mathematics

MDPI

Article

Dynamical Sphere Regrouping Particle Swarm Optimization: A Proposed Algorithm for Dealing with PSO Premature Convergence in Large-Scale Global Optimization

Martín Montes Rivera [1,*], Carlos Guerrero-Mendez [2,*], Daniela Lopez-Betancur [2] and Tonatiuh Saucedo-Anaya [2]

[1] Research and Postgraduate Studies Department, Universidad Politécnica de Aguascalientes, Aguascalientes 20342, Mexico
[2] Unidad Académica de Ciencia y Tecnologia de la Luz y la Materia, Universidad Autónoma de Zacatecas, Campus es Parque de Ciencia y Tecnología QUANTUM, Cto., Marie Curie S/N, Zacatecas 98160, Mexico; danielalopez106@uaz.edu.mx (D.L.-B.); tsaucedo@uaz.edu.mx (T.S.-A.)
* Correspondence: martin.montes@upa.edu.mx (M.M.R.); guerrero_mendez@uaz.edu.mx (C.G.-M.)

Abstract: Optimizing large-scale numerical problems is a significant challenge with numerous real-world applications. The optimization process is complex due to the multi-dimensional search spaces and possesses several locally optimal regions. In response to this issue, various metaheuristic algorithms and variations have been developed, including evolutionary and swarm intelligence algorithms and hybrids of different artificial intelligence techniques. Previous studies have shown that swarm intelligence algorithms like PSO perform poorly in high-dimensional spaces, even with algorithms focused on reducing the search space. However, we propose a modified version of the PSO algorithm called Dynamical Sphere Regrouping PSO (DSRegPSO) to avoid stagnation in local optimal regions. DSRegPSO is based on the PSO algorithm and modifies inertial behavior with a regrouping dynamical sphere mechanism and a momentum conservation physics effect. These behaviors maintain the swarm's diversity and regulate the exploration and exploitation of the search space while avoiding stagnation in optimal local regions. The DSRegPSO mechanisms mimic the behavior of birds, moving particles similar to birds when they look for a new food source. Additionally, the momentum conservation effect mimics how birds react to collisions with the boundaries in their search space or when they are looking for food. We evaluated DSRegPSO by testing 15 optimizing functions with up to 1000 dimensions of the CEC'13 benchmark, a standard for evaluating Large-Scale Global Optimization used in Congress on Evolutionary Computation, and several journals. Our proposal improves the behavior of all variants of PSO registered in the toolkit of comparison for CEC'13 and obtains the best result in the non-separable functions against all the algorithms.

Keywords: PSO; regrouping PSO; large-scale optimization; swarm optimization

MSC: 68T20

Citation: Rivera, M.M.; Guerrero-Mendez, C.; Lopez-Betancur, D.; Saucedo-Anaya, T. Dynamical Sphere Regrouping Particle Swarm Optimization: A Proposed Algorithm for Dealing with PSO Premature Convergence in Large-Scale Global Optimization. *Mathematics* **2023**, *11*, 4339. https://doi.org/10.3390/math11204339

Academic Editors: Francisco Beltran-Carbajal, Julio Cesar Rosas Caro, Juan M Ramirez and Jonathan C. Mayo-Maldonado

Received: 11 September 2023
Revised: 13 October 2023
Accepted: 16 October 2023
Published: 19 October 2023

1. Introduction

Complex real-world problems with several numerical parameters and incomplete or noisy data require global optimization in multi-dimensional search spaces. Epigenesis, Phylogeny, and Ontogeny are three optimization approaches commonly used in artificial intelligence, each with its unique characteristics and applications. Artificial neural networks utilize tentative learning in Epigenesis, while evolutionary algorithms rely on competition and survival of the fittest in Phylogeny. Swarm intelligence algorithms adopt cooperative learning in their environment in Ontogeny [1–3]. In complex search spaces, locating the global optimum or a suitable solution can prove to be a formidable task, as the likelihood of encountering local optimal regions tends to increase with higher dimensions [4]. Advancements in technology

have made solving Large-Scale Global Optimization (LSGO) more feasible, resulting in the creation of specialized algorithms. However, a standard evaluation method for comparison is necessary [5]. The IEEE Congress on Evolutionary Computation (CEC) tool is a widely used benchmark for LSGO problems. It includes optimization functions that simulate real-world problems and has been used to evaluate algorithms in [5–19]. Several metaheuristics have been applied to address the challenges associated with solving Large-Scale Global Optimization (LSGO) problems. These metaheuristics include Genetic Algorithms (GAs), Evolutionary Strategies (ES), Evolutionary Programming (EP), Memetic Algorithms (MAs), and Differential Evolution (DE), among others [1,5,6,20]. In the field of numerical optimization, swarm intelligence (SI) is considered a strong competitor to evolutionary algorithms (EAs) due to its relatively lower complexity and smaller input parameter sizes. However, in the context of Large-Scale Global Optimization (LSGO), SI algorithms tend to experience stagnation due to the presence of multiple local optimal regions [5]. Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Artificial Bee Colony (ABC) are among the widely used SI algorithms. Some other alternatives that have shown good results in various applications are the Wolf Algorithm (WA), Butterfly Algorithm (BA), Krill Algorithm (KA), and Moth Search Algorithm (MSA) [1,5,6]. The Particle Swarm Optimization (PSO) algorithm, proposed by Kennedy and Eberhart in 1995, is widely used for numerical optimization. It is based on the behavior of flocks of birds [21]. While the Particle Swarm Optimization (PSO) algorithm produces commendable outcomes in uncomplicated search spaces, it faces challenges when it comes to optimization in the presence of multiple local optimal regions [19,22–28]. Many researchers have put forth different versions of the PSO algorithm to tackle LSGO. One such variation is the GPSO, introduced in 1998 by Yuhui Shi and Russell Eberhart. The GPSO incorporates an inertial parameter that facilitates quick convergence and enhances the algorithm's ability to resist local optima by promoting exploration. However, all the particles continue to share the best position, which can potentially result in being stuck around the global best position [24,29]. GEPSO is a PSO variant that uses three inertial weights and a new speed equation with two parameters to improve swarm convergence. It has been successfully tested in 50-dimensional optimization problems [1]. In the work presented by [16], Particle Swarm Optimization (PSO) is designed to circumvent stagnation by iteratively shifting particles toward the boundaries of the search space. Regrouping PSO (RegPSO) overcomes stagnation by resetting the position of particles and redefining the search space after detecting minimal separation between all positions within the swarm. This approach offers an effective means of addressing the issue of stagnation in PSO algorithms [30]. Canonical Deterministic PSO (CD-PSO) detects stagnation in iterations and re-energizes the swarm for better search space exploration [31]. IAPSO is an accelerated version of the PSO that uses entropy analysis to detect stagnation [32]. Intrinsic Dimension and Concise PSO use time steps to detect stagnation [33]. Multiple-strategy learning PSO (MSL-PSO) uses memorization to balance exploration and exploitation during learning [27]. PSO also incorporates the concept of memory forgetting observed in biological systems. This allows the algorithm to efficiently explore the search space and improve the quality of the solutions obtained [15]. Utilizing cooperative and competitive mechanisms, as detailed in [25], presents feasible alternatives for preventing stagnation. Multi-swarm PSO (MSPSO) utilizes multiple groups of particles to search and exploit the space with variations in topology, communication, and movement to adapt to an optimization problem [34–36]. The MSPSO algorithm divides a swarm into sub-swarms for exploring and exploiting multiple locations in search space. Dynamical topology and gradual reduction in particles per sub-swarm produce varied results [22]. There are also multi-swarm alternatives that even consider the problem of stagnation in PSO, including stagnation detection mechanisms like those in [36]. Although the particles in a swarm algorithm continue to update their position based on the best global position, they may still converge in local optimal regions. Subdividing the swarm can prevent stagnation but requires more function evaluations, which increases the algorithm's running time [22,34,37,38]. PSO also hybridizes with other algorithms to improve its results similar to Multi-gradient PSO in [24], Fuzzy Self-Tuning PSO in [19], PSO with artificial neural networks in [39], Global Genetic Learning

PSO in [21], Multi-swarm PSO with evolutionary algorithms [40], Multi-chaotic PSO with deterministic chaos and evolutionary computation techniques [18], and PSO with reinforced learning and a multi-swarm approach [17].

The enhancement of the Particle Swarm Optimization (PSO) algorithm for solving Large-Scale Global Optimization (LSGO) problems with lower computational complexity, without compromising its parallel structure, can have a significant impact on the field of multi-dimensional optimization.

This paper proposes a new method called DSRegPSO, which is a variation of PSO. It uses dynamic sphere regrouping and momentum conservation to prevent position stagnation, regulate exploration and exploitation of the search space, and maximize exploration when particles travel outside the position limits. The proposed method maintains the parallel structure of PSO and could have significant implications for multi-dimensional optimization.

Contribution

DSRegPSO is a novel optimization algorithm that makes use of a regrouping dynamical sphere mechanism and momentum conservation effect to deal with LSGO problems. These mechanisms are designed to prevent the swarm from getting stuck in local optima by continuously rejuvenating the swarm and balancing the trade-off between exploration and exploitation of the search space. The dynamical sphere mechanism mimics the foraging behavior of birds when looking for food, while the momentum mechanism emulates how birds interact with their surroundings.

To evaluate the effectiveness of the DSRegPSO algorithm, we used the CEC'13 benchmark functions, which are widely used in similar works like those in [41–47] as a standard test suite for LSGO problems. The CEC'13 test is composed of 15 optimization functions, including the Sphere Function, Elliptic Function, Rastrigin's Function, Ackley's Function, Schwefel's Problem 1.2 Function, Rosenbrock's Function, and their variants [44]. These functions are designed to challenge algorithms in solving complex LSGO problems. The CEC'13 test is also used by several well-recognized journals and is the same benchmark used in the annual Special Session and Competition on Large-Scale Global Optimization (SSCLSGO), organized by the IEEE World Congress on Computational Intelligence (WCCI) [16].

2. Materials and Methods

In this section, we will discuss the biological inspiration behind the PSO algorithm. We will also cover the most commonly used variant of PSO, the GPSO, as well as the techniques used to tackle local optima in PSO. Additionally, we will introduce the original RegPSO algorithm, which is the variant that most closely resembles our proposed DSRegPSO algorithm.

2.1. Biological Inspiration

PSO is inspired by bird flocking or fish schooling. In this algorithm, birds search for a single food source in a defined area. PSO particles represent birds that seek the global optimum or the position of the food in the search space. During the search, birds adjust their position by following the current optimum or the one nearest to the food. Birds also adjust their speed depending on the distance between themselves and the best position of the swarm or its best-known position [48].

2.2. GPSO

GPSO is a Particle Swarm Optimization technique that leverages the behavior of n particles. The position of every i particle is defined by a vector, $\vec{X}_i$, which consists of D dimensions. The initial position of each particle is determined by a random array, $\vec{\psi}_i$, within the position bounds of a lower limit (L_l) and an upper limit (L_u). The velocity of every particle, represented by $\vec{V}_i$, is used to update the position of each particle during the entire run. The cost of each particle, C_i, is determined by evaluating the cost function $f\left(\vec{X}_i\right)$ for each $\vec{X}_i$. The positions that yield the

best cost are identified as the global best position $\left(\vec{P_G}\right)$, which is the position of the particle with the best cost, and the best position of each particle $\left(\vec{P_i}\right)$, which is the position where the best cost was obtained for each particle.

During each iteration, the speed of every particle is updated using Equation (1), ensuring that it stays within the lower and upper speed limits $[LS_l, LS_u]$ [21].

$$\vec{V_i} = w \cdot \vec{V_i} + c_1 \cdot \vec{R_1} \cdot \left(\vec{P_G} - \vec{X_i}\right) + c_2 \cdot \vec{R_2} \cdot \left(\vec{P_i} - \vec{X_i}\right) \quad (1)$$

The parameter w is an inertial coefficient that ranges between 0.9 and 1.2, as proposed in the original algorithm in [29]. The parameters c_1 and c_2 represent personal and social coefficients, respectively, and their values are either user-defined or typically set to their upper limit of 2. Additionally, the arrays $\vec{R_1}$ and $\vec{R_2}$ are random arrays with a size of $1 \times D$ and values in the range of $[0, 1]$. The position of particles is updated every iteration using Equation (2) while being constrained within the minimum and maximum position limits $[L_l, L_u]$.

$$\vec{X_i} = \vec{X_i} + \vec{V_i} \quad (2)$$

The particles' positions $\vec{X_i}$ are updated using Equation (2) in each iteration of the GPSO algorithm until it meets either the specified cost value (C_d) or the maximum number of iterations (k_{max}) [27]. The objective function $f\left(\vec{X_i}\right)$ is used to determine C_i and the positions of $\vec{P_G}$ and $\vec{P_i}$. The velocity $\vec{V_i}$ is determined using Equation (1) in each iteration, and then Equation (2) updates the position $\vec{X_i}$. The GPSO algorithm continues to calculate new global best positions $\vec{P_G}$ and $\vec{P_i}$ in every iteration. Algorithm 1 provides a complete description of the GPSO for each k iteration.

Algorithm 1 GPSO

Data: D, n, $f(\vec{X_\iota})$, L_l, L_u, LS_l, LS_u, c_1, c_2
Result: $\vec{P_G}$
$\vec{X_\iota} = \vec{\psi_\iota} \in [L_l, L_u]$;
$\vec{V_\iota} = 0$;
$C_i = f(\vec{X_\iota})$;
$G_{P,i} = C_i$;
$G_B = min(G_{P,i})$;
$\vec{P_\iota} = \vec{X_\iota}$;
$\vec{P_G} = \vec{X}_{argmin(f(\vec{X_\iota}))}$;
$w = [0.9, 1.2]$;
$it = 0$;
while $k < k_{max}$ **or** $G_B \leq C_d$ **do**
  $\vec{V_\iota} = w \cdot \vec{V_\iota} + c_1 \cdot \vec{R_1} \cdot (\vec{P_G} - \vec{X_\iota}) + c_2 \cdot \vec{R_2} \cdot (\vec{P_\iota} - \vec{X_\iota})$;
  $\vec{V_\iota} = \vec{V_\iota} \in [LS_l, LS_u]$;
  $\vec{X_\iota} = \vec{X_\iota} + \vec{V_\iota}$;
  $\vec{X_\iota} = \vec{X_\iota} \in [L_l, L_u]$;
  $C_i = f(\vec{X_\iota})$;
  for $i \leftarrow 1$ **to** n **do**
    if $G_{P,i} < C_i$ **then**
      $G_{P,i} = C_i$;
      $\vec{P_\iota} = \vec{X_\iota}$;
    end
  end
  $G_B = min(G_{P,i})$;
  $\vec{P_G} = argmin\left(f(\vec{P_\iota})\right)$;
end

2.3. Dealing with Local Optimums in PSO

One of the key issues with the basic formulation of PSO is that it's possible for all particles to be attracted to a local optimum, which can lead to stagnation. For instance, in the context of biology, birds tend to fly toward the nearest food source, even if it is not the best one, which can cause stagnation [30].

There are several alternatives explored when PSO gets stuck in optimal local regions, including [30]:

- Stop the search and accept the result.
- Continue the search while hoping to find a better solution.
- Restart the swarm from new locations and search again.
- Mark the areas in the search space that lead to a local optimum and avoid them.
- Reinvigorate the swarm to maintain diversity.

2.4. Regrouping PSO

Stagnation in a PSO algorithm happens when particles' positions have converged prematurely to a local optimum, keeping them in a nearby area in the search space [30].

To avoid stagnation in the swarm, RegPSO, a variation of PSO, measures the maximum Euclidean distance between each particle and the global best position for each iteration k using Equation (3) [30].

$$\delta(k) = \max\left|\vec{X_i} - \vec{P_G}\right| \tag{3}$$

Stagnation is confirmed when the distance between particles and $\vec{P_G}$ around the diameter of the search space $\mathrm{diam}(\Omega)$ is below the user-selected ϵ stagnation threshold. Ref. [30] suggests using $\epsilon = 1.1 \times 10^{-4}$ for its regrouping mechanism in Equation (4).

$$\delta_{norm} = \frac{\delta(k)}{\mathrm{diam}} < \epsilon \tag{4}$$

RegPSO reorganizes the swarm around the global best in case of stagnation or when a maximum number (k) of evaluations per grouping (max_e) is reached. This reorganization takes place between the minimum of the original range of the search space in dimension j and the product of a regrouping factor ρ times the maximum distance along dimension j of any particle compared with the global best position $\left(\vec{P_G}\right)$, as shown in Equation (5).

$$range_j(\Omega^r) = min\left(range_j\left(\Omega^0\right), \rho \cdot max\left|\vec{X_i} - \vec{P_G}\right|\right) \tag{5}$$

where the regrouping factor proposed in [30] is $\rho = \frac{6}{5\epsilon}$.

The position regrouping re-initializes the position of particles using Equation (6).

$$\vec{X_i} = \vec{P_G} + \vec{R} \circ range(\Omega^r) - \tfrac{1}{2} range(\Omega^r) \tag{6}$$

With its respective lower and upper bounds, r is the regrouping index starting at 0 and increasing by one with each regrouping.

According to this new range obtained, the algorithm sets a new maximum velocity with each regrouping, i.e., there are j upper limits determined depending on the search space dimension, as in Equation (7).

$$LS_{u,j} = \lambda \cdot range(\Omega^r) \tag{7}$$

where λ is a percentage that maintains speed under its limits $[LS_l, LS_u]$.

The algorithm's loop continues until it meets a stop criterion, such as achieving the desired cost value (C_d) or completing a specific number of iterations (k_{max}) [30]. Figure 1 presents an example of the effects of regrouping in $\left(\vec{P_G}\right)$ throughout iterations.

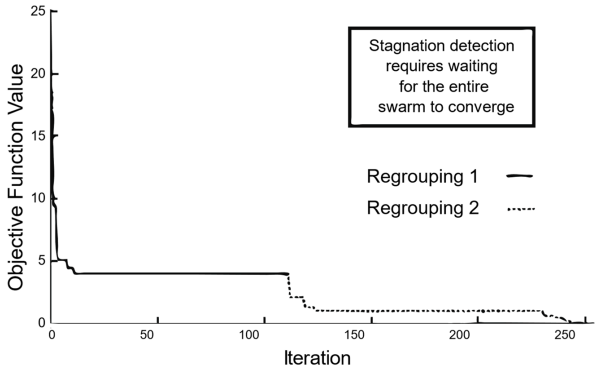

Figure 1. Regrouping behavior of cost value across iterations with the RegPSO algorithm [30].

Algorithm 2 contains a comprehensive explanation of RegPSO.

