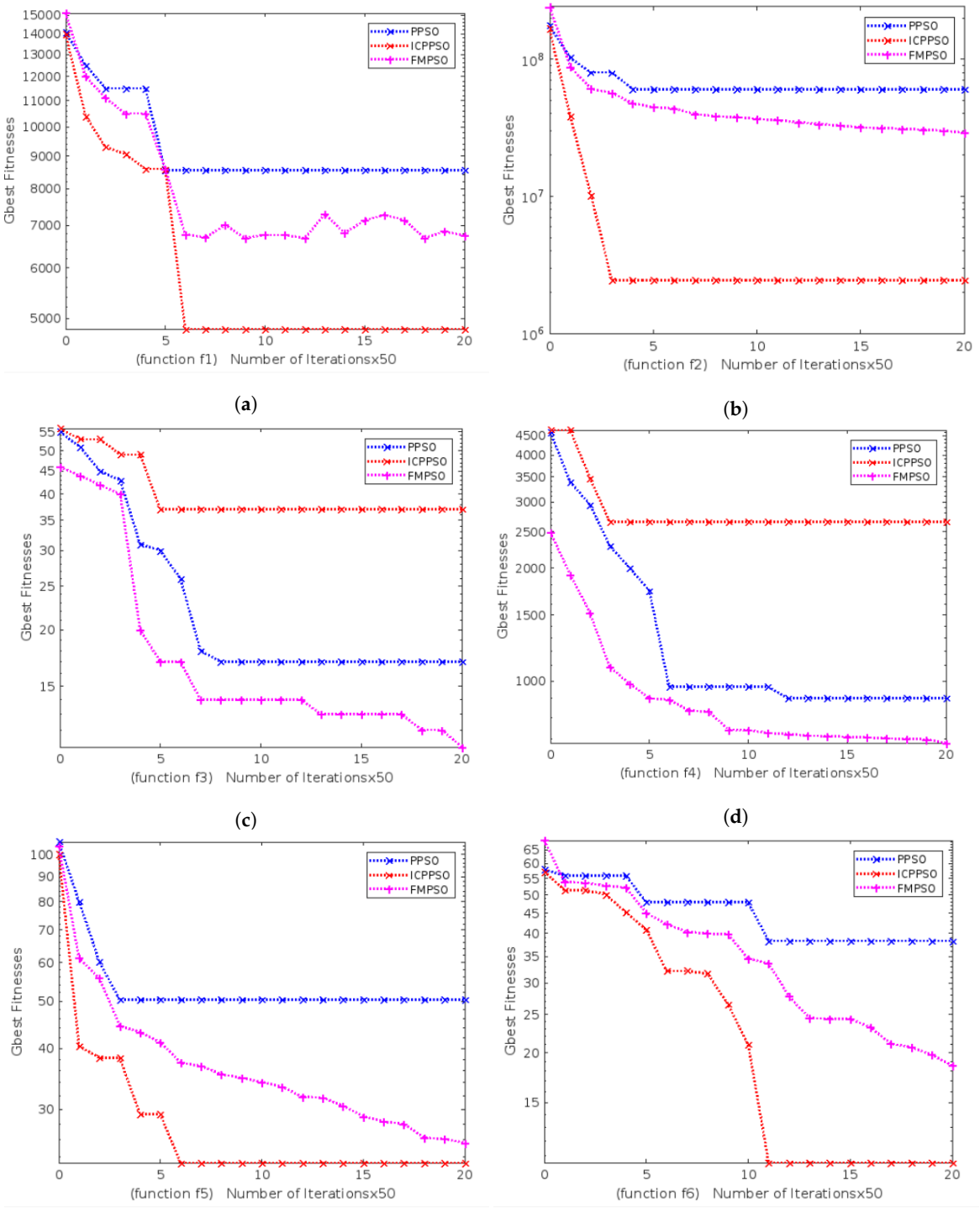


Figure 3. Logarithmic convergence of PPSO, ICPPSO, and FMPSO algorithms, with NP = 200 and D = 30.

Tables 7 and 8 present the results using a population size of 200 and a dimension size of 50, with the convergence graphs being shown in Figure 4a–f. The experimental results are indicative of the superior performance of the proposed ICPPSO compared to both PPSO and FMPSO.

Table 7. Average fitness values for f_1 to f_3, with NP = 200 and D = 50.

Iterations	*f1*			*f2*			*f3*		
	PPSO	**ICPPSO**	**FMPSO**	**PPSO**	**ICPPSO**	**FMPSO**	**PPSO**	**ICPPSO**	**FMPSO**
0	124,665	123,689	132,712	159,000,000	159,000,000	172,801,337	86	85	94
50	100,633	81,228	96,953	99,300,000	33,600,000	74,821,574	71	63	72
100	100,633	81,228	95,420	99,300,000	33,600,000	71,587,729	71	63	70
150	100,633	81,228	94,388	99,300,000	33,600,000	66,450,164	71	63	67
200	100,633	81,228	92,938	99,300,000	33,600,000	62,723,479	71	63	67
250	100,633	81,228	91,402	99,300,000	33,600,000	60,642,275	71	63	67
300	100,633	81,228	90,192	99,300,000	33,600,000	60,094,319	71	63	67
350	100,633	81,228	89,454	99,300,000	33,600,000	59,332,208	71	63	67
400	100,633	81,228	88,351	99,300,000	33,600,000	57,250,660	71	63	66
450	100,633	81,228	88,049	99,300,000	33,600,000	552,869,66	71	63	66
500	100,633	81,228	87,921	99,300,000	33,600,000	51,938,099	71	63	66
550	100,633	81,228	86,443	99,300,000	33,600,000	50,313,669	71	63	66
600	100,633	81,228	85,106	99,300,000	33,600,000	47,799,437	71	63	66
650	100,633	81,228	84,200	99,300,000	33,600,000	46,279,179	71	63	65
700	100,633	81,228	83,998	99,300,000	33,600,000	43,236,207	71	63	65
750	100,633	81,228	83,898	99,300,000	33,600,000	41,989,082	71	63	65
800	100,633	81,228	83,771	99,300,000	33,600,000	39,503,950	71	63	65
850	100,633	81,228	83,626	99,300,000	33,600,000	38,500,179	71	63	65
900	100,633	81,228	83,593	99,300,000	33,600,000	38,039,391	71	63	65
950	100,633	81,228	83,451	99,300,000	33,600,000	37,526,643	71	63	65
1000	100,633	81,228	83,345	99,300,000	33,600,000	37,416,983	71	63	65

A population size of NP = 200 and a dimension size of D = 50 were selected. Six fitness functions (f_1, f_2, f_3, f_4, f_5, f_6) were used, and the experiments were run for 1000 iterations using the PPSO, ICPPSO, and FMPSO algorithms. The convergence results are shown in Figure 4, demonstrating the superior performance of the proposed ICPPSO across all functions.

The experimental results show the fitness values of the PPSO, FMPSO, and ICPPSO algorithms for functions f_1 to f_6. These results were generated with population sizes of 100, 150, and 200 and dimensions of 10, 30, and 50. The results for all three algorithms were generated using the same parameter settings. It is evident from the results that the performance of ICPPSO surpassed that of PPSO and FMPSO for most of the functions. In the case of D = 30, the performance of the proposed algorithm was superior for functions f_1, f_2, f_5, and f_6, whereas FMPSO exhibited better performance for functions f_3 and f_4. However, ICPPSO achieved competitive performance for functions f_3 and f_4. The convergence graphs also show that ICPPSO exhibited the overall best performance for the considered suite of benchmark functions.