Algorithm 2 RegPSO

Data: $D,\ n,\ f(\overrightarrow{X_i}),\ L_l,\ L_u,\ LS_l,\ LS_u,\ c_1,\ c_2$
Result: $\overrightarrow{P_G}$
for $j \leftarrow 1$ **to** D **do**
  $range_j(\Omega^r) = min\left(range_j(\Omega^0), \rho \cdot max\left|\overrightarrow{X_i} - \overrightarrow{P_G}\right|\right);$
  $LS_{u,j} = \lambda \cdot range(\Omega^r);$
  for $i \leftarrow 1$ **to** n **do**
    $\overrightarrow{V_i} = \overrightarrow{V_i} \in \left[LS_{l,j}, LS_{u,j}\right]$
  end
end
for $i \leftarrow 1$ **to** n **do**
  $\overrightarrow{V_i} = \overrightarrow{V_i} \in \left[LS_{l,j}, LS_{u,j}\right]$
end
for $i \leftarrow 1$ **to** n **do**
  $\overrightarrow{X_i} = \overrightarrow{\psi_i} \in [L_l, L_u];$
  $\overrightarrow{P_i} = \overrightarrow{X_i};$
end
if $r = 0$ **then**
  $G_B = min\left(G_{P,i}\right);$
  $\overrightarrow{P_G} = argmin\left(f\left(\overrightarrow{X_i}\right)\right);$
end
$k = 1;$
while $k < k_{max}$ or $G_B \leq C_d$ **do**
  $\overrightarrow{V_i} = w \cdot \overrightarrow{V_i} + c_1 \cdot \overrightarrow{R_1} \cdot \left(\overrightarrow{P_G} - \overrightarrow{X_i}\right) + c_2 \cdot \overrightarrow{R_2} \cdot \left(\overrightarrow{P_i} - \overrightarrow{X_i}\right);$
  $\overrightarrow{V_i} = \overrightarrow{V_i} \in [LS_l, LS_u];$
  $\overrightarrow{X_i} = \overrightarrow{X_i} + \overrightarrow{V_i};$
  $\overrightarrow{X_i} = \overrightarrow{X_i} \in [L_l, L_u];$
  $C_i = f(\overrightarrow{X_i});$
  for $i \leftarrow 1$ **to** n **do**
    if $G_{P,i} < C_i$ **then**
      $G_{P,i} = C_i;$
      $\overrightarrow{P_i} = \overrightarrow{X_i};$
    end
  end
  $G_B = min\left(G_{P,i}\right);$
  $\overrightarrow{P_G} = argmin\left(f\left(\overrightarrow{X_i}\right)\right);$
  $\delta(k) = \max|\overrightarrow{X_i} - \overrightarrow{P_G}|;$
  $k = k + 1;$
  if $\frac{\delta(k)}{\text{diam}} <$ **or** $k = max_e$ **then**
    $range_j(\Omega^r) = min\left(range_j(\Omega^0), \rho \cdot max\left|\overrightarrow{X_i} - \overrightarrow{P_G}\right|\right);$
    $\overrightarrow{X_i} = \overrightarrow{P_G} + \vec{R} \circ range(\Omega^r) - \frac{1}{2} range(\Omega^r);$
    $LS_{u,j} = \lambda \cdot range(\Omega^r);$
  end
end

2.5. Dynamical Sphere Regrouping PSO (DSRegPSO)

In this section, we will explore the inspirations behind the proposed DSRegPSO algorithm, as well as the algorithm itself. We aim to provide a comprehensive understanding of the development and implementation of DSRegPSO.

2.5.1. DSRegPSO Inspiration

Similar to other heuristics, PSO strives to enhance a cost function that is depicted as a mathematical expression. Achieving this objective involves multiple assessments that adjust the numerical parameters optimized either using Algorithm 1, Algorithm 2, the novel algorithm presented in this research, Algorithm 3, or alternative optimization methodologies.

The PSO algorithm involves birds or particles that continuously move toward the position with the best cost evaluation, denoted as $\vec{P_G}$, or the position closest to 0 in a minimizing problem. The algorithm tests i trajectories every iteration, with i particles searching for the optimal point where the cost function converges to 0, as shown in Equation (8).

$$\lim_{(\vec{X})\to(\vec{X_i})} f\left(\vec{X}\right) = 0 \tag{8}$$

To establish the validity of the limit, it is necessary to demonstrate its existence using the epsilon–delta definition, which has been adapted from reference [49]. This can be achieved by defining the particle's position as $\vec{X}$, the cost function as $f\left(\vec{X}\right)$, the global best cost as G_B, and the best global position as $\vec{P_G}$, as outlined in Definition 1.

Definition 1. *Let $f\left(\vec{X}\right)$ be a function defined on an open interval around $\vec{X_i}$. The limit of $f\left(\vec{X}\right)$ as $\vec{X}$ approaches $\vec{P_G}$ is G_B , i.e., $\lim_{(\vec{X})\to(\vec{P_G})} f\left(\vec{X}\right) = G_B$. If for every $\epsilon > 0$, there exists a $\delta > 0$ such that for all $\vec{X}$, Equation (9) is satisfied.*

$$0 < \left|\vec{X} - \vec{P_G}\right| < \delta \Rightarrow \left|f\left(\vec{X}\right) - G_B\right| < \epsilon \tag{9}$$

Definition 1 considers a hyper-sphere delimited by diameter δ that approximates the difference $\left|f\left(\vec{X}\right) - G_B\right|$ to a value ϵ when $\delta > 0$, i.e., $0 < \left|\vec{X} - \vec{P_G}\right| < \delta$, but if $\delta = 0$, then $\left|\vec{X} - \vec{P_G}\right| = 0$ and $\epsilon = 0$; therefore, $\left|f\left(\vec{X}\right) - G_B\right| = 0$. In other words, if $\delta = 0$, then $\vec{X} = \vec{P_G}$ and $f\left(\vec{X}\right) = G_B$, which support that stagnation problems occur when all particles remain in approximately the same position, with almost the same cost.

Based on Definition 1, an alternative approach for revitalizing the swarm is to adjust the position of particles within a hypersphere that has an acceptable diameter of δ. This adjustment limits the approximation of vector X_i to vector P_G. Particles under this δ diameter can be repositioned to enhance the possibility of discovering a new cost C_i that may be superior to G_B with the new approximation trajectories. The swarm can avoid a local optimal region in the green function that has multiple possible local optimums by repositioning particles inside the hypersphere in red, as illustrated in Figure 2.

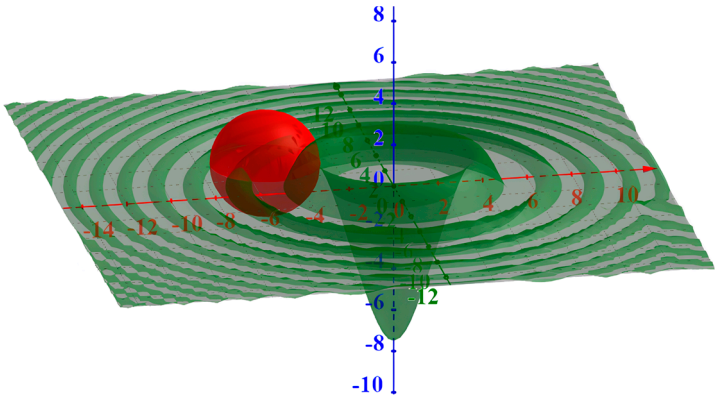

Figure 2. Hypersphere controlling repositioning of particles to avoid stagnation.

According to [49], it is possible that trajectories may lead to a different value of C_i in $f\left(\vec{X}\right)$ when the dimension of position D is greater than 1. Therefore, it is recommended to establish a boundary or the maximum diameter for approximation to determine when to relocate a particle in the search space to preserve the diversity of the swarm.

In a biological context, δ represents the situation when birds have visited and exhausted several locations with less food than the position with the most food (P_G). In this case, the birds could return to eat at P_G while there is enough food for them (their position is outside the diameter δ). If the food is exhausted, the remaining birds without food continue flying from their current location to a different one or the algorithm repositions the particles outside δ.

The new position updating uses a random speed array ψ_i, generated when the δ_i distance between the position of particle i and P_G is under the desired δ diameter. This change in position means that particles too near to P_G do not spend time exploiting this search space region. Allowing δ to be a dynamic parameter that increases and decreases the hyper-sphere diameter changes the swarm behavior for exploring or exploiting during the entire run.

When δ is greater, particles explore a more extensive region, and when it is smaller, particles exploit a smaller region. Furthermore, in this proposal, the delta value increases each iteration that G_B remains under the desired percentage of change ζ, allowing the algorithm to automatically change from the exploration to exploitation stage as the global best improves.

This behavior resembles the number of birds allowed given the amount of food and how fewer birds are admitted over time while food decreases in that location, forcing them to leave their current position and look for more food.

Since particles are repositioned when δ_i is less than δ, particles are repositioned with ψ_i when δ reaches its maximum value δ_{max} due to minor improvements in G_B. The main improvement in GPSO is the use of the previous speed with inertial momentum, allowing exploration at the beginning of the run, but its effect diminishes as the speed is approximated to 0 because $(P_G - X_i)$ and $(P_i - X_i)$ are near 0.

The inertial momentum varies depending on delta because it already controls exploration and exploitation. Thus, the inertial momentum increases when δ reaches a higher value or G_B remains with minor changes, i.e., when particles are more likely to converge. On the other hand, the inertial momentum decreases, maximizing exploitation when δ reaches a lower value because G_B improved more than $\zeta \cdot G_B$.

In DSRegPSO, the maximum allowed velocity LS_u is controlled depending on the speed of hyper-sphere expansion S_s, or the parameter that controls how fast we change δ each iteration, and the improvement is lower than $\zeta \cdot G_B$. The update mechanism for S_s increases it every time δ reaches its maximum value δ_{max}. Similar to delta, S_s has

boundaries: a maximum speed of expansion S_{max} and a minimum one S_{min}. When the algorithm reaches S_{max}, S_s restarts to the value S_{min}.

Several researchers have controlled the maximum velocity since they found that unlimited velocity produces divergence in the algorithm [50]. Thus, when the max velocity is higher, there is more exploration or divergence; when it is lower, more exploitation occurs.

Considering the effect of velocity in exploration and exploitation, in DSRegPSO, we control the maximum allowed velocity LS_u depending on the speed of hyper-sphere expansion S_s, or the parameter that controls how fast we change δ each iteration, and the improvement is lower than $\zeta \cdot G_B$.

Biologically, suppose birds are repeatably attracted to the same position without finding more food. In that case, it is as if they suffer desperation to find it, causing an increment in their speed while their energy reserves allow it. The desperate situation of food about to end makes birds travel faster and more vaguely.

Thus, the higher the S_s, the more vagueness of particles maximizing exploration. Additionally, we also maximize exploration in the inertial momentum by including S_s and δ in its update.

2.5.2. The DSRegPSO Algorithm

The DSRegPSO algorithm's initialization process involves the GPSO initialization parameters outlined in Algorithm 1, with the exception of parameter w. In its place, Equation (10) utilizes parameter ω. This parameter uniformly distributes the maximum inertial momentum (M_{max}) across its dependencies, $\frac{\delta}{\delta_{max}}$ and $\frac{S}{S_{max}}$, both of which fall within the range of $[0, 1]$.

$$\omega = \frac{M_{max}}{2} \tag{10}$$

In DSRegPSO, a new updating speed equation is defined for each iteration (k) in Equation (11), taking into account the hyper-sphere diameter and its expansion speed. The inertial momentum component varies based on these factors. This approach is described in detail in Section 2.5.1.

$$\vec{V_i} = \left[\left(\frac{\delta}{\delta_{max}} + \frac{S_s}{S_{max}}\right) \cdot \omega \cdot \vec{V_i}\right] + c_1 \cdot \vec{R_1} \cdot \left(\vec{P_G} - \vec{X_i}\right) + c_2 \cdot \vec{R_2} \cdot \left(\vec{P_i} - \vec{X_i}\right) \tag{11}$$

The initialization of LS_u and LS_l involves determining LS_{ui} using Equation (12). LS_u is assigned the value of LS_{ui} (Equation (13)), and LS_l is obtained using Equation (14).

$$LS_{ui} = \lambda(L_l - L_u) \tag{12}$$

$$LS_u = LS_{ui} \tag{13}$$

$$LS_l = -LS_u \tag{14}$$

The λ coefficient modifies the starting limits of speed depending on the size of the search space, then speed limits LS_u and LS_l change according to the speed of expansion S_s, obtaining the maximum velocity allowed when $S_s = S_{max}$.

DSRegPSO recalculates LS_u and LS_l each iteration because maximum velocities vary during the algorithm depending on S_s (birds' desperation for food or the expansion speed described in Section 2.5.1), allowing it to break barriers in local optimums. After all, more significant speeds increase the exploration range while lower speeds increase exploitation, as described in [51]. Thus, LS_u and LS_l are updated with Equations (14) and (15), respectively.

$$LS_u = S_s \cdot LS_u \tag{15}$$

Once DSRegPSO recalculates the speeds of the i particles and retains them under its limits, Equation (16) updates the position of particles, randomly varying the components of the particles upper the hyper-sphere diameter δ.

$$\vec{X_i} = (1 - U_i) \cdot \left(\vec{X_i} + \vec{V_i} \right) + U_i \cdot \psi \tag{16}$$

$\vec{\psi_i}$ is a random array under the limits of position $[L_l, L_u]$. $\vec{\psi_i}$ reinvigorates the swarm by changing the components in the position vector according to U_i, which indicates if a particle is too near to $\vec{P_G}$, depending on the hyper-sphere diameter δ, as in Equation (17).

$$U_i(\delta_i) = \begin{cases} 1 & \delta_i \leq \delta \\ 0 & \delta_i > \delta \end{cases} \tag{17}$$

with the distance between the particle and the best position (δ_i) determined with Equation (18).

$$\delta_i = \sum \left(\left| \vec{X_{i,d}} - \vec{P_G} \right| \right) \tag{18}$$

The sphere diameter δ is determined in each iteration using Equation (19) when there is insufficient improvement in G_B at that iteration; in other words, $|G_B(k) - G_B(k-1)| \leq \zeta \cdot G_B(k)$, with ζ as the improvement factor. Additionally, if δ has reached its maximum, i.e., if $\delta = \delta_{max}$, then we set $\delta = \delta_{min}$ and S_s obtains its value according to Equation (20). In the case that there is enough improvement, then we set $\delta = \delta_{min}$ and $S_s = S_{min}$ to maximize exploitation.

$$\delta = \begin{cases} \delta = \delta + \delta_{max} \cdot S_s & \delta < \delta_{max} \\ \delta_{min} & else \end{cases} \tag{19}$$

$$S_s = \begin{cases} S_s + S_{min} & S_s < S_{max} \\ S_{min} & else \end{cases} \tag{20}$$

After obtaining all the parameters in Equations (15) to (20) and the speed of particles in Equation (11), then Equation (21) updates the position of particles with the proposed conservation of momentum to retain all positions under the limits $[L_l, L_u]$. In GPSO, rearrangement of position sets the position value to either L_l, if crossing the lower boundary, or L_u, if crossing the upper boundary. However, in this work, we propose the conservation of momentum principle for PSO, making particles change their direction backward when crossing the boundaries. This conservational momentum maximizes exploration by returning particles to different positions in the search space if the speed vector makes them travel beyond the search space boundaries.

$$\vec{X_{i,d}} = \begin{cases} max\left(L_l, L_u + \left(L_u - \vec{X_{i,d}} \right) \right) & \vec{X_{i,d}} > L_u \\ min\left(L_u, L_l + \left(L_l - \vec{X_{i,d}} \right) \right) & \vec{X_{i,d}} < L_l \end{cases} \tag{21}$$

We let δ_{max} and δ_{min} vary depending on $\vec{P_G}$, progressively reducing the size of the search space around the best position since the components of a position refine across iterations, and then we reduce the search space based on that value. However, we let the user control that refinement with fd_{max} and fd_{min}, as in Equations (22) and (23).

$$\delta_{\max} = \max\left(\vec{P_G} \right) \cdot fd_{max} \tag{22}$$

$$\delta_{min} = \min\left(\vec{P_G} \right) \cdot fd_{min} \tag{23}$$

Finally, we update the new speed limits with Equations (24) and (25), and the process iterates continually while $it < it_{max}$ or $G_B \leq C_d$.

Algorithm 3 DSRegPSO

Data: D, n, $f(\overrightarrow{X_i})$, L_l, L_u, c_1, c_2, M_{max}, λ, fd_{max}, fd_{min}, S_{max}, S_{min}, ζ
Result: $\overrightarrow{P_G}$
$LS_{ui} = \lambda(L_u - L_l)$;
$LS_u = LS_{ui}$;
$LS_l = -LS_u$;
$\omega = \frac{M_{max}}{2}$;
$\overrightarrow{X_i} = \overrightarrow{\psi_i} \in [L_l, L_u]$;
$\overrightarrow{V_i} = 0$;
$C_i = f(\overrightarrow{X_i})$;
$G_{P,i} = C_i$;
$\overrightarrow{P_i} = \overrightarrow{X_i}$;
$G_B = min(G_{P,i})$;
$\overrightarrow{P_G} = argmin\left(f(\overrightarrow{X_i})\right)$;
$S_s = S_{min}$;
$\delta_{\max} = \max(\overrightarrow{P_G}) \cdot fd_{max}$;
$\delta_{min} = \min(\overrightarrow{P_G}) \cdot fd_{min}$;
$\delta = \delta_{min}$;
$it = 0$;
while $k < k_{max}$ **or** $G_B \le C_d$ **do**
  $\overrightarrow{V_i} = \left[\left(\frac{\delta}{\delta_{max}} + \frac{s}{S_{max}}\right) \cdot \omega \cdot \overrightarrow{V_i}\right] + c_1 \cdot R_1 \cdot \left(\overrightarrow{P_G} - \overrightarrow{X_i}\right) + c_2 \cdot R_2 \cdot \left(\overrightarrow{P_i} - \overrightarrow{X_i}\right)$;
  $\overrightarrow{V_i} = \overrightarrow{V_i} \in [LS_l, LS_u]$;
  $\delta_i = \sum\left(\left|\overrightarrow{X_{i,d}} - \overrightarrow{P_G}\right|\right)$;
  $U_i(\delta_i) = \begin{cases} 0 & \delta_i \le \delta \\ 1 & \delta_i > \delta \end{cases}$;
  $\overrightarrow{X_i} = (1 - U_i) \cdot \left(\overrightarrow{X_i} + \overrightarrow{V_i}\right) + U_i \cdot \psi$;
  $\overrightarrow{X_{i,d}} = \begin{cases} max\left(L_l, L_u + \left(L_u - \overrightarrow{X_{i,d}}\right)\right) & \overrightarrow{X_{i,d}} > L_u \\ min\left(L_u, L_l + \left(L_l - \overrightarrow{X_{i,d}}\right)\right) & \overrightarrow{X_{i,d}} < L_l \end{cases}$;
  $\overrightarrow{X_i} = \overrightarrow{X_i} \in [L_l, L_u]$;
  $C_i = f(\overrightarrow{X_i})$;
  for $i \leftarrow 1$ **to** n **do**
    if $C_i < G_{P,i}$ **then**
      $G_{P,i} = C_i$;
      $\overrightarrow{P_i} = \overrightarrow{X_i}$;
    end
    if $G_{P,i} < G_B$ **then**
      $G_B = G_{P,i}$;
      $\overrightarrow{P_G} = \overrightarrow{P_i}$;
    end
  end
  $it = it + 1$;
  if $|G_B(k) - G_B(k-1)| \le \zeta \cdot G_B(k)$ **then**
    if $\delta < \delta_{max}$ **then**
      $\delta = \delta + \delta_{max} \cdot S_s$;
    else
      $\delta = \delta_{\min}$;
      if $S_s \ge S_{max}$ **then**
        $S_s = S_{\min}$;
      else
        $S_s = S_s + S_{min}$;
      end
    end
  else
    $\delta = \delta_{\min}$;
    $S_s = S_{\min}$;
  end
  $\delta_{\max} = \max(\overrightarrow{P_G}) \cdot fd_{max}$;
  $\delta_{min} = \min(\overrightarrow{P_G}) \cdot fd_{min}$;
  $LS_u = \frac{\left(\frac{\delta}{\delta_{max}} + \frac{S_s}{S_{max}}\right)}{2} \cdot LS_{ui}$;
  $LS_l = -LS_u$;
end

$$LS_u = \frac{\left(\frac{\delta}{\delta_{max}} + \frac{S_S}{S_{max}}\right)}{2} \cdot LS_{ui} \tag{24}$$

$$LS_l = -LS_u \tag{25}$$

The DSRegPSO technique is presented in Algorithm 3, which applies the previous formulas while incorporating additional input parameters, namely, fd_{max}, fd_{min}, S_{max}, S_{min}, and ζ. Unlike the original PSO approach, LS_l and LS_u do not need to be specified as input parameters.