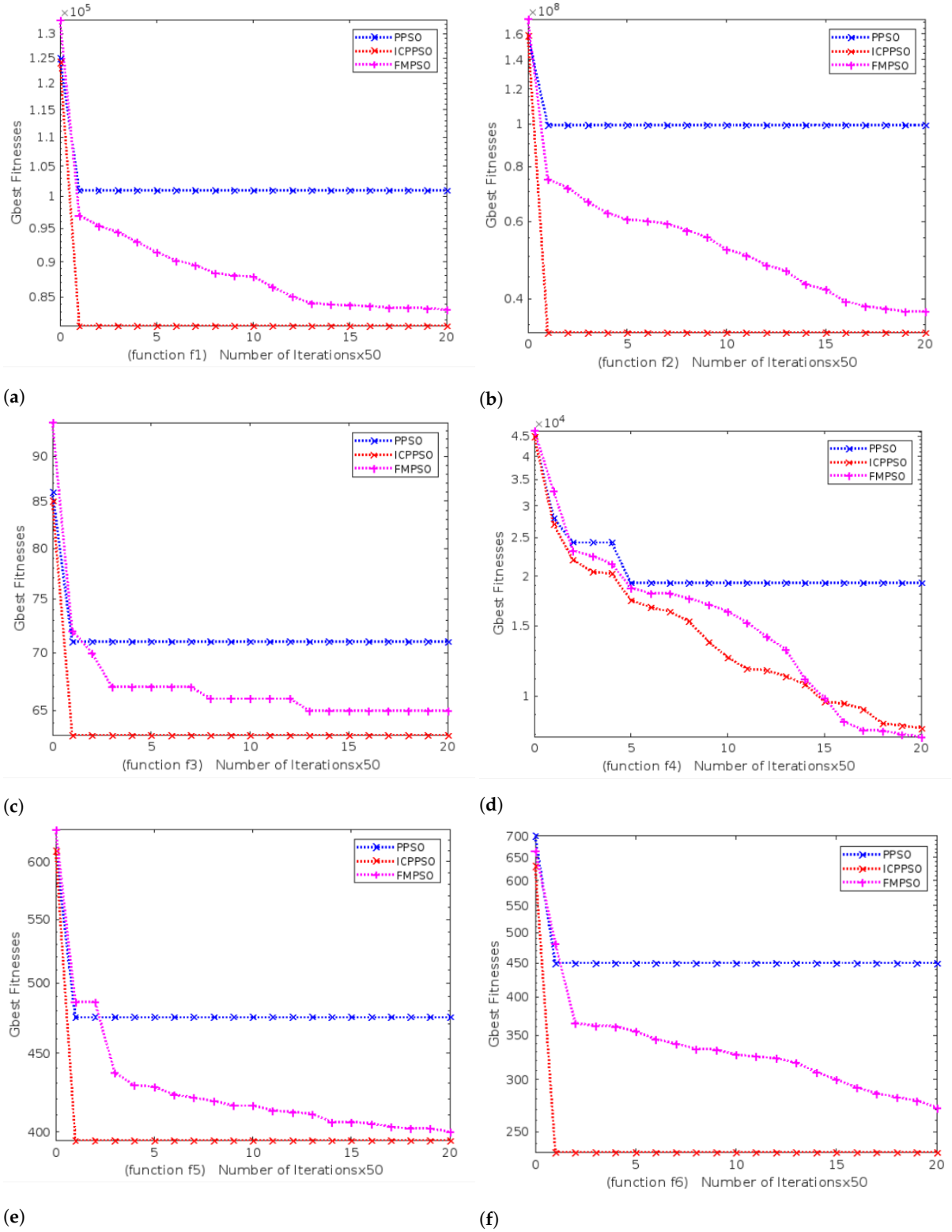


Figure 4. Logarithmic convergence of PPSO, ICPPSO, and FMPSO algorithms.

Besides improvements in the convergence of the proposed ICPPSO algorithm, it also helped increase memory efficiency. Memory usage is discussed here in the context of sub-populations rather than the whole population. Since the proposed algorithm divided the whole population into six sub-populations, it increased memory efficiency.

Table 8. Average fitness values for f_4 to f_6, with NP = 200 and D = 50.

Iterations	*f4*			*f5*			*f6*		
	PPSO	**ICPPSO**	**FMPSO**	**PPSO**	**ICPPSO**	**FMPSO**	**PPSO**	**ICPPSO**	**FMPSO**
0	45,000	44,965	46,373	610	610	630	700	629	665
50	27,950	26,975	32,771	475	395	486	450	233	482
100	24,300	22,008	23,247	475	395	486	450	233	365
150	24,300	20,477	22,509	475	395	437	450	233	362
200	24,300	20,305	21,541	475	395	429	450	233	361
250	19,157	17,383	18,701	475	395	428	450	233	355
300	19,157	16,664	18,107	475	395	423	450	233	345
350	19,157	16,311	18,060	475	395	421	450	233	340
400	19,157	15,432	17,648	475	395	419	450	233	334
450	19,157	13,674	17,016	475	395	416	450	233	333
500	19,157	12,541	16,342	475	395	416	450	233	327
550	19,157	11,677	15,297	475	395	413	450	233	325
600	19,157	11,565	14,123	475	395	412	450	233	323
650	19,157	11,221	13,076	475	395	411	450	233	318
700	19,157	10,654	11,007	475	395	406	450	233	308
750	19,157	9679	9901	475	395	406	450	233	300
800	19,157	9569	8615	475	395	405	450	233	292
850	19,157	9272	8217	475	395	403	450	233	286
900	19,157	8532	8181	475	395	402	450	233	282
950	19,157	8417	8005	475	395	402	450	233	279
1000	19,157	8304	7901	475	395	400	450	233	272

So, it can be concluded from these results that the competitive coevolution process increases diversity in the whole population, which helps avoid premature convergence. Diversity is incorporated because the fitness of an individual is estimated following the exchange of information with individuals from other sub-populations in the competitive coevolution process.

Statistical Analysis

The statistical analysis was performed using the Wilcoxon significance test. A null hypothesis states that there is no significant difference between the performance of ICPPSO and that of the other two algorithms, whereas the alternate hypothesis states that there is a significant difference between the average fitness performance of ICPPSO and the other two algorithms (PPSO and FMPSO). We applied the Wilcoxon significance test by considering a significance level of 0.05, which was compared with the p-value of the used test.

The Wilcoxon significance test was applied to the average fitness results of all functions used in this study by considering ICPPSO with PPSO and FMPSO separately. The results from these tests are reported in Table 9 with a significance level of 0.05 across all experiments. We used the default *wilcox* zero method in the Wilcoxon test. The significance results showed that all experiments of average fitness values have different statistics that demonstrated varying performance, and the p-values were smaller at the 0.05 significance level across all functions. It can be observed in Table 9 that the proposed ICPPSO exhibited

a significant difference in performance at the 0.05 significance level compared to the PPSO and FMPSO algorithms.

Table 9. Wilcoxon significance test results for the proposed algorithm (ICPPSO) vs. the PPSO and FMPSO algorithms for functions f1–f6, using 10D at a significance level of 0.05.

Function	Wilcoxon Test for PPSO vs. ICPPSO	Wilcoxon Test for FMPSO vs. ICPPSO
f1	Statistics = 17.000 ; p = 0.000	Statistics = 4.000; p = 0.000
f2	Statistics = 12.000; p = 0.000	Statistics = 10.000; p = 0.000
f3	Statistics = 1.000; p = 0.000	Statistics = 0.000; p = 0.000
f4	Statistics = 18.000; p = 0.000	Statistics = 3.000; p = 0.000
f5	Statistics = 1.000; p = 0.000	Statistics = 0.000; p = 0.000
f6	Statistics = 0.000; p = 0.000	Statistics = 0.000; p = 0.000

A significance level of 5% was used to conduct the Wilcoxon significance test on functions f1 to f6. The results of the proposed algorithm were then compared to those of the PPSO and FMPSO algorithms. It can be observed from the results of the significance test that the statistics for all these functions varied in value. However, the p-value for all the cases was 0.0000, leading to the rejection of the null hypothesis and acceptance of the alternate hypothesis. Therefore, it can be deduced that there is a significant difference in the performance of the ICPPSO algorithm compared to the state-of-the-art PPSO and FMPSO algorithms.