3. Results and Discussion

In this section, the proposed algorithm's performance is evaluated by testing it on the 15 functions in benchmark CEC'13, as specified in [52]. The aim of the tests is to verify the algorithm's convergence capabilities and robustness by performing several runs for every function optimized, as described in [52]. The leading convergence indicator in the CEC'13 test is the mean of the best costs, while the Standard Deviation (SD) indicates the robustness. The benchmark runs 25 times during 3.0E+06 function evaluations, and each function has 1000 dimensions. To compare DSRegPSO with other heuristics, we used the Toolkit for Automatic Comparison of Optimizers (TACO), which is also distributed by CEC and described in [53]. The comparison is based on the mean cost obtained after 25 runs of 3.0E+06 function evaluations of the 15 functions in the benchmark.

We set the DSRegPSO (Algorithm 3) decision variables or input parameters depending on the optimization requirements for each test, and the best performance obtained according to:

- $D, n, f\left(\vec{X_i}\right)$, L_l, and L_u were specified by the requirements for the optimized functions in each function of CEC'13.
- We assumed that the remaining input parameters are linearly independent. Based on this assumption, we chose the values that resulted in the best cost for each benchmark by varying them heuristically within the ranges specified in Section 3.1.

Additionally, we tested the original GPSO in Algorithm 2 with CEC'13. Again, we used the same process to select the best input parameters heuristically. Then, we included the results in the comparison.

The computer used for data analysis and result generation was an Alienware m17 R2 running Microsoft Windows 10 Home, version 10.0.19045, build 19045. The system was configured with an Intel(R) Core(TM) i7-9750H CPU @ 2.60 GHz, 16.0 GB of physical RAM, and a total of 39.8 GB of virtual memory

3.1. Results of the CEC'13 Test

The 15 functions utilized for testing in CEC'13 are categorized as fully separable, partially additively separable, overlapping, or non-separable. Each function boasts a unique domain and global optimum. To ensure accuracy, CEC'13 applies transformations such as position translation, rotation, and disturbance operations to the objective function prior to evaluation, as noted in [52].

1. Fully separable functions:
   a. f_1:= Elliptic with $\vec{X} \in [-100, 100]$ and $f_1\left(\vec{X^{\text{opt}}}\right) = 0$.
   b. f_2:= Rastrigin with $\vec{X} \in [-5, 5]$ and $f_2\left(\vec{X^{\text{opt}}}\right) = 0$.
   c. f_3:= Ackley with $\vec{X} \in [-32, 32]$ and $f_3\left(\vec{X^{\text{opt}}}\right) = 0$.
2. Partially Additively Separable Functions:
   - Functions with a separable subcomponent:

a. f_4:= Elliptic with $\vec{X} \in [-100, 100]$ and $f_4\left(\vec{X^{\text{opt}}}\right) = 0$.

b. f_5:= Rastrigin with $\vec{X} \in [-5, 5]$ and $f_5\left(\vec{X^{\text{opt}}}\right) = 0$.

c. f_6:= Ackley with $\vec{X} \in [-32, 32]$ and $f_6\left(\vec{X^{\text{opt}}}\right) = 0$.

d. f_7:= Schwefels Problem 1.2 with $\vec{X} \in [-100, 100]$ and $f_7\left(\vec{X^{\text{opt}}}\right) = 0$.

- Functions with no separable subcomponents:

a. f_8:= Elliptic with $\vec{X} \in [-100, 100]$ and $f_8\left(\vec{X^{\text{opt}}}\right) = 0$.

b. f_9:= Rastrigin with $\vec{X} \in [-5, 5]$ and $f_9\left(\vec{X^{\text{opt}}}\right) = 0$.

c. f_{10}:= Ackley with $\vec{X} \in [-32, 32]$ and $f_{10}\left(\vec{X^{\text{opt}}}\right) = 0$.

d. f_{11}:= Schwefels Problem 1.2 with $\vec{X} \in [-100, 100]$ and $f_{11}\left(\vec{X^{\text{opt}}}\right) = 0$.

3. Overlapping Functions:

a. f_{12}:= Rosenbrock's with $\vec{X} \in [-100, 100]$ and $f_{12}\left(\vec{X^{\text{opt}}} + 1\right) = 0$.

b. f_{13}:= Schwefels with Conforming Overlapping Subcomponents with $\vec{X} \in [-100, 100]$ and $f_{13}\left(\vec{X^{\text{opt}}}\right) = 0$.

c. f_{14}:= Schwefels with Conflicting Overlapping Subcomponents with $\vec{X} \in [-100, 100]$ and $f_{14}\left(\vec{X^{\text{opt}}}\right) = 0$.

4. Non-separable Functions:

a. f_{15}:= Schwefels Problem 1.2 with $\vec{X} \in [-100, 100]$ and $f_{15}\left(\vec{X^{\text{opt}}}\right) = 0$.

The DSRegPSO algorithm can use a heuristic approach to find values for fd_{min}, fd_{max}, ζ, c_2, M_{max}, λ, S_{max}, S_{min}, and n. The values that were heuristically tested are provided below. The limits of the parameters related to DSRegPSO: fd_{min}, fd_{max}, ζ, S_{max}, and S_{min} were determined by increasing or decreasing those limits in order to achieve better results. For the parameters related to PSO: M_{max}, λ, c_2, and n, we used limits that were recommended in the literature, specifically in [54,55].

- $fd_{min} = \{1E-200, 1E-100, 1E-50, 1E-25, 1E-10, 1E-5, 1E-1, 1\}$.
- $fd_{max} = \{1E-200, 1E-100, 1E-50, 1E-25, 1E-10, 1E-5, 1E-1, 1\}$.
- $\zeta = \{1E-5, 1E-4, 1E-3, 1E-2, 5E-2, 1E-1, 5E-1\}$.
- $M_{max} = \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3\}$.
- $c_2 = \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.5, 2.0\}$.
- $\lambda = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3\}$.
- $S_{max} = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$.
- $S_{min} = \{0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1\}$.
- $n = \{1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$.

Table 1 displays the optimal parameters for the DSRegPSO algorithm based on the average best cost after being heuristically evaluated for five runs of 5.0E+05 function evaluations per CEC'13 function, in order to obtain the best possible cost per function. We tested the algorithm parameters using 5.0E+05 function evaluations instead of 3.0E+06, as the latter would not be feasible due to the complexity of the CEC'13 test.

Table 1. DSRegPSO parameters determined with heuristic selection in the CEC'13 test.

$f\left(\vec{X}\right)$	fd_{min}	fd_{max}	ζ	c_2	M_{max}	λ	S_{max}	S_{min}	n
f_1	1.0E − 200	1.0E − 200	5.0E − 02	1.5E + 00	0.0E + 00	7.0E − 01	1.5E + 00	2.0E − 02	5.0E + 01
f_2	1.0E − 50	1.0E − 01	1.0E − 02	1.5E + 00	1.0E − 01	1.9E + 00	5.0E − 01	5.0E − 02	5.0E + 00
f_3	1.0E − 100	1.0E − 05	5.0E − 01	1.5E + 00	1.0E − 01	2.0E − 01	1.0E − 01	5.0E − 02	2.0E + 01
f_4	1.0E − 50	1.0E − 01	1.0E − 03	1.0E + 00	0.0E + 00	1.0E + 00	5.0E − 01	5.0E − 02	2.0E + 01
f_5	1.0E − 50	1.0E + 00	5.0E − 02	2.0E + 00	1.3E + 00	1.3E + 00	3.0E − 01	5.0E − 02	4.0E + 01
f_6	1.0E − 50	1.0E − 25	1.0E − 03	8.0E − 01	7.0E − 01	7.0E − 01	8.0E − 01	8.0E − 02	5.0E + 01
f_7	1.0E − 50	1.0E − 10	1.0E − 02	1.5E + 00	4.0E − 01	3.0E − 01	9.0E − 01	5.0E − 02	3.0E + 01
f_8	1.0E − 25	1.0E − 10	5.0E − 02	6.0E − 01	6.0E − 01	4.0E − 01	4.0E − 01	5.0E − 02	5.0E + 01
f_9	1.0E − 25	1.0E − 25	1.0E − 01	2.0E + 00	1.2E + 00	1.3E + 00	3.0E − 01	5.0E − 02	3.0E + 01
f_{10}	1.0E − 25	1.0E − 01	1.0E − 03	8.0E − 01	3.0E − 01	7.0E − 01	5.0E − 01	5.0E − 02	5.0E + 01
f_{11}	1.0E − 50	1.0E + 00	1.0E − 02	1.3E + 00	2.0E − 01	5.0E − 01	5.0E − 01	4.0E − 02	3.0E + 01
f_{12}	1.0E − 25	1.0E − 01	1.0E − 02	1.0E − 01	3.0E − 01	1.3E + 00	1.0E − 01	1.0E − 01	1.0E + 00
f_{13}	1.0E − 50	1.0E − 01	1.0E − 02	1.3E + 00	3.0E − 01	5.0E − 01	5.0E − 01	5.0E − 02	3.0E + 01
f_{14}	1.0E − 25	1.0E + 00	5.0E − 01	1.0E + 00	5.0E − 01	4.0E − 01	5.0E − 01	5.0E − 02	4.0E + 01
f_{15}	1.0E − 50	1.0E − 25	1.0E − 02	1.3E + 00	4.0E − 01	6.0E − 01	9.0E − 01	5.0E − 02	3.0E + 01

Similarly, we select the c_1, c_2, w_{max}, λ, and n input parameters of the GPSO with a heuristic approach. The values heuristically tested are below. Again, in the case that the best parameter found corresponds to the lower or upper value, we tested extra values by increasing or decreasing those limits with the same step:

- $c_1 = \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.5, 2.0\}$.
- $c_2 = \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.5, 2.0\}$.
- $w_{max} = \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3\}$.
- $\lambda = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3\}$.
- $n = \{1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$.

Table 2 shows the best parameters for GPSO based on the mean best cost after heuristically testing them against five runs of 5.0E + 05 function evaluations for each CEC'13 function.

Table 2. GPSO parameters determined with heuristic selection in the CEC'13 test.

$f\left(\vec{X}\right)$	c_1	c_2	w_{max}	λ	n
f_1	2.0E + 00	1.5E + 00	5.0E − 01	1.0E − 01	8.0E + 01
f_2	2.0E + 00	2.0E + 00	0.0E + 00	1.3E + 00	1.0E + 02
f_3	2.0E + 00	1.0E + 00	7.0E − 01	1.0E − 01	3.0E + 01
f_4	1.5E + 00	1.5E + 00	7.0E − 01	1.0E − 01	8.0E + 01
f_5	1.2E + 00	2.0E + 00	7.0E − 01	2.0E − 01	7.0E + 01
f_6	1.5E + 00	8.0E − 01	7.0E − 01	1.0E − 01	7.0E + 01
f_7	1.5E + 00	1.3E + 00	7.0E − 01	2.0E − 01	5.0E + 01
f_8	1.3E + 00	2.0E + 00	7.0E − 01	1.0E − 01	5.0E + 01
f_9	2.0E + 00	1.0E + 00	7.0E − 01	1.0E − 01	9.0E + 01
f_{10}	2.0E + 00	1.0E + 00	6.0E − 01	7.0E − 01	5.0E + 01
f_{11}	1.5E + 00	1.5E + 00	7.0E − 01	1.0E − 01	8.0E + 01
f_{12}	1.0E − 01	2.0E + 00	1.0E − 01	1.0E − 01	1.0E + 02
f_{13}	1.5E + 00	1.5E + 00	7.0E − 01	1.0E − 01	5.0E + 01
f_{14}	1.5E + 00	1.5E + 00	7.0E − 01	1.0E − 01	9.0E + 01
f_{15}	1.5E + 00	1.0E + 00	7.0E − 01	1.0E − 01	8.0E + 01

The results obtained using the DSRegPSO and the GPSO using the selected input parameters in the CEC'13 benchmark with 1000 dimensions for 3.0E + 06 function evaluations and 25 runs are shown below in Table 3. The average time per iteration varies depending

on the functions and the algorithm. Despite the proposed modifications to PSO, the time per iteration remains below 100 µs.

Table 3. DSRegPSO mean, SD, worst, and best results using the CEC'13 benchmark.

$f\left(\vec{X}\right)$	Average Time per Iteration in Seconds		Mean		SD		Worst		Best	
Algorithms	**DSRegPSO**	**GPSO**	**DSRegPSO**	**GPSO**	**DSRegPSO**	**GPSO**	**DSRegPSO**	**GPSO**	**DSRegPSO**	**GPSO**
f_1	3.16E − 05	1.02E − 05	4.07E − 04	1.44E + 10	1.73E − 04	1.55E + 10	1.07E − 03	4.63E + 10	1.90E − 04	2.91E + 09
f_2	4.44E − 05	1.19E − 05	8.63E + 02	4.36E + 04	1.29E + 02	8.31E + 02	1.16E + 03	4.54E + 04	6.87E + 02	4.19E + 04
f_3	4.79E − 05	1.33E − 05	2.00E + 01	2.03E + 01	1.10E − 09	2.74E − 02	2.00E + 01	2.03E + 01	2.00E + 01	2.02E + 01
f_4	4.44E − 05	1.26E − 05	2.15E + 09	8.63E + 10	7.17E + 08	5.17E + 10	3.84E + 09	2.06E + 11	1.24E + 09	2.75E + 10
f_5	4.95E − 05	1.46E − 05	7.76E + 06	8.38E + 06	2.64E + 06	1.79E + 06	1.45E + 07	1.35E + 07	3.76E + 06	5.66E + 06
f_6	4.72E − 05	1.10E − 05	1.01E + 06	1.03E + 06	1.26E + 04	7.76E + 03	1.04E + 06	1.04E + 06	9.96E + 05	1.01E + 06
f_7	2.24E − 05	7.74E − 06	5.85E + 04	3.84E + 09	1.26E + 04	3.29E + 09	9.24E + 04	1.79E + 10	3.59E + 04	5.73E + 08
f_8	9.60E − 05	1.50E − 05	3.34E + 13	4.24E + 14	2.31E + 13	3.30E + 14	1.12E + 14	1.80E + 15	1.06E + 13	1.32E + 14
f_9	4.56E − 02	1.77E − 05	4.65E + 08	9.12E + 08	1.93E + 08	1.50E + 08	1.28E + 09	1.26E + 09	3.05E + 08	6.12E + 08
f_{10}	4.01E − 03	1.52E − 05	9.25E + 07	9.20E + 07	5.49E + 05	6.64E + 05	9.39E + 07	9.31E + 07	9.17E + 07	9.07E + 07
f_{11}	4.32E − 05	1.62E − 05	5.42E + 08	1.25E + 11	8.35E + 07	1.06E + 11	7.57E + 08	3.98E + 11	4.32E + 08	3.44E + 09
f_{12}	3.36E − 05	1.10E − 06	2.48E + 03	1.60E + 12	1.36E + 03	3.58E + 10	6.69E + 03	1.67E + 12	1.56E + 03	1.53E + 12
f_{13}	6.72E − 05	6.72E − 06	1.37E + 07	1.18E + 10	2.58E + 06	5.31E + 09	2.02E + 07	2.52E + 10	1.03E + 07	3.03E + 09
f_{14}	4.32E − 05	7.20E − 06	2.47E + 08	1.09E + 11	2.85E + 07	7.19E + 10	3.08E + 08	2.96E + 11	1.92E + 08	7.87E + 09
f_{15}	2.88E − 05	3.96E − 06	6.73E + 05	2.31E + 12	6.04E + 04	3.07E + 12	8.18E + 05	1.44E + 13	5.79E + 05	2.80E + 10

According to the results presented in Table 3, DSRegPSO outperforms GPSO in terms of both achieving the best cost for each function and demonstrating higher stability. This is evident from the fact that DSRegPSO has the best mean, best, SD, and worst values among the two algorithms, and it shows lower SD values in 14 out of the total 15 functions of the CEC'13 test.

Figure 3 shows the performance in the first 25,000 function evaluations for the proposed DSRegPSO in the optimization of f_1 of CEC'13. Figure 4 shows the performance in the entire run with function evaluations on a logarithmic scale. The behaviors of the DSRegPSO in Figures 3 and 4 show that the proposed algorithm continually improves and resets the particle's position without requiring stagnation to reinvigorate the swarm similar to RegPSO (Figure 1).

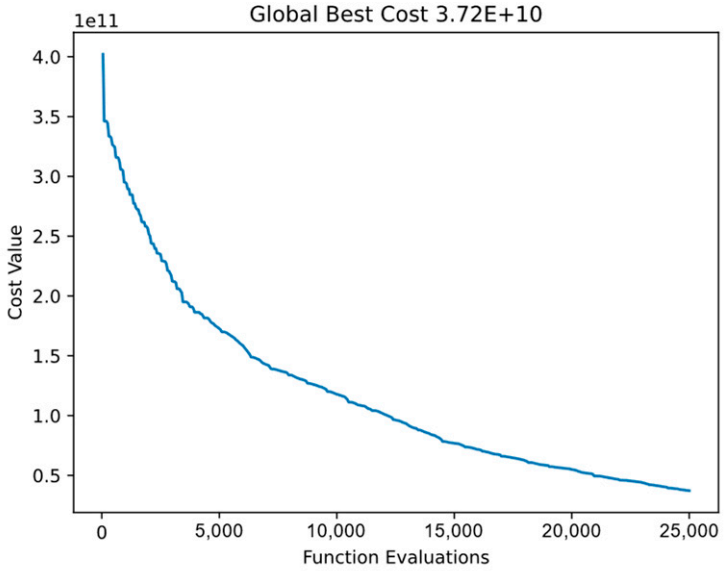

Figure 3. Cost value for the first 25,000 function evaluations with DSRegPSO and f_1 of CEC'13.

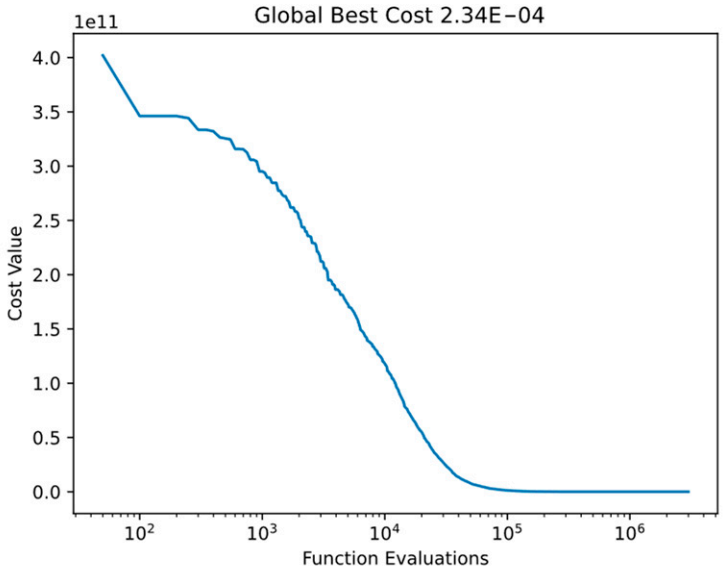

Figure 4. Cost value with the x-axis on a logarithmic scale for the DSRegPSO in f_1 of CEC'13.

We also used Principal Component Analysis (PCA) to condense the particle position data from 1000 to 3 dimensions. This allowed us to create 3D position plots, which helped us visualize the revival of the swarm and prevent stagnation. Analyzing the three principal components provided us with valuable insights into particle behavior and optimized the swarm. Additionally, we have included convergence diagrams for functions 1 to 15 of CEC'13 for DSRegPSO to assess the convergence and stagnation of our proposal. These diagrams show a comparison per run for the 3.00E+06 function evaluations with checkpoints registered at 1.20E+05, 3.00E+05, 6.00E+05, 9.00E+05, 1.20E+06, 1.50E+06, 1.80E+06, 2.10E+06, 2.40E+06, 2.70E+06, and 3.00E+06 function evaluations, as per the default configuration of CEC'13.

Figures 5 and 6 display the convergence and PCA of Function 1 of CEC'13. As depicted in the figures, the algorithm continues to improve even after 3.00E+06 function evaluations and does not converge. Additionally, the comparison among the runs demonstrates the level of stability in the algorithm for this function.

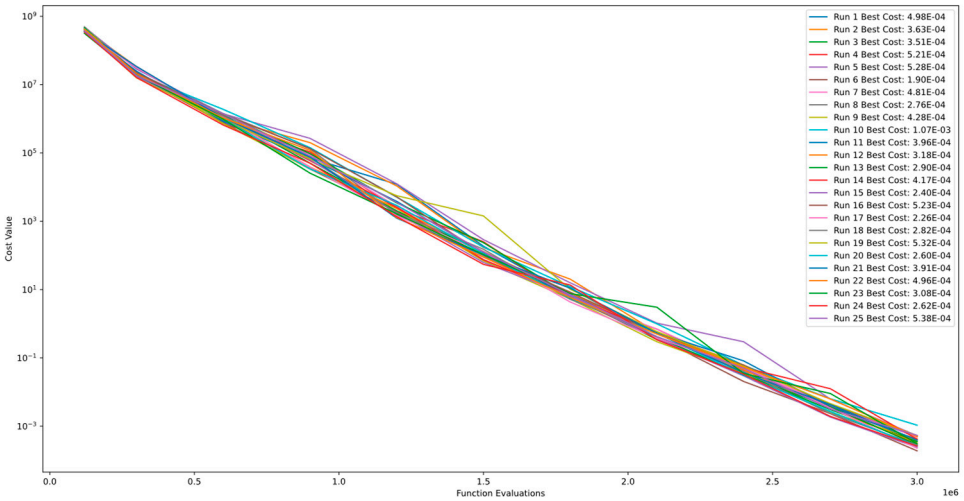

Figure 5. Convergence diagram of Function 1 of CEC'13 for DSRegPSO with 25 runs and 3.00+E06 function evaluations.

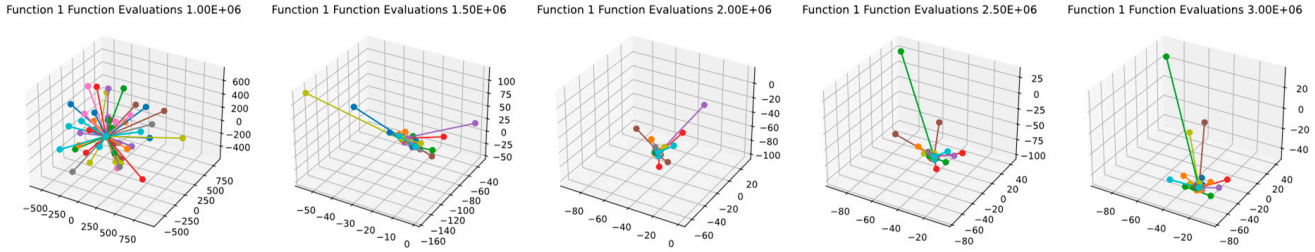

Figure 6. PCA of 50 particle positions in Function 1 of CEC'13 for DSRegPSO.

Figures 7 and 8 showcase the convergence and PCA of Function 2 of CEC'13. As seen in the figures, the algorithm shows continual improvement even after 3.00E+06 function evaluations and does not converge. Moreover, the comparison between runs illustrates the algorithm's stability level for this function.

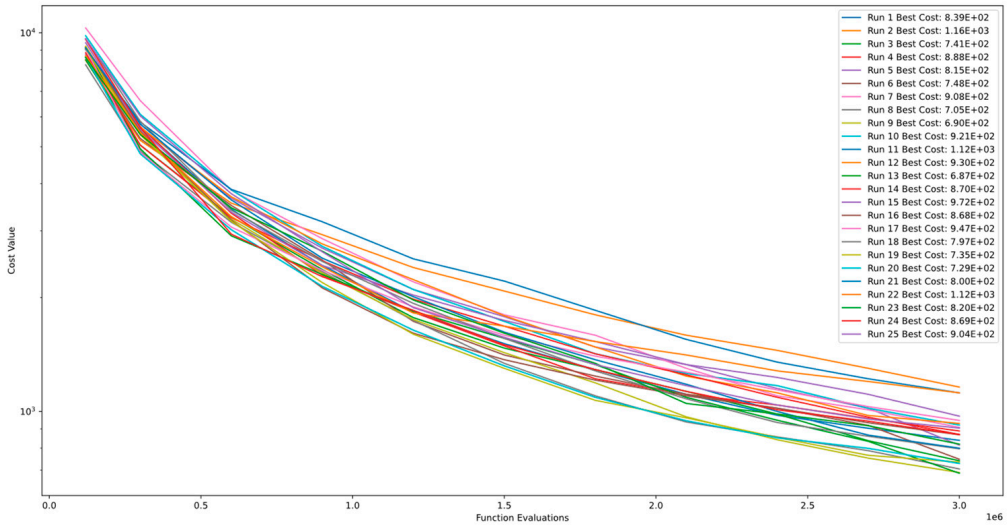

Figure 7. Convergence diagram of Function 2 of CEC'13 for DSRegPSO with 25 runs and 3.00+E06 function evaluations.

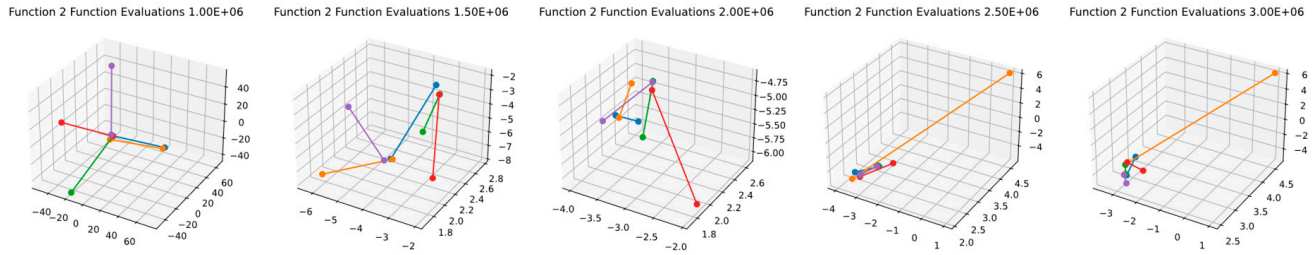

Figure 8. PCA of 5 particle positions in Function 2 of CEC'13 for DSRegPSO.

Figures 9 and 10 presented in CEC'13 highlight the convergence and PCA of Function 3. Figure 9 demonstrates the early convergence of the global best cost, and a comparison between the runs reveals stagnation caused by Function 3 being a pinhole function that cannot be solved with PSO. However, Figure 10 depicts that particles never encounter stagnation or convergence, as indicated by the PCA.

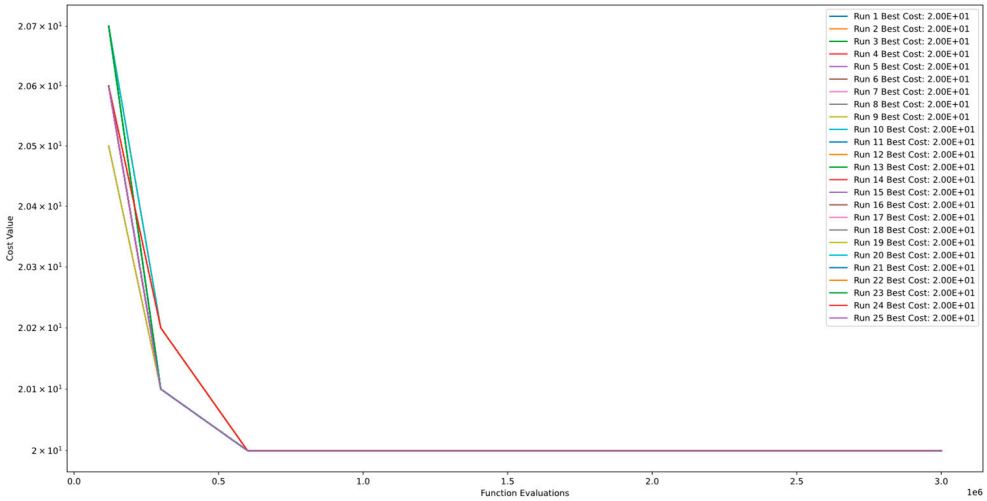

Figure 9. Convergence diagram of Function 3 of CEC'13 for DSRegPSO with 25 runs and 3.00+E06 function evaluations.

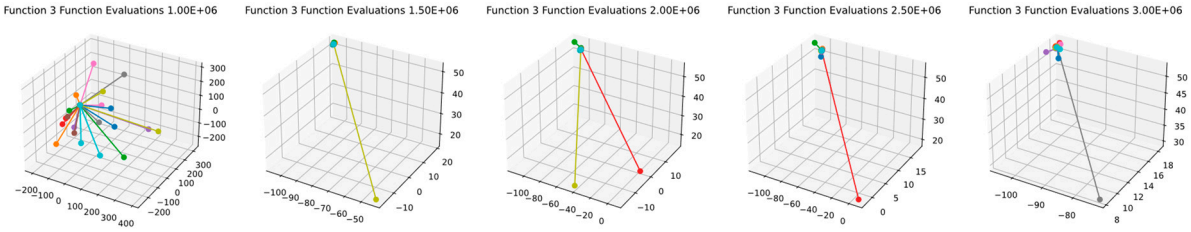

Figure 10. PCA of 20 particle positions in Function 3 of CEC'13 for DSRegPSO.

Figures 11 and 12 demonstrate the convergence and PCA of Function 4 of CEC'13. The figures show that the algorithm exhibits continuous improvement even after 3.00E+06 function evaluations and does not converge. Additionally, the comparison between runs indicates the algorithm's level of stability for this function.

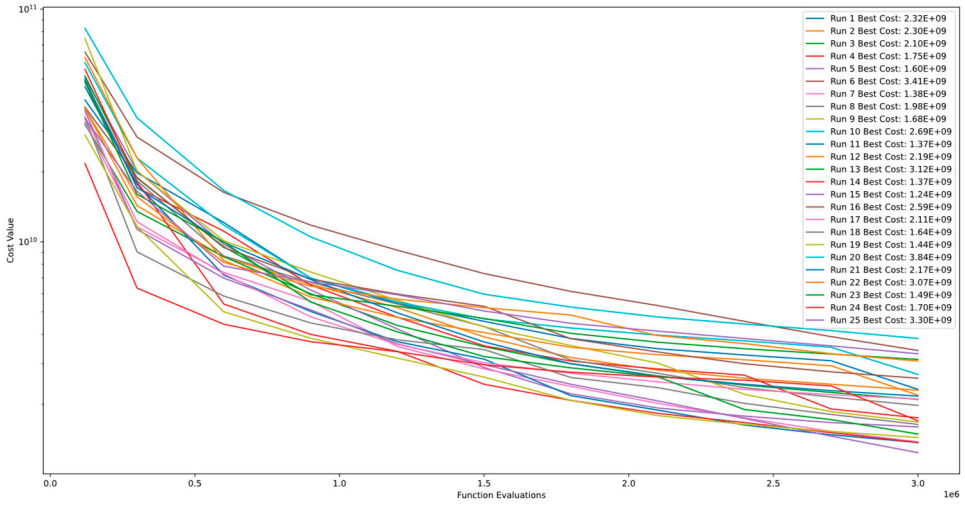

Figure 11. Convergence diagram of Function 4 of CEC'13 for DSRegPSO with 25 runs and 3.00+E06 function evaluations.

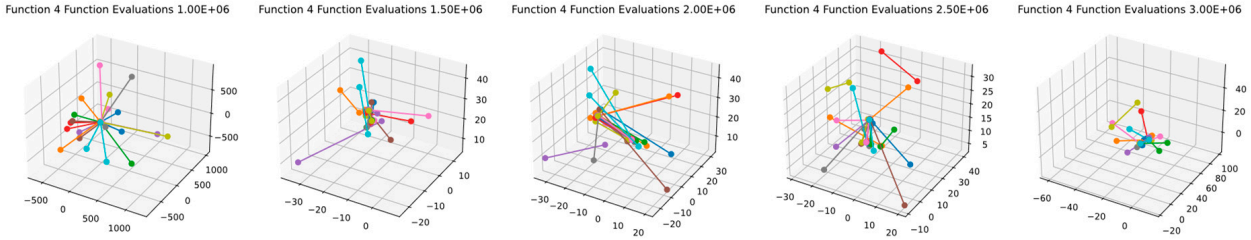

Figure 12. PCA of 20 particle positions in Function 4 of CEC'13 for DSRegPSO.

Figures 13 and 14 showcase the convergence and PCA of Function 5 of CEC'13. These figures indicate that the algorithm continues to improve even after 3.00E+06 function evaluations and does not converge. However, comparing the results of multiple runs highlights that the algorithm's stability level is poor for this function.

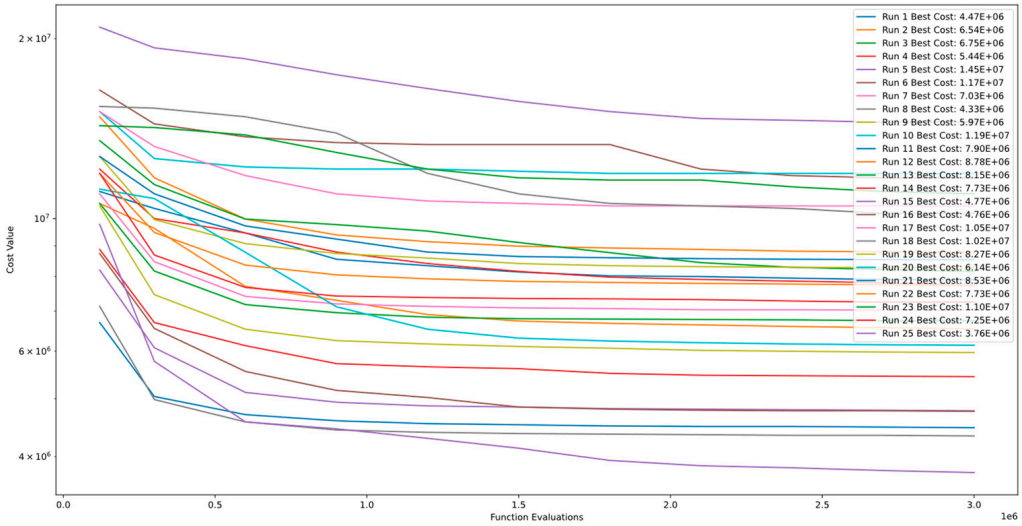

Figure 13. Convergence diagram of Function 5 of CEC'13 for DSRegPSO with 25 runs and 3.00+E06 function evaluations.

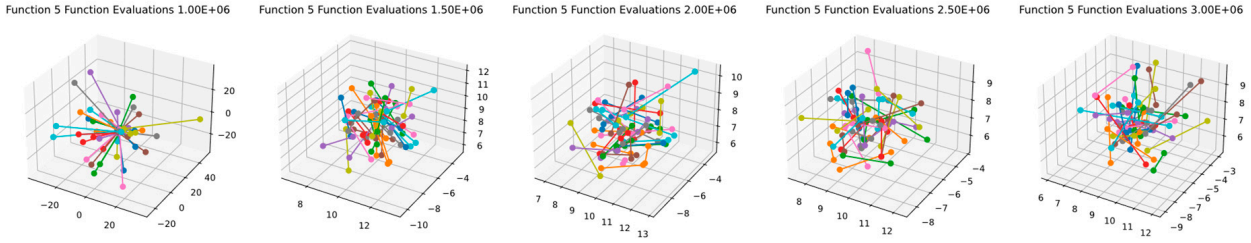

Figure 14. PCA of 40 particle positions in Function 5 of CEC'13 for DSRegPSO.

Figures 15 and 16 demonstrate the convergence and PCA of Function 6 of CEC'13. The figures show that the algorithm exhibits continuous improvement even after 3.00E+06 function evaluations and does not converge. However, comparing the results of multiple runs highlights that the algorithm's stability level is poor.

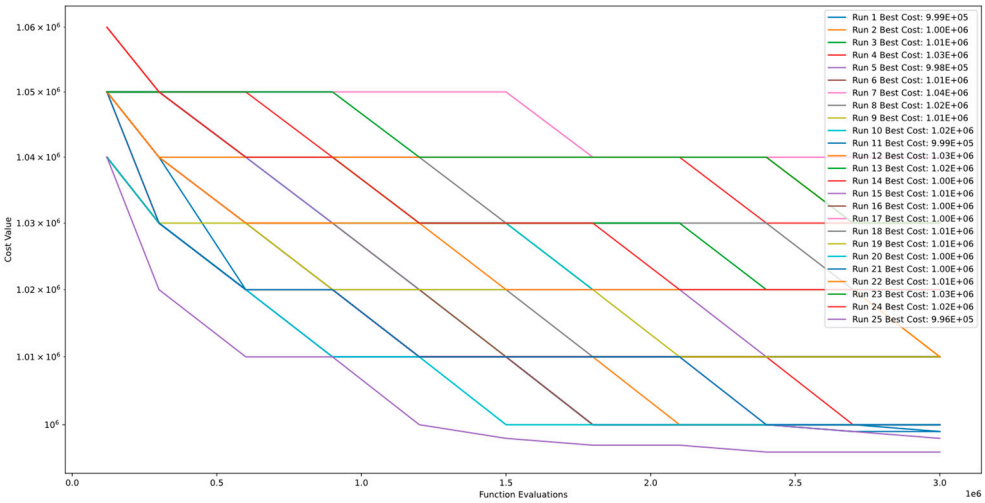

Figure 15. Convergence diagram of Function 6 of CEC'13 for DSRegPSO with 25 runs and 3.00+E06 function evaluations.

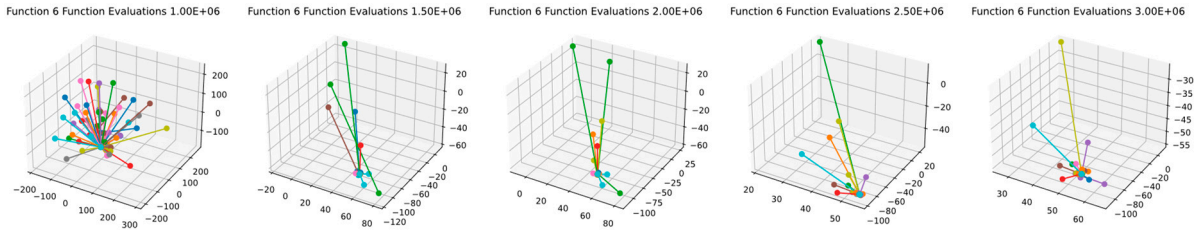

Figure 16. PCA of 50 particle positions in Function 6 of CEC'13 for DSRegPSO.

Figures 17 and 18 demonstrate the convergence and PCA of Function 7 of CEC'13. The figures show that the algorithm exhibits continuous improvement even after 3.00E+06 function evaluations and does not converge. Additionally, the comparison between runs indicates the algorithm's level of stability for this function.

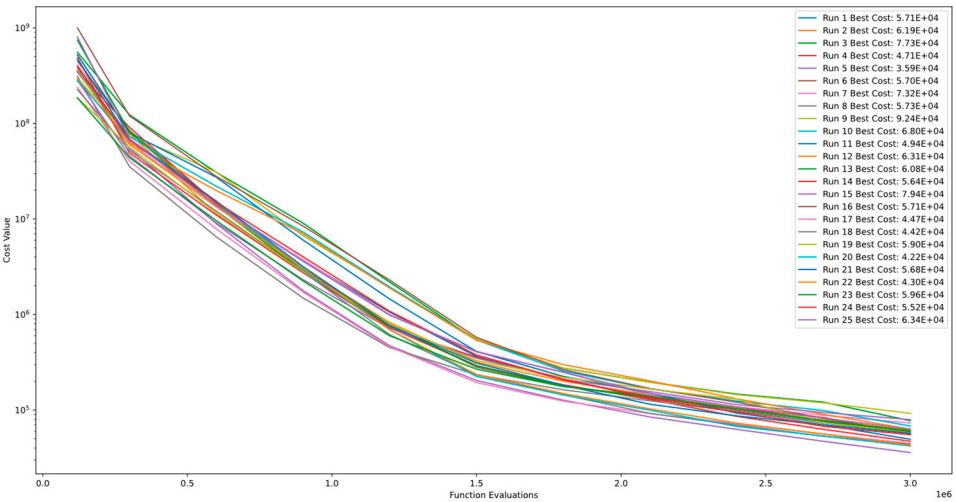

Figure 17. Convergence diagram of Function 7 of CEC'13 for DSRegPSO with 25 runs and 3.00+E06 function evaluations.