7. Conclusions

Different variants of PSO have the ability to solve a diverse variety of problems. Some variants, like MPSO, are effective in managing large-scale populations. The large size of the population is directly proportional to a wide diversity of solutions. In the cooperative approach, the whole population is divided into many sub-swarms, each of which coevolves with the others to form a complete solution. In competitive coevolution, the population is also divided into multiple sub-swarms, and two of these sub-swarms are selected to compete for coevolution; the swarm with the best fitness ultimately earns the right to represent itself. This competitive coevolution process is a great technique for tackling large-scale problems. The hybridization of PPSO with competitive coevolution brings about many effective enhancements for the algorithm. These enhancements include increased efficiency in handling large-scale problems, reduced solution stagnation, decreased risk of premature convergence, and greater population diversity within the algorithm. This study proposed ICPPSO, which incorporates competitive coevolution into PPSO to increase its efficiency for large populations. Six fitness functions were used to demonstrate the enhancements of PPSO. Competitive coevolution not only increased population diversity but also enabled the handling of more complex problems. Mostly, a multi-swarm strategy is used for large-scale optimization problems in PSO. A combination of competitive and cooperative coevolutionary processes was implemented to improve PPSO. Cooperative coevolution is also used independently in large-scale problems of PPSO. A more effective strategy needs to be developed without dividing the population into multiple sub-populations. Experiments were performed by considering six standard benchmark functions and parameter settings. The experimental results were discussed regarding the average fitness values of the PPSO, FMPSO, and ICPPSO algorithms. Additionally, their associated convergence graphs were presented for varying parameters. It can be concluded from the experimental results that the performance of ICPPSO is superior to that of FMPSO and PPSO for the considered benchmark functions and varying parameters. The performance comparison of ICPPSO against PPSO and FMPSO showed the superior performance of the proposed ICPPSO algorithm. The experimental results for the average fitness of the ICPPSO algorithm

indicated average improvements of 15%, 20%, 30%, and 35% over the PPSO and FMPSO algorithms. The results of the Wilcoxon statistical significance test for the ICPPSO algorithm rejected the null hypothesis, showing a significant difference in performance compared to the PPSO and FMPSO algorithms at a 0.05 confidence level. Our future work will involve the incorporation of a dynamic and efficient self-adaptive selection process into ICPPSO for large-scale problems.

Author Contributions: Conceptualization, O.A. and Q.A.; Data curation, Q.A. and K.M.; Formal analysis, O.A. and K.M.; Funding acquisition, E.B.T.; Investigation, E.B.T. and J.A.; Methodology, K.M.; Software, J.A.; Supervision, I.A.; Validation, I.A.; Visualization, E.B.T. and J.A.; Writing—original draft, O.A. and Q.A.; Writing—review and editing, I.A. All authors have read and agreed to the published version of the manuscript.

Funding: This study was funded by the European University of the Atlantic.

Data Availability Statement: Not Applicable.

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

References

1. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
2. Zheng, Y.L.; Ma, L.H.; Zhang, L.Y.; Qian, J.X. On the convergence analysis and parameter selection in particle swarm optimization. In Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 03EX693), Xi'an, China, 5 November 2003; IEEE: Piscataway, NJ, USA, 2003; Volume 3, pp. 1802–1807.
3. Sana, M.U.; Li, Z.; Javaid, F.; Hanif, M.W.; Ashraf, I. Improved particle swarm optimization based on blockchain mechanism for flexible job shop problem. *Clust. Comput.* **2021**, *26*, 2519–2537. [CrossRef]
4. Shi, Y.; Eberhart, R.C. Parameter selection in particle swarm optimization. In Proceedings of the Evolutionary Programming VII: 7th International Conference, EP98, San Diego, CA, USA, 25–27 March 1998; Proceedings 7; Springer: Berlin/Heidelberg, Germany, 1998; pp. 591–600.
5. Bansal, J.C.; Singh, P.; Saraswat, M.; Verma, A.; Jadon, S.S.; Abraham, A. Inertia weight strategies in particle swarm optimization. In Proceedings of the 2011 Third World Congress on Nature and Biologically Inspired Computing, Salamanca, Spain, 19–21 October 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 633–640.
6. Tian, M.; Gao, Y.; He, X.; Zhang, Q.; Meng, Y. Differential Evolution with Group-Based Competitive Control Parameter Setting for Numerical Optimization. *Mathematics* **2023**, *11*, 3355. [CrossRef]
7. Chen, W.N.; Zhang, J.; Chung, H.S.; Zhong, W.L.; Wu, W.G.; Shi, Y.H. A novel set-based particle swarm optimization method for discrete optimization problems. *IEEE Trans. Evol. Comput.* **2009**, *14*, 278–300. [CrossRef]
8. Liao, C.J.; Tseng, C.T.; Luarn, P. A discrete version of particle swarm optimization for flowshop scheduling problems. *Comput. Oper. Res.* **2007**, *34*, 3099–3111. [CrossRef]
9. Elbes, M.; Alzubi, S.; Kanan, T.; Al-Fuqaha, A.; Hawashin, B. A survey on particle swarm optimization with emphasis on engineering and network applications. *Evol. Intell.* **2019**, *12*, 113–129. [CrossRef]
10. Goh, C.K.; Tan, K.C.; Liu, D.; Chiam, S.C. A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design. *Eur. J. Oper. Res.* **2010**, *202*, 42–54. [CrossRef]
11. Xia, X.; Gui, L.; He, G.; Wei, B.; Zhang, Y.; Yu, F.; Wu, H.; Zhan, Z.H. An expanded particle swarm optimization based on multi-exemplar and forgetting ability. *Inf. Sci.* **2020**, *508*, 105–120. [CrossRef]
12. Shi, Y.; Eberhart, R. A modified particle swarm optimizer. In Proceedings of the 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360), Anchorage, AK, USA, 4–9 May 1998; IEEE: Piscataway, NJ, USA, 1998; pp. 69–73.
13. Eberhart, R.C.; Shi, Y. Tracking and optimizing dynamic systems with particle swarms. In Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546), Seoul, Republic of Korea, 27–30 May 2001; IEEE: Piscataway, NJ, USA, 2001; Volume 1, pp. 94–100.
14. Lei, K.; Qiu, Y.; He, Y. A new adaptive well-chosen inertia weight strategy to automatically harmonize global and local search ability in particle swarm optimization. In Proceedings of the 2006 1st International Symposium on Systems and Control in Aerospace and Astronautics, Harbin, China, 19–21 January 2006; IEEE: Piscataway, NJ, USA, 2006; p. 4.
15. Eberhart, R.C.; Shi, Y. Comparing inertia weights and constriction factors in particle swarm optimization. In Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512), La Jolla, CA, USA, 16–19 July 2000; IEEE: Piscataway, NJ, USA, 2000; Volume 1, pp. 84–88.
16. Yang, C.H.; Lin, Y.D.; Chuang, L.Y.; Chang, H.W. Double-bottom chaotic map particle swarm optimization based on chi-square test to determine gene-gene interactions. *BioMed Res. Int.* **2014**, *2014*, 172049. [CrossRef] [PubMed]