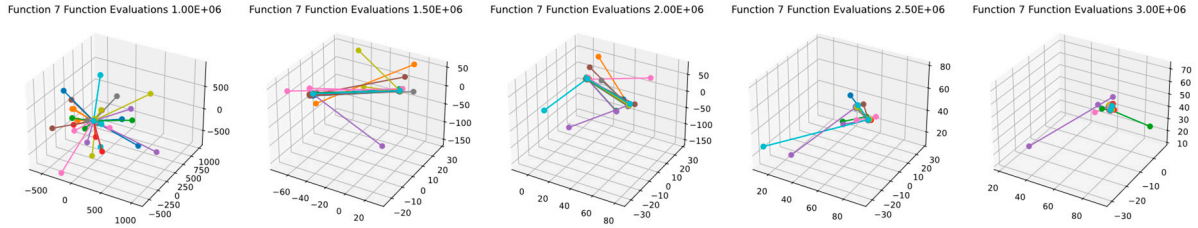

Figure 18. PCA of 30 particles' positions in Function 7 of CEC'13 for DSRegPSO.

Figures 19 and 20 demonstrate the convergence and PCA of Function 8 of CEC'13. The figures show that the algorithm exhibits continuous improvement even after 3.00E+06 function evaluations and does not converge. However, the comparison between runs indicates that the algorithm's level of stability is poor.

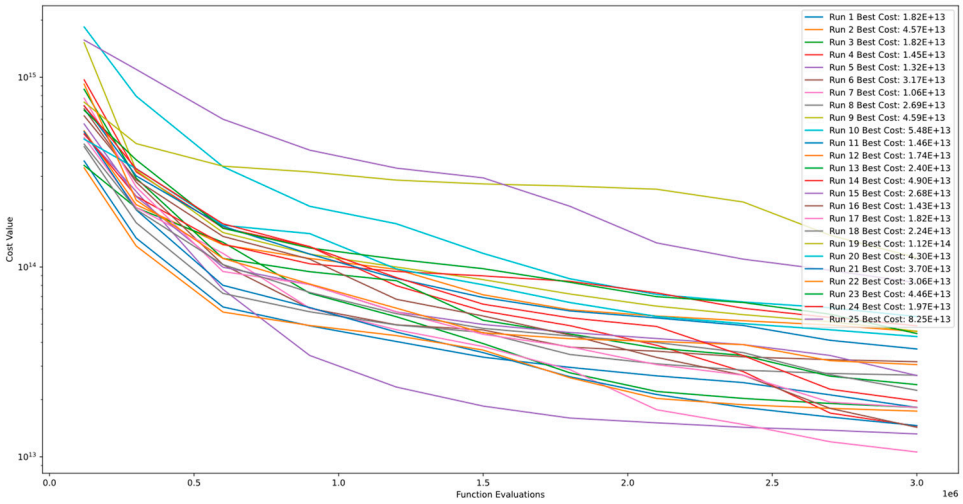

Figure 19. Convergence diagram of Function 8 of CEC'13 for DSRegPSO with 25 runs and 3.00+E06 function evaluations.

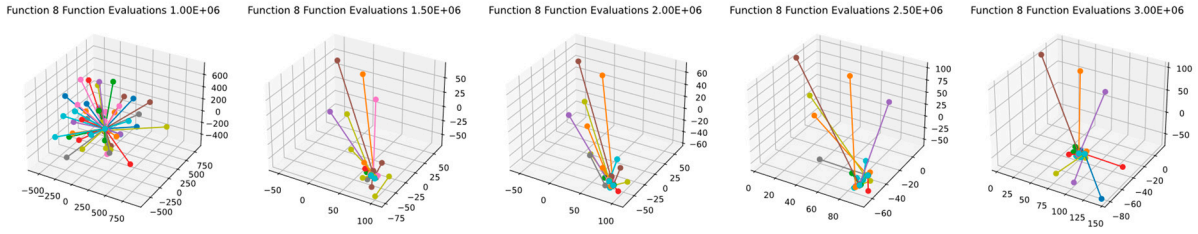

Figure 20. PCA of 50 particle positions in Function 8 of CEC'13 for DSRegPSO.

Figures 21 and 22 showcase the convergence and PCA of Function 9 of CEC'13. These figures indicate that the algorithm continues to improve even after 3.00E+06 function evaluations and does not converge. However, comparing the results of multiple runs highlights that the algorithm's stability level is poor in this function.

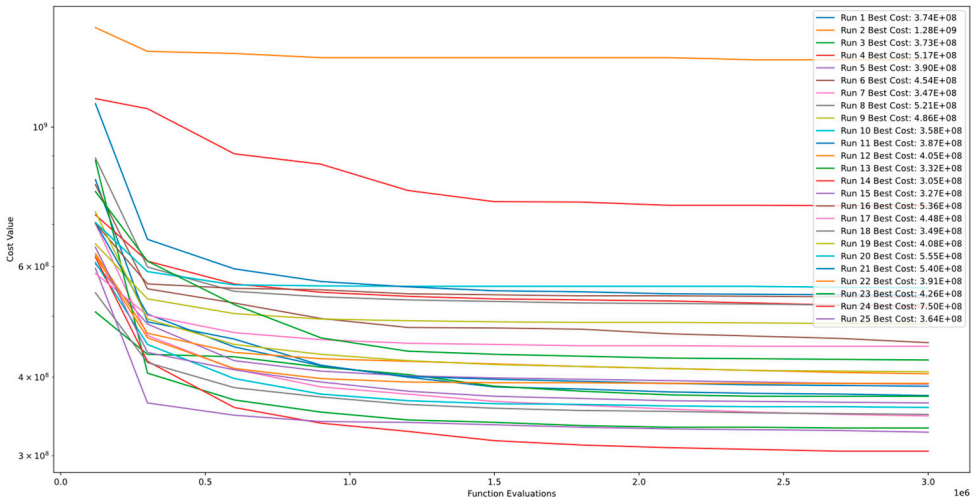

Figure 21. Convergence diagram of Function 9 of CEC'13 for DSRegPSO with 25 runs and 3.00+E06 function evaluations.

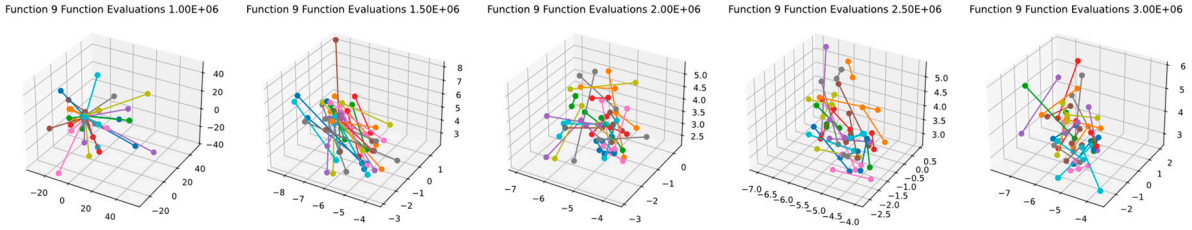

Figure 22. PCA of 30 particle positions in Function 9 of CEC'13 for DSRegPSO.

Figures 23 and 24 demonstrate the convergence and PCA of Function 10 of CEC'13. The figures show that the algorithm exhibits continuous improvement even after 3.00E+06 function evaluations and does not converge. However, the comparison between runs indicates that the algorithm's level of stability is poor.

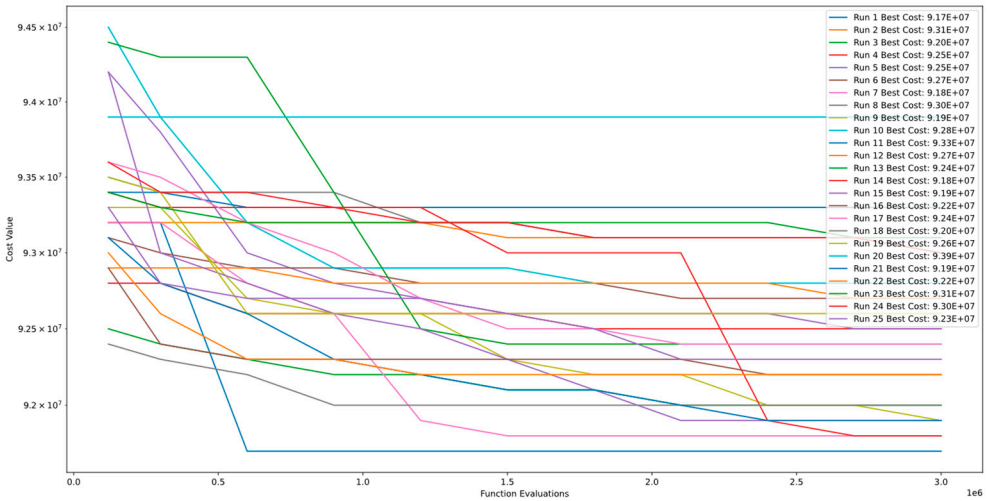

Figure 23. Convergence diagram of Function 10 of CEC'13 for DSRegPSO with 25 runs and 3.00+E06 function evaluations.

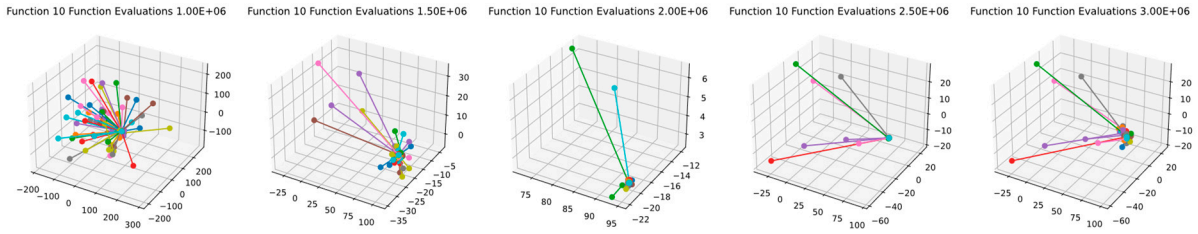

Figure 24. PCA of 50 particle positions in Function 10 of CEC'13 for DSRegPSO.

Figures 25 and 26 demonstrate the convergence and PCA of Function 11 of CEC'13. The figures show that the algorithm exhibits continuous improvement even after 3.00E+06 function evaluations and does not converge. Additionally, the comparison between runs indicates the algorithm's level of stability for this function.

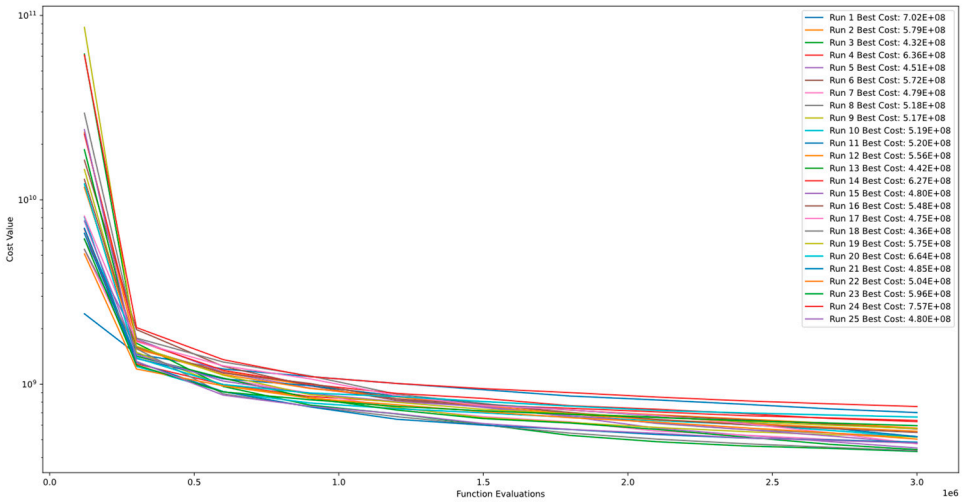

Figure 25. Convergence diagram of Function 11 of CEC'13 for DSRegPSO with 25 runs and 3.00+E06 function evaluations.

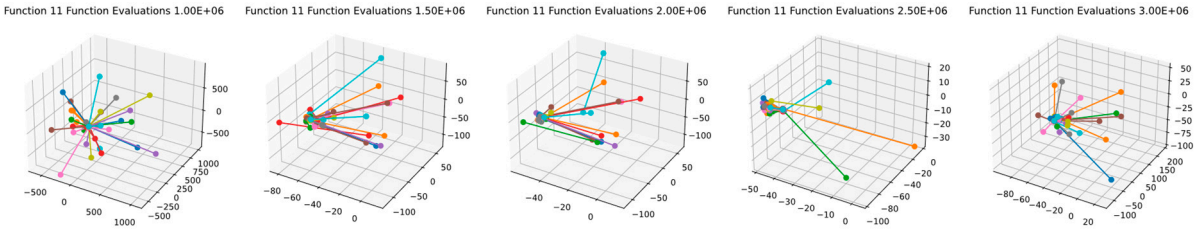

Figure 26. PCA of 30 particle positions in Function 11 of CEC'13 for DSRegPSO.

From Figures 27 and 28, it is evident that Function 12 of CEC'13 does not converge even after 3.00E+06 function evaluations. The PCA was conducted on the results obtained after setting the number of particles to five, as PCA cannot be performed with only one particle. It is worth noting that the algorithm shows continuous improvement, and comparing the runs indicates the stability level of the algorithm for this specific function.

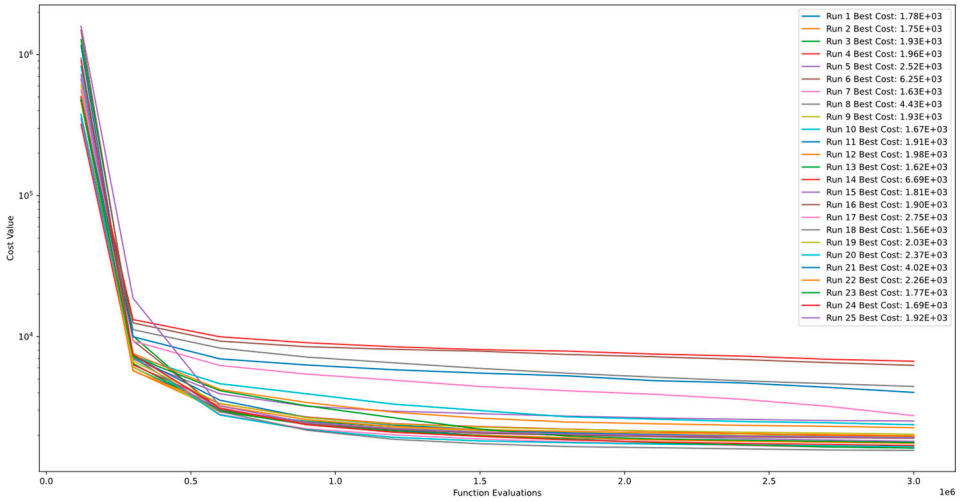

Figure 27. Convergence diagram of Function 12 of CEC'13 for DSRegPSO with 25 runs and 3.00+E06 function evaluations.

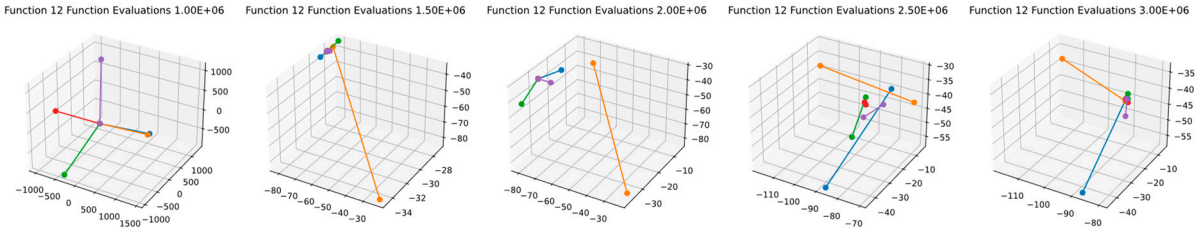

Figure 28. PCA of 5 particle positions in Function 12 of CEC'13 for DSRegPSO.

Figures 29 and 30 demonstrate the convergence and PCA of Function 13 of CEC'13. The figures show that the algorithm exhibits continuous improvement even after 3.00E+06 function evaluations and does not converge. Additionally, the comparison between runs indicates the algorithm's level of stability for this function.

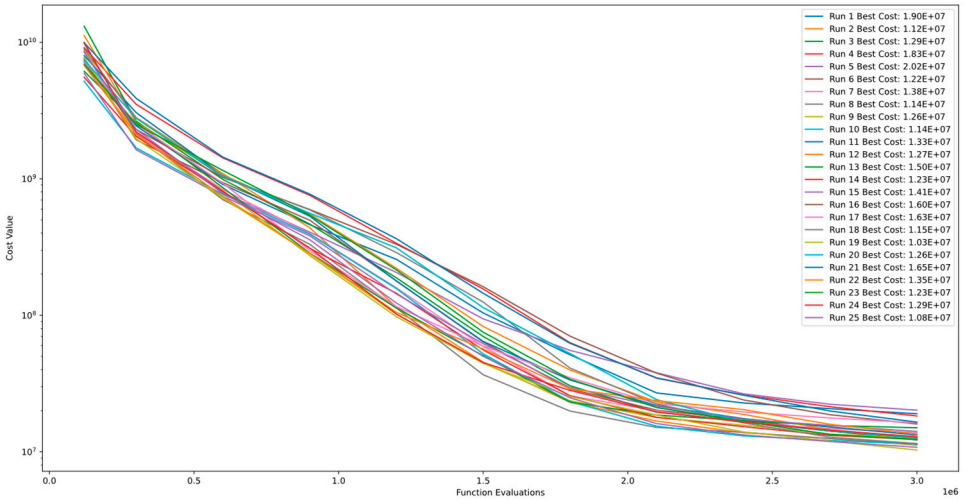

Figure 29. Convergence diagram of Function 13 of CEC'13 for DSRegPSO with 25 runs and 3.00+E06 function evaluations.

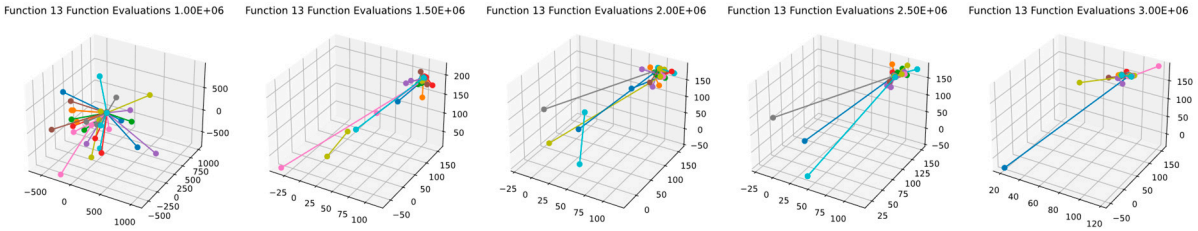

Figure 30. PCA of 40 particle positions in Function 13 of CEC'13 for DSRegPSO.

Figures 31 and 32 demonstrate the convergence and PCA of Function 14 of CEC'13. The figures show that the algorithm exhibits continuous improvement even after 3.00E+06 function evaluations and does not converge. Additionally, the comparison between runs indicates the algorithm's level of stability for this function.

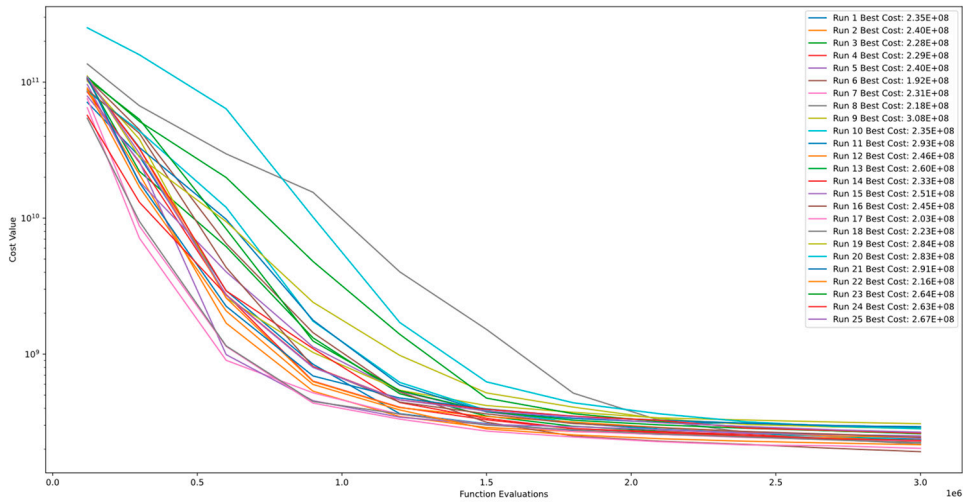

Figure 31. Convergence diagram of Function 14 of CEC'13 for DSRegPSO with 25 runs and 3.00+"E" 06 function evaluations.

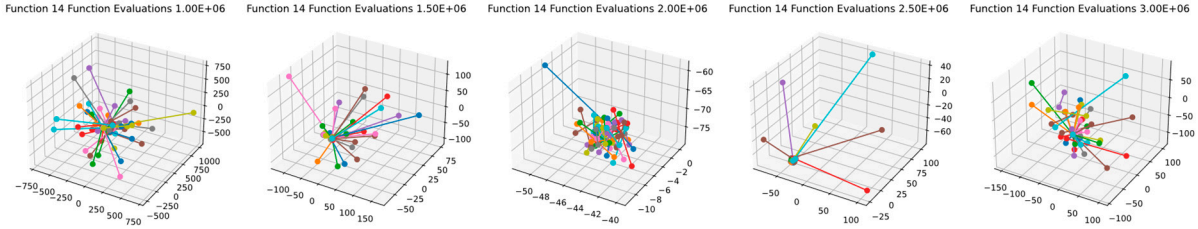

Figure 32. PCA of 40 particle positions in Function 14 of CEC'13 for DSRegPSO.

Figures 33 and 34 demonstrate the convergence and PCA of Function 15 of CEC'13. The figures show that the algorithm exhibits continuous improvement even after 3.00E+06 function evaluations and does not converge. Additionally, the comparison between runs indicates the algorithm's level of stability for this function.

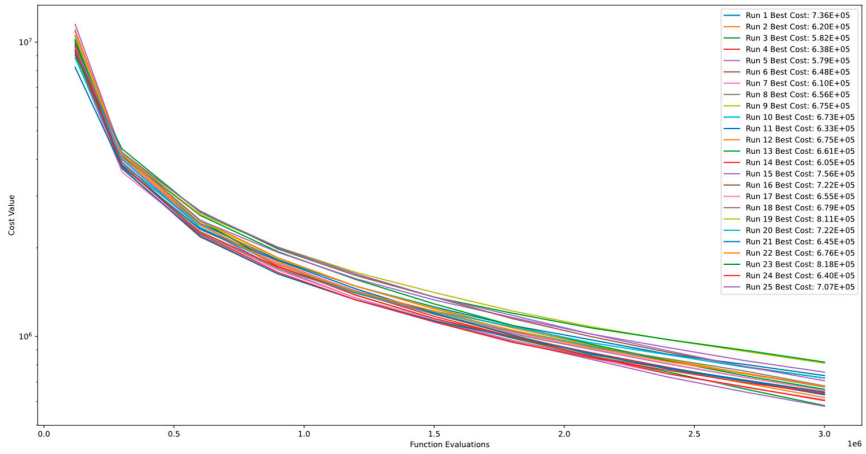

Figure 33. Convergence diagram of Function 15 of CEC'13 for DSRegPSO with 25 runs and 3.00+"E" 06 function evaluations.

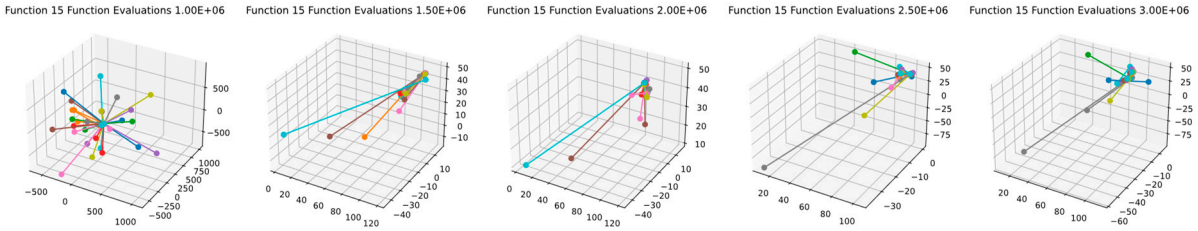

Figure 34. PCA of 30 particle positions in Function 15 of CEC'13 for DSRegPSO.

Table 4 shows the CEC'13 TACO ranking comparison with DSRegPSO based on the best cost obtained after 1.20E + 05, 6.00E + 05, and 3.00E + 06 function evaluations for the 25 runs, where the index registered is the number of functions when the algorithm obtains the best results. The proposed algorithm DSRegPSO obtains the best results in optimizing the non-separable function.

Table 4. Ranking comparison based on mean cost with CEC'13, including the proposed DSRegPSO.

Algorithm	1.20E+05	6.00E+05	3.00E+06
AMO	0	0	0
APO	0	0	0
AQO	0	0	0
BICCA	0	1	1
CC-CMA-ES	0	1	1
DECC-G	6	0	0
DEEPSO	0	0	0
DMO	0	0	0
DPO	0	0	0
DQO	1	0	0
DSRegPSO	4	1	1
IHDELS	1	0	0
MLSHADE-SPA	0	4	4
MOS	1	0	0
RO	0	0	0
SACC	0	1	0
SHADEILS	2	8	8
VMODE	0	0	0

Table 5 compares the costs obtained for each algorithm registered in CEC'13 TACO in the functions of the CEC'13 benchmark against DSRegPSO and GPSO. Our proposal obtains the best value in the more complex test with the non-separable function.

Table 5. Mean cost comparison with CEC'13, including the proposed DSRegPSO.

Algorithm	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
AMO	0.00E+00	8.48E+02	0.00E+00	1.57E+08	6.69E+06	1.63E+05	2.07E+04	8.71E+12	4.09E+08	8.99E+05	5.16E+07	3.17E+02	3.14E+06	2.69E+07	2.44E+06
APO	0.00E+00	8.32E+02	0.00E+00	1.62E+08	7.01E+06	1.45E+05	3.31E+02	1.70E+13	3.96E+08	7.07E+05	2.54E+07	1.06E+02	7.88E+05	9.92E+06	2.08E+06
AQO	0.00E+00	8.39E+02	0.00E+00	1.61E+08	6.84E+06	1.79E+05	1.52E+04	7.31E+12	4.08E+08	9.46E+05	4.65E+07	1.91E+02	3.68E+06	2.69E+07	2.40E+06
BICCA	0.00E+00	8.46E−07	7.27E−01	8.85E+08	2.58E+06	1.46E+05	1.82E+05	3.78E+12	2.18E+08	1.24E+06	2.85E+07	1.40E+03	1.09E+07	4.27E+07	3.16E+06
CC-CMA-ES	5.80E−09	1.33E+03	0.00E+00	2.19E+09	7.28E+14	5.87E+05	7.44E+06	3.88E+14	3.71E+08	7.55E+05	1.58E+08	1.27E+03	6.69E+08	7.10E+07	3.03E+07
DECC-G	0.00E+00	1.03E+03	3.00E−10	2.12E+10	5.07E+06	6.08E+04	4.27E+08	3.88E+14	4.17E+08	1.19E+07	1.60E+11	1.07E+03	3.36E+10	6.27E+11	6.01E+07
DEEPSO	1.44E+08	1.49E+04	2.04E+01	4.77E+09	1.45E+07	1.02E+06	1.54E+07	5.42E+12	9.17E+08	9.07E+07	5.60E+08	1.54E+10	8.75E+08	4.33E+08	7.04E+06
DMO	0.00E+00	8.16E+02	0.00E+00	2.20E+08	7.12E+06	1.50E+05	5.26E+04	1.07E+13	5.28E+08	5.70E+05	1.16E+08	2.45E+02	6.55E+06	4.57E+07	3.02E+07
DPO	0.00E+00	1.05E+03	0.00E+00	2.71E+08	6.85E+06	1.38E+05	2.52E+04	2.33E+13	4.02E+08	1.08E+06	9.88E+07	3.45E+02	4.04E+06	2.86E+07	2.80E+06
DQO	0.00E+00	8.41E+02	0.00E+00	1.56E+08	7.06E+06	1.52E+05	2.06E+04	7.52E+12	4.10E+08	8.02E+05	5.43E+07	2.07E+02	3.21E+06	2.43E+07	2.38E+06
DSRegPSO	1.90E−04	6.87E+02	2.00E+01	1.24E+09	3.76E+06	9.96E+05	3.59E+04	1.06E+13	3.05E+08	9.17E+07	4.32E+08	1.56E+03	1.03E+07	1.92E+08	5.79E+05
GPSO	2.91E+09	4.19E+04	2.02E+01	2.75E+10	5.66E+06	1.01E+06	5.73E+08	1.32E+14	6.12E+08	9.07E+07	3.44E+09	1.53E+12	3.03E+09	7.87E+09	2.80E+10
IHDELS	4.34E−28	1.32E+03	2.01E+01	3.04E+08	9.59E+06	1.03E+06	3.46E+04	1.36E+12	6.74E+08	9.16E+07	1.07E+07	3.77E+02	3.80E+06	1.58E+07	2.81E+06
MLSHADE-SPA	1.94E−22	7.89E+01	0.00E+00	6.90E+08	1.80E+06	1.40E+03	5.31E+04	9.77E+12	1.61E+08	6.56E+02	4.04E+07	1.04E+02	7.21E+07	1.52E+07	2.76E+07
MOS	0.00E+00	8.32E+02	0.00E+00	1.74E+08	6.94E+06	1.48E+05	1.62E+04	8.00E+12	3.83E+08	9.02E+05	5.22E+07	2.47E+02	3.40E+06	2.56E+07	2.35E+06
RO	0.00E+00	8.09E+02	0.00E+00	2.25E+08	6.33E+06	1.29E+05	3.46E+04	8.43E+12	3.85E+08	6.14E+05	8.53E+07	4.81E+02	4.61E+06	3.44E+07	1.00E+07
SACC	0.00E+00	5.71E+02	1.21E+00	3.66E+10	6.95E+06	2.07E+05	1.58E+07	9.86E+14	5.77E+08	2.11E+07	5.30E+08	8.74E+02	1.51E+09	7.34E+09	1.88E+06
SHADEILS	2.69E−24	1.00E+03	2.01E+01	1.48E+08	1.39E+06	1.02E+06	7.41E+01	3.17E+11	1.64E+08	9.18E+07	5.11E+05	6.18E+01	1.00E+05	5.76E+06	6.25E+05
VMODE	8.51E−04	5.51E+03	3.41E−04	8.48E+09	7.28E+14	1.99E+05	3.44E+06	3.26E+13	7.51E+08	9.91E+06	1.58E+08	2.34E+03	2.43E+07	9.35E+07	1.11E+07

Additionally, we compare the results in accuracy delivered with the TACO toolkit in Figure 35. Here, we find that Shadeils—an algorithm focusing on LSGO—still has the best results across all the algorithms in the CEC'13 test. However, despite not focusing on LSGO, our proposal improves the results in several functions compared with the other algorithms and obtains the best results compared with algorithms inspired by PSO. Moreover, the cyan region in the plot is the biggest for all the algorithms since our proposal is the best one in the non-separable function.

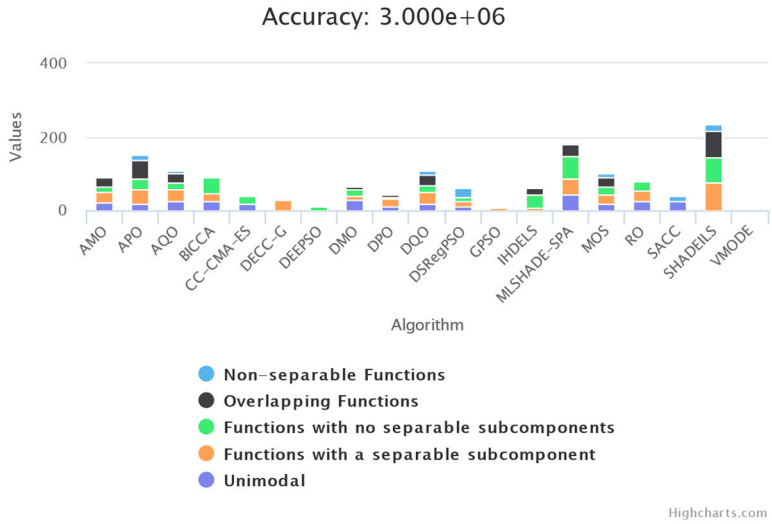

Figure 35. TACO comparison across all algorithms, including DSRegPSO and GPSO in the CEC'13 test.

Figure 36 shows the TACO comparison, just considering the non-separable function, showing that our algorithm is the best at optimizing it.

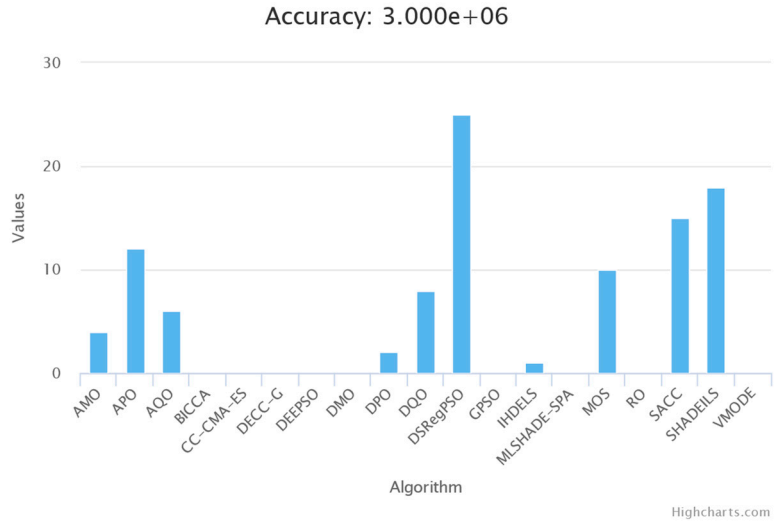

Figure 36. TACO comparison across all algorithms, including DSRegPSO and GPSO in the non-separable CEC'13 function.

The Wilcoxon test is a non-parametric statistical test that compares the median rank of two related samples. It is proper when the normality assumption of the samples cannot be met and is a suitable alternative to the Student's *t*-test [53]. Table 6 shows the results of the statistical Wilcoxon test as defined in the TACO toolkit. It compares the algorithm's *p*-value in each row (first) concerning the algorithm in each column (second).

Table 6. Wilcoxon test comparing algorithms registered in TACO in the CEC'13 test.

Algorithm	AMO	APO	AQO	BICCA	CCCMA-ES	DECC-G	DEEPSO	DMO	DPO	DQO	DSRegPSO	GPSO	IHDELS	MLSHADE-SPA	MOS	RO	SACC	SHADEILS	VMODE	Accum. Error (%)
AMO	1.0E+00	2.1E−01	9.0E−01	8.9E−01	8.4E−03	3.4E−03	8.4E−03	2.0E−02	2.4E−02	4.5E−01	8.3E−02	4.3E−04	1.9E−01	9.3E−01	6.2E−01	6.6E−01	5.7E−03	4.8E−02	6.1E−05	6.0E+01
APO	2.1E−01	1.0E+00	1.7E−01	3.3E−01	2.0E−03	3.4E−03	8.4E−03	7.9E−02	8.4E−03	9.0E−02	3.9E−01	4.3E−04	5.5E−02	9.3E−01	2.1E−01	6.2E−01	1.2E−02	6.4E−02	6.1E−05	6.0E+01
AQO	9.0E−01	1.7E−01	1.0E+00	8.9E−01	8.4E−03	3.4E−03	8.4E−03	1.4E−02	2.8E−02	9.0E−01	8.3E−02	4.3E−04	1.9E−01	9.3E−01	1.0E+00	1.9E−01	5.7E−03	5.5E−02	6.1E−05	6.0E+01
BICCA	8.9E−01	3.3E−01	8.9E−01	1.0E+00	3.4E−03	3.4E−03	6.1E−05	3.6E−01	6.8E−01	8.9E−01	2.2E−02	6.1E−05	4.5E−01	6.0E−01	8.9E−01	9.8E−01	2.6E−03	4.1E−02	1.8E−04	6.0E+01
CCCMA-ES	8.4E−03	2.0E−03	8.4E−03	3.4E−03	1.0E+00	1.3E−01	1.5E−01	4.3E−03	8.4E−03	8.4E−03	3.0E−01	8.3E−02	7.3E−02	6.1E−05	6.7E−03	2.0E−03	1.9E−01	1.0E−02	8.9E−01	6.0E+01
DECC-G	3.4E−03	3.4E−03	3.4E−03	3.4E−03	1.3E−01	1.0E+00	3.3E−01	2.2E−02	6.7E−03	3.4E−03	2.2E−02	6.4E−01	1.4E−01	6.1E−05	3.4E−03	3.4E−03	8.0E−01	8.4E−03	2.3E−01	6.0E+01
DEEPSO	8.4E−03	8.4E−03	8.4E−03	6.1E−05	1.5E−01	3.3E−01	1.0E+00	2.6E−02	8.4E−03	8.4E−03	1.8E−02	1.2E−02	8.5E−04	2.6E−02	8.4E−03	1.8E−02	8.0E−01	8.5E−04	4.2E−01	6.0E+01
DMO	2.0E−02	7.9E−02	1.4E−02	3.6E−01	4.3E−03	2.2E−02	2.6E−02	1.0E+00	4.1E−01	1.7E−02	6.4E−01	4.3E−04	1.0E+00	9.5E−02	1.4E−02	3.8E−02	2.0E−02	4.8E−02	2.0E−03	6.0E+01
DPO	2.4E−02	8.4E−03	2.8E−02	6.8E−01	8.4E−03	6.7E−03	8.4E−03	4.1E−01	1.0E+00	6.0E−02	3.9E−01	4.3E−04	3.9E−01	3.3E−01	1.0E−02	2.6E−01	5.7E−03	2.2E−02	6.1E−05	6.0E+01
DQO	4.5E−01	9.0E−02	9.0E−01	8.9E−01	8.4E−03	3.4E−03	8.4E−03	1.7E−02	6.0E−02	1.0E+00	8.3E−02	4.3E−04	1.9E−01	9.3E−01	8.5E−01	1.9E−01	1.4E−02	5.5E−02	6.1E−05	6.0E+01
DSRegPSO	8.3E−02	3.9E−01	8.3E−02	2.2E−02	3.0E−01	2.2E−02	1.8E−02	6.4E−01	3.9E−01	8.3E−02	1.0E+00	4.3E−04	2.8E−01	3.0E−02	7.3E−02	8.3E−02	4.1E−02	4.8E−02	2.1E−01	6.0E+01
GPSO	4.3E−04	4.3E−04	4.3E−04	6.1E−05	8.3E−02	6.4E−01	1.2E−02	4.3E−04	4.3E−04	4.3E−04	4.3E−04	1.0E+00	1.2E−02	6.1E−05	4.3E−04	3.1E−04	1.1E−01	8.5E−04	2.2E−02	6.0E+01
IHDELS	1.9E−01	5.5E−02	1.9E−01	4.5E−01	7.3E−02	1.4E−01	8.5E−04	1.0E+00	3.9E−01	1.9E−01	2.8E−01	1.2E−02	1.0E+00	8.5E−01	1.9E−01	8.9E−01	3.9E−01	1.5E−03	1.5E−02	6.0E+01
MLSHADE-SPA	9.3E−01	9.3E−01	9.3E−01	6.0E−01	6.1E−05	6.1E−05	2.6E−02	9.5E−02	3.3E−01	9.3E−01	3.0E−02	6.1E−05	8.5E−01	1.0E+00	9.3E−01	8.5E−01	2.6E−03	1.5E−01	1.2E−02	6.0E+01
MOS	6.2E−01	2.1E−01	1.0E+00	8.9E−01	6.7E−03	3.4E−03	8.4E−03	1.4E−02	1.0E−02	8.5E−01	7.3E−02	4.3E−04	1.9E−01	9.3E-01	1.0E+00	5.2E−02	5.7E−03	4.8E−02	6.1E−05	6.0E+01
RO	6.6E−01	6.2E−01	1.9E−01	9.8E−01	2.0E−03	3.4E−03	1.8E−02	3.8E−02	2.6E−01	1.9E−01	8.3E−02	3.1E−04	8.9E−01	8.5E−01	5.2E−02	1.0E+00	5.7E−03	4.8E−02	6.1E−05	6.0E+01
SACC	5.7E−03	1.2E−02	5.7E−03	2.6E−03	1.9E−01	8.0E−01	8.0E−01	2.0E−02	5.7E−03	1.4E−02	4.1E−02	1.1E−01	3.9E−01	2.6E−03	5.7E−03	5.7E−03	1.0E+00	2.2E−02	2.1E−01	6.0E+01
SHADEILS	4.8E−02	6.4E−02	5.5E−02	4.1E−02	1.0E−02	8.4E−03	8.5E−04	4.8E−02	2.2E−02	5.5E−02	4.8E−02	8.5E−04	1.5E−03	1.5E−01	4.8E−02	4.8E−02	2.2E−02	1.0E+00	1.0E−02	6.0E+01
VMODE	6.1E−05	6.1E−05	6.1E−05	1.8E−04	8.9E−01	2.3E−01	4.2E−01	2.0E−03	6.1E−05	6.1E−05	2.1E−01	2.2E−02	1.5E−02	1.2E−02	6.1E−05	6.1E−05	2.1E−01	1.0E−02	1.0E+00	6.0E+01