17. Arumugam, M.S.; Rao, M. 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. *Appl. Soft Comput.* **2008**, *8*, 324–336. [CrossRef]
18. Cai, X.; Cui, Y.; Tan, Y. Predicted modified PSO with time-varying accelerator coefficients. *Int. J.-Bio-Inspired Comput.* **2009**, *1*, 50–60. [CrossRef]
19. Xing, L.; Li, J.; Cai, Z.; Hou, F. Evolutionary Optimization of Energy Consumption and Makespan of Workflow Execution in Clouds. *Mathematics* **2023**, *11*, 2126. [CrossRef]
20. Meng, Z.; Zhong, Y.; Mao, G.; Liang, Y. PSO-sono: A novel PSO variant for single-objective numerical optimization. *Inf. Sci.* **2022**, *586*, 176–191. [CrossRef]
21. Lazzus, J.A.; Vega-Jorquera, P.; Lopez-Caraballo, C.H.; Palma-Chilla, L.; Salfate, I. Parameter estimation of a generalized lotka–volterra system using a modified pso algorithm. *Appl. Soft Comput.* **2020**, *96*, 106606. [CrossRef]
22. Nabi, S.; Ahmad, M.; Ibrahim, M.; Hamam, H. AdPSO: Adaptive PSO-based task scheduling approach for cloud computing. *Sensors* **2022**, *22*, 920. [CrossRef] [PubMed]
23. Eltamaly, A.M. A novel strategy for optimal PSO control parameters determination for PV energy systems. *Sustainability* **2021**, *13*, 1008. [CrossRef]
24. Yang, S.; Wang, M. A quantum particle swarm optimization. In Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753), Portland, OR, USA, 19–23 June 2004; IEEE: Piscataway, NJ, USA, 2004; Volume 1, pp. 320–324.
25. Fallahi, S.; Taghadosi, M. Quantum-behaved particle swarm optimization based on solitons. *Sci. Rep.* **2022**, *12*, 13977. [CrossRef]
26. Wang, B.; Zhang, Z.; Song, Y.; Chen, M.; Chu, Y. Application of Quantum Particle Swarm Optimization for task scheduling in Device-Edge-Cloud Cooperative Computing. *Eng. Appl. Artif. Intell.* **2023**, *126*, 107020. [CrossRef]
27. Tran, B.; Xue, B.; Zhang, M. Bare-bone particle swarm optimisation for simultaneously discretising and selecting features for high-dimensional classification. In Proceedings of the Applications of Evolutionary Computation: 19th European Conference, EvoApplications 2016, Porto, Portugal, 30 March–1 April 2016; Proceedings, Part I 19; Springer: Berlin/Heidelberg, Germany, 2016; pp. 701–718.
28. Zhao-Hui, R.; Xiu-Yan, G.; Yuan, Y.; He-Ping, T. Determining the heat transfer coefficient during the continuous casting process using stochastic particle swarm optimization. *Case Stud. Therm. Eng.* **2021**, *28*, 101439. [CrossRef]
29. Pornsing, C.; Sodhi, M.S.; Lamond, B.F. Novel self-adaptive particle swarm optimization methods. *Soft Comput.* **2016**, *20*, 3579–3593. [CrossRef]
30. Li, G.; Wang, W.; Zhang, W.; Wang, Z.; Tu, H.; You, W. Grid search based multi-population particle swarm optimization algorithm for multimodal multi-objective optimization. *Swarm Evol. Comput.* **2021**, *62*, 100843. [CrossRef]
31. Ghasemi, M.; Akbari, E.; Rahimnejad, A.; Razavi, S.E.; Ghavidel, S.; Li, L. Phasor particle swarm optimization: A simple and efficient variant of PSO. *Soft Comput.* **2019**, *23*, 9701–9718. [CrossRef]
32. Liu, P.; Liu, J. Multi-leader PSO (MLPSO): A new PSO variant for solving global optimization problems. *Appl. Soft Comput.* **2017**, *61*, 256–263. [CrossRef]
33. Figueiredo, E.M.; Ludermir, T.B. Effect of the PSO Topologies on the Performance of the PSO-ELM. In Proceedings of the 2012 Brazilian Symposium on Neural Networks, Curitiba, Brazil, 20–25 October 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 178–183.
34. Rada-Vilela, J.; Zhang, M.; Seah, W. A performance study on synchronous and asynchronous updates in particle swarm optimization. In Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, Dublin, Ireland, 12–16 July 2011; pp. 21–28.
35. Liang, P.; Li, W.; Huang, Y. Multi-population Cooperative Particle Swarm Optimization with Covariance Guidance. In Proceedings of the 2022 4th International Conference on Data-driven Optimization of Complex Systems (DOCS), Chengdu, China, 28–30 Qctober 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–6.
36. Tian, D.; Shi, Z. MPSO: Modified particle swarm optimization and its applications. *Swarm Evol. Comput.* **2018**, *41*, 49–68. [CrossRef]
37. Wei, B.; Xia, X.; Yu, F.; Zhang, Y.; Xu, X.; Wu, H.; Gui, L.; He, G. Multiple adaptive strategies based particle swarm optimization algorithm. *Swarm Evol. Comput.* **2020**, *57*, 100731. [CrossRef]
38. Xia, X.; Gui, L.; Yu, F.; Wu, H.; Wei, B.; Zhang, Y.L.; Zhan, Z.H. Triple archives particle swarm optimization. *IEEE Trans. Cybern.* **2019**, *50*, 4862–4875. [CrossRef]
39. Ma, X.; Li, X.; Zhang, Q.; Tang, K.; Liang, Z.; Xie, W.; Zhu, Z. A survey on cooperative co-evolutionary algorithms. *IEEE Trans. Evol. Comput.* **2018**, *23*, 421–441. [CrossRef]
40. Van den Bergh, F.; Engelbrecht, A.P. A cooperative approach to particle swarm optimization. *IEEE Trans. Evol. Comput.* **2004**, *8*, 225–239. [CrossRef]
41. Niu, B.; Zhu, Y.; He, X.; Wu, H. A multi-swarm cooperative particle swarm optimizer. *Appl. Math. Comput.* **2007**, *185*, 1050–1062. [CrossRef]
42. Wang, R.Y.; Hsiao, Y.T.; Lee, W.P. A new cooperative particle swarm optimizer with dimension partition and adaptive velocity control. In Proceedings of the 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Seoul, Republic of Korea, 14–17 October 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 103–109.
43. Hu, Y.; Ding, Y.; Hao, K. An immune cooperative particle swarm optimization algorithm for fault-tolerant routing optimization in heterogeneous wireless sensor networks. *Math. Probl. Eng.* **2012**, *2012*, 743728. [CrossRef]