Blue indicates that the first algorithm is better than the second, and red indicates that the first is worse than the second, with a significance of 5%. Cyan indicates that the first algorithm is better than the second one, and yellow indicates that the first one is worse than the second. For cyan and yellow colors, there is a significance of 10%. Black means that there are no detected significant differences.

The Wilcoxon test results show that despite the limitations of PSO exploring LSGO problems, our proposal still has better results than the four algorithms with 5% significance. Furthermore, we maintain similar performance or inconclusive advantage with eight algorithms.

Table 7 shows the Wilcoxon test considering only the algorithms inspired in PSO (GPSO and DEEPSO). The results support that DSRegPSO is the best PSO-inspired algorithm with results in the CEC'13 test. Again, Blue indicates that the first algorithm is better than the second, and red indicates that the first is worse than the second, with a significance of 5%. Cyan indicates that the first algorithm is better than the second one, and yellow indicates that the first one is worse than the second. For cyan and yellow colors, there is a significance of 10%. Black means that there are no detected significant differences.

Table 7. Wilcoxon test taking into consideration PSO-inspired algorithms.

Algorithm	DEEPSO	DSRegPSO	GPSO	Accum. Error (%)
DEEPSO	1.0E+00	1.8E−02	1.2E−02	9.8E+00
DSRegPSO	1.8E−02	1.0E+00	4.3E−04	9.8E+00
GPSO	1.2E−02	4.3E−04	1.0E+00	9.8E+00

In algorithm design, it is essential to assess both robustness and sensitivity, especially in situations where disturbances or noise are expected. Robustness refers to the ability of an algorithm to maintain its performance despite perturbations, while sensitivity measures how much these perturbations can affect the outputs of the algorithm [56,57].

A robust algorithm can handle different types of disturbances, such as input errors or process variations, and still generate consistent and reliable results. This robustness is demonstrated by the algorithm's ability to continuously improve and converge toward a solution even when noise is present in particle positions. On the other hand, algorithm sensitivity explains how the algorithm reacts to minor variations or disturbances. A highly sensitive algorithm will produce significant variations in outcomes even with minor disturbances [57].

In our study, we evaluated the robustness and sensitivity of DSRegPSO by conducting 25 runs of CEC'13 under identical circumstances without controlling the seeds. This allowed us to generate different initial positions of particles. The results in Table 3 and the convergence diagrams demonstrate that DSRegPSO performs even better than the original PSO in terms of stability.

Additionally, as a means of testing the robustness of the DSRegPSO algorithm in the presence of noise, we carried out two experiments using the CEC'13 functions. The first experiment involved introducing random noise of varying magnitudes (1%, 5%, 10% 15%, and 20%) to the particle positions before evaluating the functions. The purpose of this experiment was to evaluate how the algorithm responds to perturbations in the search space for high-dimensional functions. Specifically, it aimed to determine whether the algorithm could still improve or converge to an optimal or near-optimal solution even when particles in a high-dimensional space are slightly displaced from their original positions due to noise.

In the second test, we introduce random noise of magnitudes (1%, 5%, 10%, 15%, and 20%) into the result of the cost function evaluation. This test assesses the algorithm's ability to handle errors or variations in the evaluation of the objective function in high-dimensional spaces. Specifically, we investigate how the algorithm responds to the perceived "quality" of a solution (measured by its objective function) in the presence of noise. The goal is to

understand how noise affects the algorithm's ability to still improving or converge to an optimal or near-optimal solution.

Figure 37 shows a comparison between convergence diagrams with noise in particle positions and cost of Function 1 of CEC'13. Despite noise in particle positions, the algorithm continues to improve in the initial evaluations but rapidly converges to similar cost values. This could be due to the sensitivity of the cost function to small changes in particle positions. However, the algorithm still improves in tests with noise added to the cost, demonstrating the robustness of DSRegPSO.

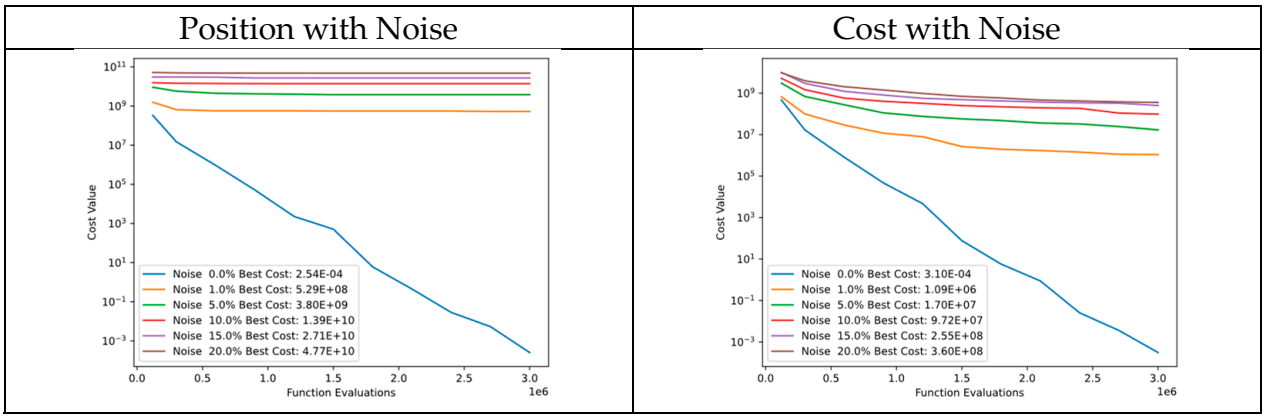

Figure 37. Convergence diagrams of Function 1 of CEC'13 adding noise to position and cost.

In Figure 38, there is a comparison of convergence diagrams between noise in particle positions and cost of Function 2 of CEC'13. Despite the noise in particle positions, the algorithm continues to perform well in the initial evaluations and subsequently converges to similar cost values. This could be because the cost function is very sensitive to even small changes in particle positions. However, the algorithm still performs well in tests where noise is added to the cost, which demonstrates the robustness of DSRegPSO.

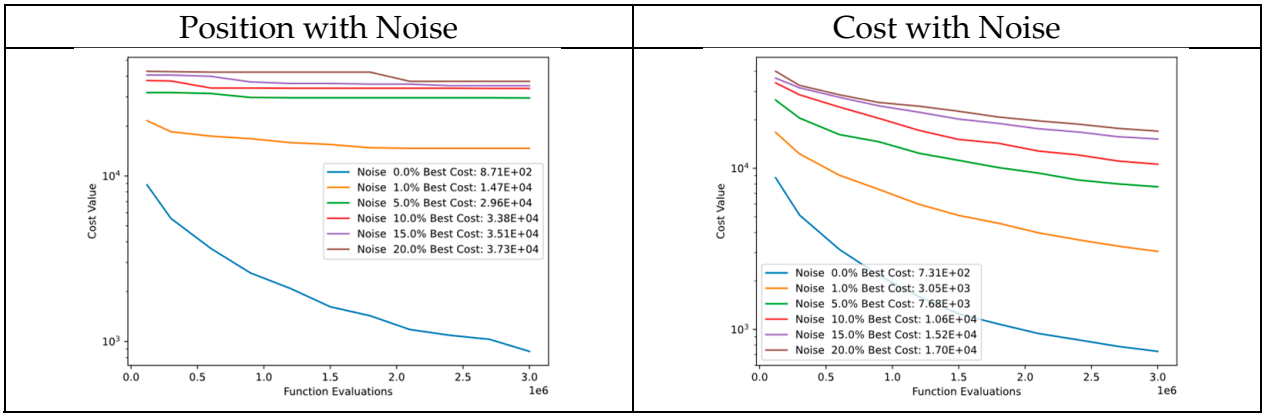

Figure 38. Convergence diagrams of Function 2 of CEC'13 adding noise to position and cost.

In the comparison of convergence diagrams between noise in particle positions and the cost of Function 3 of CEC'13 depicted in Figure 39, adding noise to particle positions or the cost value resulted in convergence to similar cost values, with minimal improvements. This could be due to the fact that the cost function is highly sensitive to changes in position, and the cost value variation is also too small compared with the noise, as evidenced by both convergence diagrams. As a result, the algorithm showed a lack of robustness in noise tests for Function 3.

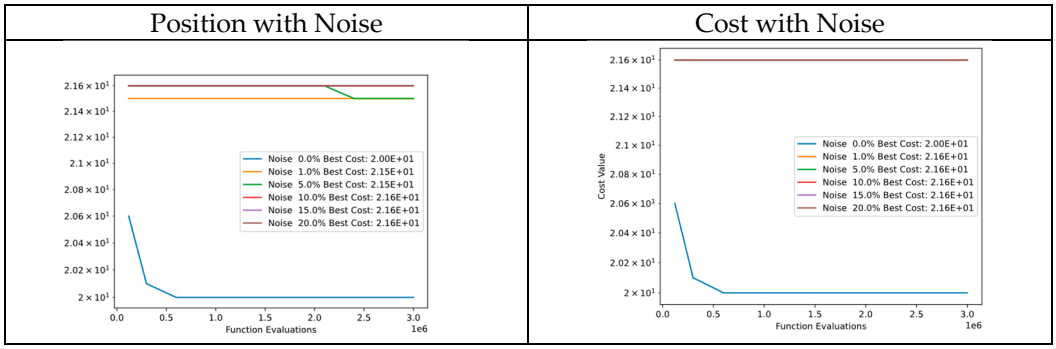

Figure 39. Convergence diagrams of Function 3 of CEC'13 adding noise to position and cost.

Figure 40 presents a comparison of convergence diagrams between particle position noise and cost of Function 4 of CEC'13. Despite noise in particle positions, the DSRegPSO algorithm exhibits good performance in the initial evaluations and subsequently converges to similar cost values. This is due to the high sensitivity of the cost function to even minor alterations in particle positions. However, tests where noise is added to the cost indicate that DSRegPSO is robust and continues to perform well.

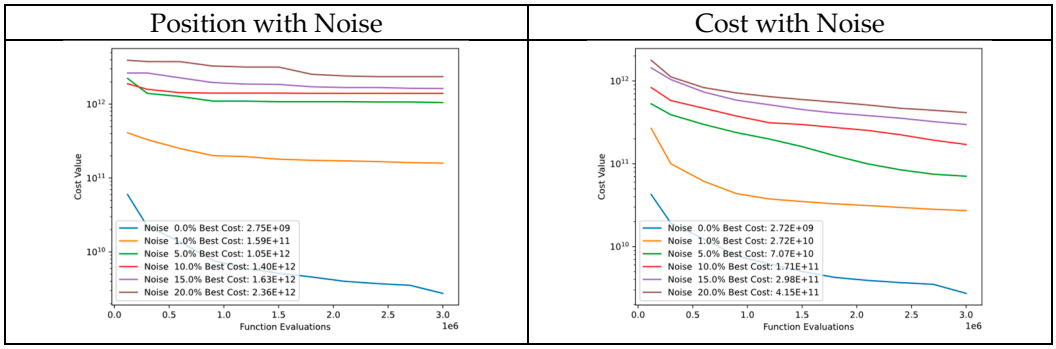

Figure 40. Convergence diagrams of Function 4 of CEC'13 adding noise to position and cost.

Figure 41 showcases a comparison of convergence diagrams adding noise to particle position and cost of Function 5 of CEC'13. Despite the presence of noise in particle positions, the DSRegPSO algorithm exhibits commendable performance, converging to similar cost values. This can be attributed to the cost function's high sensitivity to even minute changes in particle positions. However, tests that introduced noise to the cost indicate that DSRegPSO is robust and continues to deliver good performance.

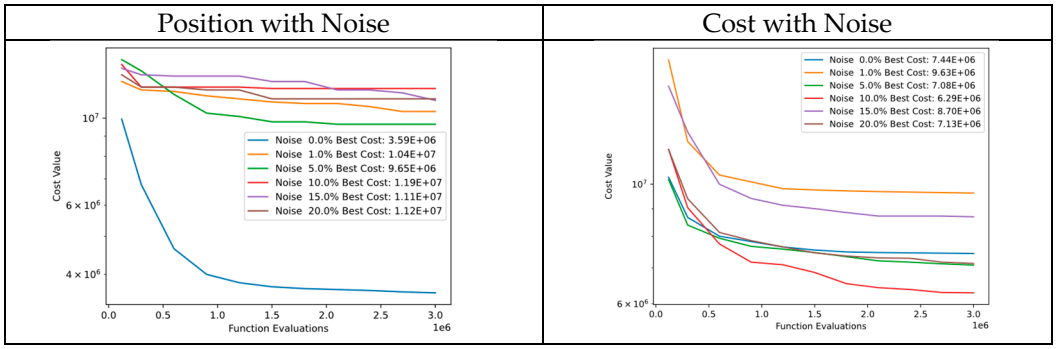

Figure 41. Convergence diagrams of Function 5 of CEC'13 adding noise to position and cost.

When comparing convergence diagrams between noise in particle positions and cost of Function 6 of CEC'13 (Figure 42), it was found that adding noise to either the

particle positions or cost value led to convergence to similar cost values, with only minimal improvements. This could be because the cost function is highly sensitive to changes in position and the cost value variation is too small compared with the noise. Consequently, the algorithm demonstrated a lack of robustness in noise tests for Function 6.

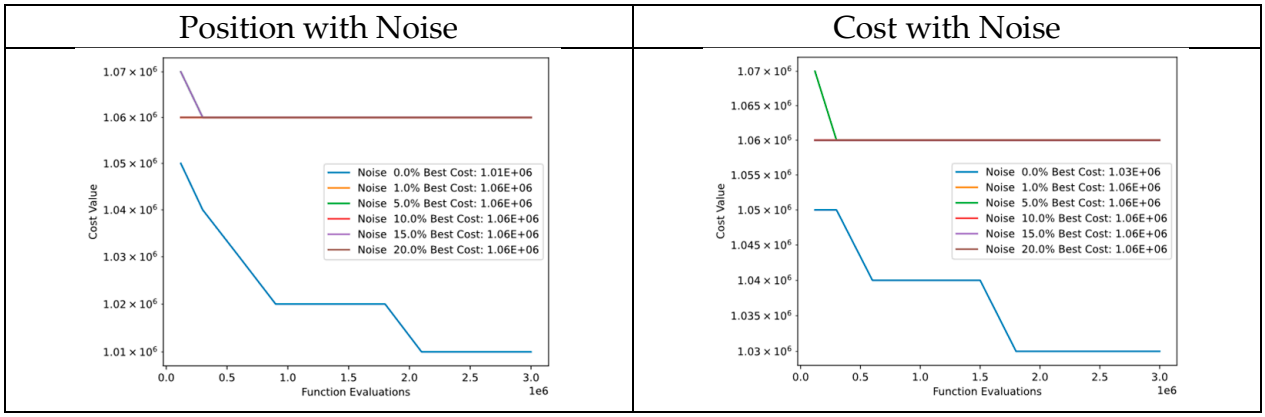

Figure 42. Convergence diagrams of Function 6 of CEC'13 adding noise to position and cost.

In Figure 43, there is a comparison of convergence diagrams between noise in particle positions and cost of Function 7 of CEC'13. Despite the noise in particle positions, the algorithm continues to perform well in the initial evaluations and subsequently reaches similar cost values. This could be because the cost function is very sensitive to even small changes in particle positions. However, the algorithm still performs well in tests where noise is added to the cost, which demonstrates the robustness of DSRegPSO.

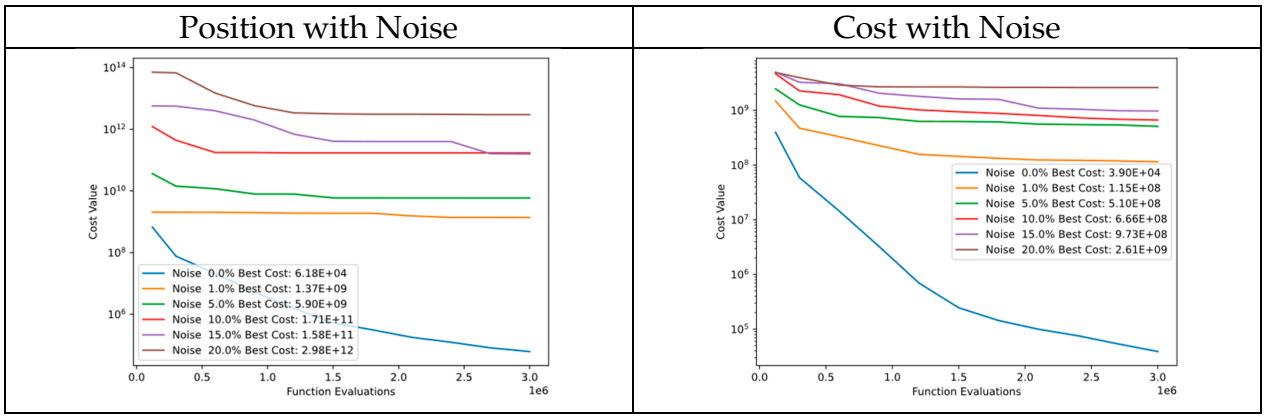

Figure 43. Convergence diagrams of Function 7 of CEC'13 adding noise to position and cost.