44. Wang, L. An improved cooperative particle swarm optimizer. *Telecommun. Syst.* **2013**, *53*, 147–154. [CrossRef]
45. Li, W.; Meng, X.; Huang, Y.; Fu, Z.H. Multipopulation cooperative particle swarm optimization with a mixed mutation strategy. *Inf. Sci.* **2020**, *529*, 179–196. [CrossRef]
46. Zhao, H.; Chen, Z.G.; Zhan, Z.H.; Kwong, S.; Zhang, J. Multiple populations co-evolutionary particle swarm optimization for multi-objective cardinality constrained portfolio optimization problem. *Neurocomputing* **2021**, *430*, 58–70. [CrossRef]
47. Tang, M.; Zhu, W.; Sun, S.; Xin, Y. Mathematical modeling of resource allocation for cognitive radio sensor health monitoring system using coevolutionary quantum-behaved particle swarm optimization. *Expert Syst. Appl.* **2023**, *228*, 120388. [CrossRef]
48. Madani, A.; Engelbrecht, A.; Ombuki-Berman, B. Cooperative coevolutionary multi-guide particle swarm optimization algorithm for large-scale multi-objective optimization problems. *Swarm Evol. Comput.* **2023**, *78*, 101262. [CrossRef]
49. Kushwaha, N.; Pant, M. Modified particle swarm optimization for multimodal functions and its application. *Multimed. Tools Appl.* **2019**, *78*, 23917–23947. [CrossRef]
50. Zhang, Y.; Li, J.; Li, L. A reward population-based differential genetic harmony search algorithm. *Algorithms* **2022**, *15*, 23. [CrossRef]
51. Huang, X.; Zeng, X.; Han, R.; Wang, X. An enhanced hybridized artificial bee colony algorithm for optimization problems. *IAES Int. J. Artif. Intell.* **2019**, *8*, 87. [CrossRef]
52. Bhattacharya, S.; Tripathi, S.L.; Kamboj, V.K. Design of tunnel FET architectures for low power application using improved Chimp optimizer algorithm. *Eng. Comput.* **2023**, *39*, 1415–1458. [CrossRef]
53. Kumari, C.L.; Kamboj, V.K.; Bath, S.; Tripathi, S.L.; Khatri, M.; Sehgal, S. A boosted chimp optimizer for numerical and engineering design optimization challenges. *Eng. Comput.* **2023**, *39*, 2463–2514. [CrossRef]
54. Izci, D.; Ekinci, S.; Kayri, M.; Eker, E. A novel improved arithmetic optimization algorithm for optimal design of PID controlled and Bode's ideal transfer function based automobile cruise control system. *Evol. Syst.* **2022**, *13*, 453–468. [CrossRef]
55. Taherdangkoo, M.; Paziresh, M.; Yazdi, M.; Bagheri, M.H. An efficient algorithm for function optimization: Modified stem cells algorithm. *Cent. Eur. J. Eng.* **2013**, *3*, 36–50. [CrossRef]
56. Al-Betar, M.A.; Khader, A.T.; Awadallah, M.A.; Alawan, M.H.; Zaqaibeh, B. Cellular harmony search for optimization problems. *J. Appl. Math.* **2013**, *2013*, 139464. [CrossRef]