Figure 44 presents a comparison of convergence diagrams between particle position noise and cost of Function 8 of CEC'13. Despite noise in particle positions, the DSRegPSO algorithm exhibits good performance in the initial evaluations and subsequently converges to similar cost values. This is due to the high sensitivity of the cost function to even minor alterations in particle positions. However, tests where noise is added to the cost indicate that DSRegPSO is robust and continues to perform well.

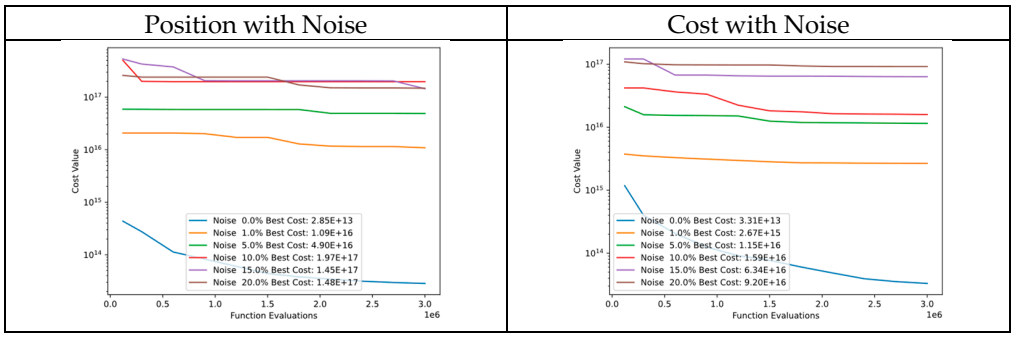

Figure 44. Convergence diagrams of Function 8 of CEC'13 adding noise to position and cost.

Figure 45 showcases a comparison of convergence diagrams adding noise to particle position and cost of Function 9 of CEC'13. Despite the presence of noise in particle positions, the DSRegPSO algorithm exhibits commendable performance, converging to similar cost values. This can be attributed to the cost function's high sensitivity to even minute changes in particle positions. However, tests that introduced noise to the cost indicate that DSRegPSO is robust and continues to deliver good performance.

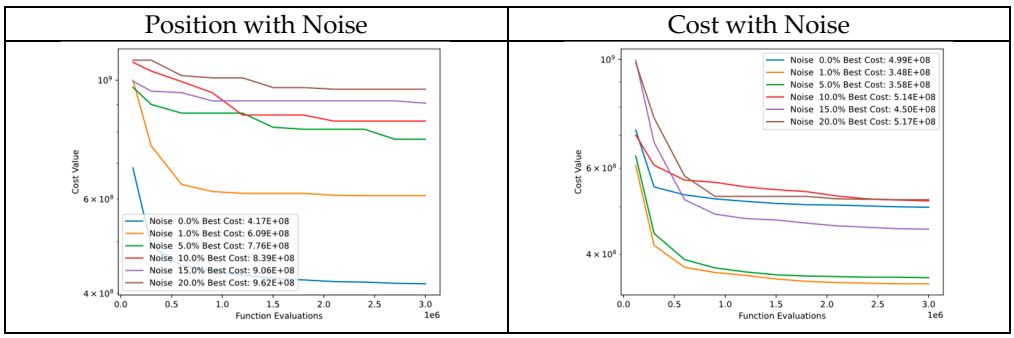

Figure 45. Convergence diagrams of Function 9 of CEC'13 adding noise to position and cost.

Figure 46 presents a comparison of convergence diagrams between particle position noise and cost of Function 10 of CEC'13. Despite noise in particle positions, the DSRegPSO algorithm exhibits good performance in the initial evaluations and subsequently converges to similar cost values. This is due to the high sensitivity of the cost function to even minor alterations in particle positions. However, tests where noise is added to the cost indicate that DSRegPSO is robust and continues to perform well.

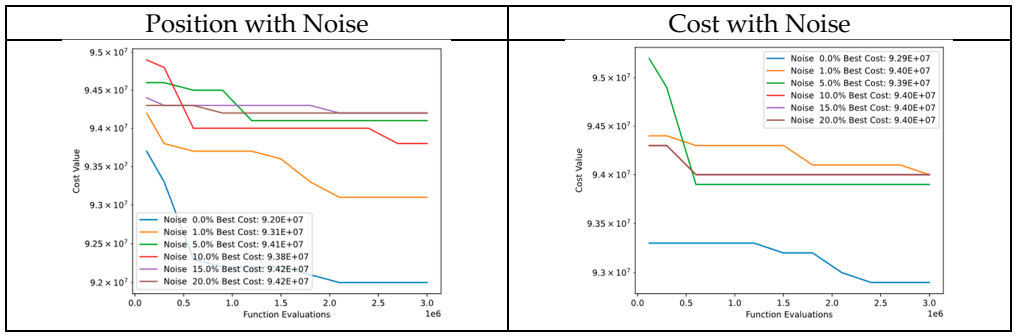

Figure 46. Convergence diagrams of Function 10 of CEC'13 adding noise to position and cost.

In Figure 47, there is a comparison of convergence diagrams between noise in particle positions and cost of Function 11 of CEC'13. Despite the noise in particle positions, the algorithm continues to perform well in the initial evaluations and continue improving in its cost values without reaching the response without noise. This could be because the

cost function is very sensitive to even small changes in particle positions. However, the algorithm still performs well in tests where noise is added to the cost, which demonstrates the robustness of DSRegPSO.

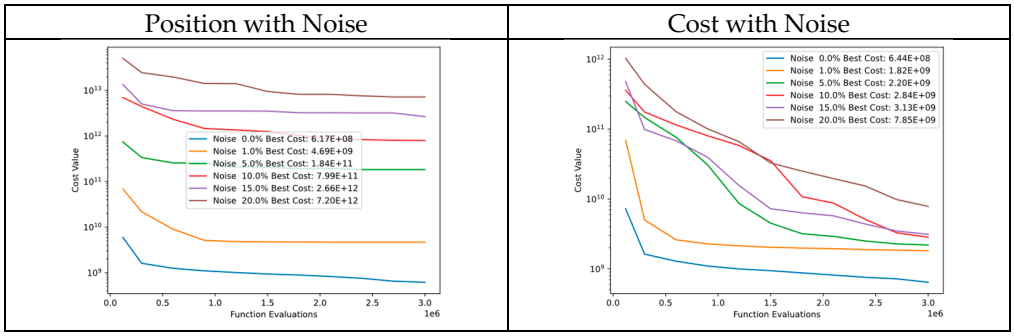

Figure 47. Convergence diagrams of Function 11 of CEC'13 adding noise to position and cost.

Figure 48 showcases a comparison of convergence diagrams adding noise to particle position and cost of Function 12 of CEC'13. Despite the presence of noise in particle positions, the DSRegPSO algorithm exhibits commendable performance, converging to similar cost values. This can be attributed to the cost function's high sensitivity to even minute changes in particle positions. However, tests that introduced noise to the cost indicate that DSRegPSO is robust and continues to deliver good performance.

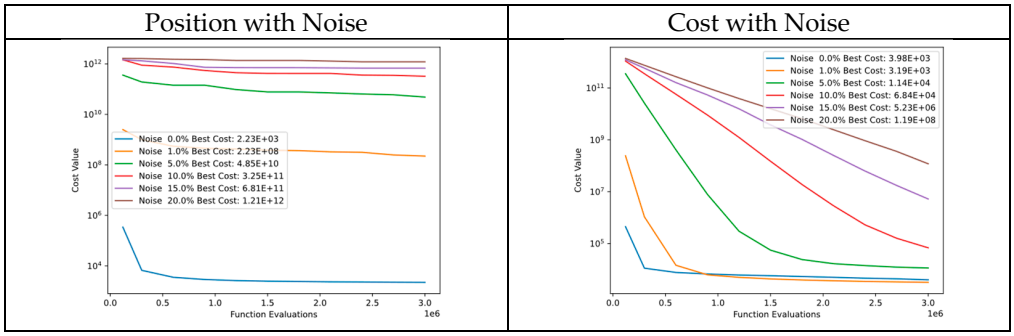

Figure 48. Convergence diagrams of Function 12 of CEC'13 adding noise to position and cost.

In Figure 49, there is a comparison of convergence diagrams between noise in particle positions and cost of Function 13 of CEC'13. Despite the noise in particle positions, the algorithm continues to perform well in the initial evaluations and subsequently reaches similar cost values. This could be because the cost function is very sensitive to even small changes in particle positions. However, the algorithm still performs well in tests where noise is added to the cost, which demonstrates the robustness of DSRegPSO.

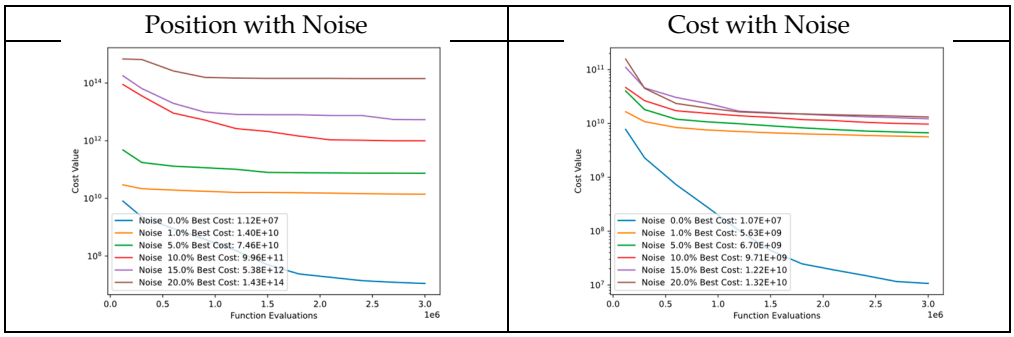

Figure 49. Convergence diagrams of Function 13 of CEC'13 adding noise to position and cost.

Figure 50 shows convergence diagrams for noise in particle positions and the cost of Function 14 of CEC'13. Despite the noise, the algorithm performs well in initial evaluations and continues to improve cost values. The cost function is sensitive to small changes in particle positions, which may explain this. However, DSRegPSO shows better robustness in tests where noise is added to the cost instead of the position of particles.

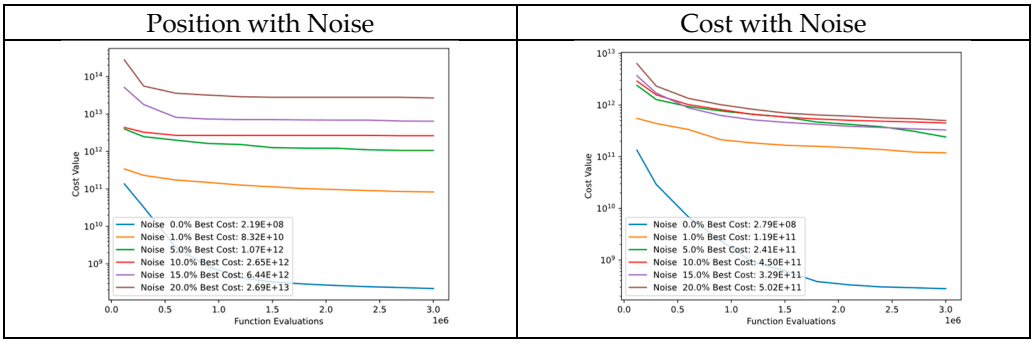

Figure 50. Convergence diagrams of Function 14 of CEC'13 adding noise to position and cost.

In Figure 51, there is a comparison of convergence diagrams between noise in particle positions and cost of Function 15 of CEC'13. Despite the noise in particle positions, the algorithm continues to perform well in the initial evaluations and subsequently reaches similar cost values. This could be because the cost function is very sensitive to even small changes in particle positions. However, the algorithm still performs well in tests where noise is added to the cost, which demonstrates the robustness of DSRegPSO.

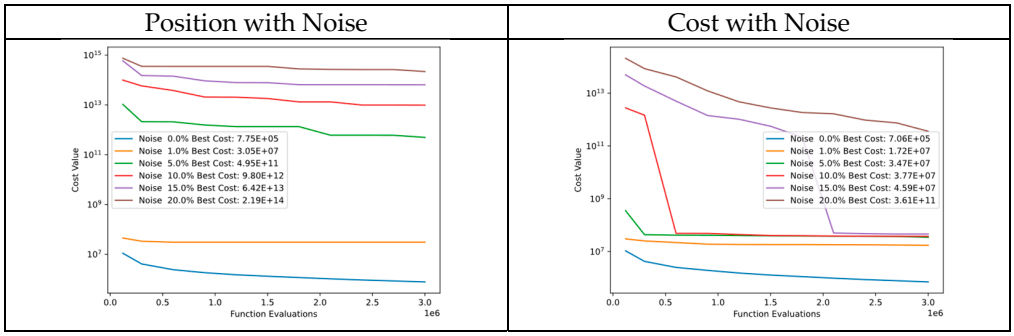

Figure 51. Convergence diagrams of Function 15 of CEC'13 adding noise to position and cost.

4. Conclusions

In this work, we present a novel adaptation of Particle Swarm Optimization (PSO) known as Dynamical Sphere Regrouping PSO (DSRegPSO). The proposed variant addresses the issue of stagnation that commonly occurs in global optimization problems, particularly those that are large-scale. To avoid stagnation, we introduced two mechanisms.

The first mechanism involves the use of a dynamical sphere to regulate exploration and exploitation during a run, thereby preventing the particles from converging. The second mechanism is based on the conservation of physics momentum, which returns particles that travel outside the search space in the opposite direction, enabling maximum exploration of the search space.

We improved the original regrouping PSO algorithm in three main ways. Firstly, we introduced dynamical sphere regrouping, which allows the swarm to be continually reinvigorated without waiting for stagnation to be detected. This mechanism also helps regulate exploration and exploitation during a run. Secondly, we modified the inertial effects, varying them across iterations and increasing them when required for exploration. Lastly, we used momentum conservation to maintain diversity in the swarm by keeping particles within the search space when the speed equation takes them out.

Despite the existence of several variants and alternatives in the PSO algorithm, not all of them have been registered in the Toolkit of Automatic Comparison of Optimizers in CEC'13, which is a standard for LSGO evaluation. Moreover, some proposals only work with their own proposed tests or on specific problems. However, our DSRegPSO algorithm underwent testing against the CEC'13 functions for evaluating new machine learning algorithms in LSGO and achieved superior results compared with other algorithms and PSO-inspired algorithms, such as DEEPSO and GPSO.

In order to sustain our conclusion, we conducted a Wilcoxon test to compare all algorithms that use CEC'13 as support for their results. The Wilcoxon test results demonstrated that DSRegPSO is superior to DECC-G, DEEPSO, GPSO, and SACC and has similar results to APO, CCCMA-ES, DMO, DPO, IHDELS, and VMODE with a statistical significance of 5%. Additionally, our algorithm produced the best results for optimizing the non-separable function reaching a cost value of 5.79E+05, surpassing all other CEC'13 TACO registered algorithms. Thus, we recommend utilizing the proposed algorithm in applications of non-separable problems to explore the capabilities of DSRegPSO in real-world scenarios. Again, we conducted a Wilcoxon test, but based on PSO-inspired algorithms, and found that DSRegPSO outperforms GPSO and DEEPSO with 5% significance level.

In the Wilcoxon test, we also found that Shadeils, an algorithm focused on LSGO, delivered the best results in separable functions across all algorithms in the CEC'13 test. We identified that DSRegPSO did not produce the best results in separable functions because it did not prioritize detecting possible simplifications of the search space by separately optimizing dimensions, although this was not the focus of DSRegPSO in our work.

Despite not focusing on LSGO, our proposal achieved superior results in several functions compared with other algorithms and outperformed PSO-inspired algorithms.

Additionally, we used PCA to transform the particle positions from 1000 dimensions to 3 and plot 3D points with their positions across iterations. This confirms that the algorithm continuously avoids stagnation and keeps looking for better positions to improve the global best without converging in position. Although the algorithm may converge early in the global best cost of Function 3 of CEC'13, it never converges in position.

Algorithm design requires assessing robustness and sensitivity to expected disturbances or noise. Robustness ensures consistent performance despite perturbations, while sensitivity measures the impact of perturbations on an algorithm's outputs.

We conducted 25 runs of CEC'13 without controlling the seeds to vary starting conditions and evaluate DSRegPSO's robustness. The results show that DSRegPSO performs better than the original PSO in terms of stability, as demonstrated by the convergence diagrams in Table 3 that compare mean, best, worst, and standard deviation.

Furthermore, we conducted two experiments using CEC'13 functions to test the robustness and sensitivity of the DSRegPSO algorithm against noise and registered the convergence diagrams. The first experiment involved adding random noise (1%, 5%, 10%, 15%, and 20%) to particle positions to evaluate the algorithm's ability to converge in a high-dimensional space. In the second experiment, we introduced random noise of varying levels (1%, 5%, 10%, 15%, and 20%) to assess the algorithm's performance in handling errors or variations in the high-dimensional objective function evaluation. Based on our tests, we found that the DSRegPSO is a robust algorithm that can handle noise effectively. The algorithm continued to improve even in the presence of noise in 13 out of 15 functions of CEC'13. However, in all the functions, it did react to noise by modifying the reached global best. We believe that this was due to the high sensitivity of CEC'13 functions.

Future Work

The DSRegPSO algorithm has demonstrated superior performance in the non-separable function of the CEC'13 when compared with non-PSO-inspired algorithms. To further improve its utility in future versions, we propose the implementation of mechanisms for detecting separability, allowing for a simplified search space with separable functions, as seen in other LSGO algorithms.

Additionally, a thorough analysis of the parameters in the proposed algorithm is suggested to determine the optimal values for optimizing different situations. We particularly suggest focusing on the parameters related to the maximum diameter of the hyper-sphere and the expansion speed, as an optimal choice for these parameters could lead to less exploration and, consequently, fewer iterations.

Author Contributions: M.M.R., conceptualization and methodology; M.M.R. and C.G.-M., validation; formal analysis and investigation; all authors, experimentation and results; M.M.R. and C.G.-M., writing—original draft; D.L.-B., T.S.-A., M.M.R. and C.G.-M., writing—review and editing, visualization, and supervision; M.M.R. project administration. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data used in this research as the support for the results in this work is publicly available and can be accessed at the following URL: https://github.com/mmrMontes/DSRegPSO.

Acknowledgments: We acknowledge the support, time, and space for experimentation in Unidad Académica de Ciencia y Tecnologia de la Luz y la Materia, Universidad Autonoma de Zacatecas, Campus Siglo XXI, Zacatecas 98160. We also thank CONAHCyT for its support in the scholarship Estancias Posdoctorales México 2022(1) (CVU: 471898).

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

References

1. Sedighizadeh, D.; Masehian, E.; Sedighizadeh, M.; Akbaripour, H. GEPSO: A New Generalized Particle Swarm Optimization Algorithm. *Math. Comput. Simul.* **2021**, *179*, 194–212. [CrossRef]
2. Senapati, M.K.; Pradhan, C.; Calay, R.K. A Computational Intelligence Based Maximum Power Point Tracking for Photovoltaic Power Generation System with Small-Signal Analysis. *Optim. Control Appl. Methods* **2023**, *44*, 617–636. [CrossRef]
3. Vahedipour-Dahraie, M.; Rashidizadeh-Kermani, H.; Anvari-Moghaddam, A. Risk-Based Stochastic Scheduling of Resilient Microgrids Considering Demand Response Programs. *IEEE Syst. J.* **2021**, *15*, 971–980. [CrossRef]
4. van den Bergh, F.; Engelbrecht, A.P. A Cooperative Approach to Participle Swam Optimization. *IEEE Trans. Evol. Comput.* **2004**, *8*, 225–239. [CrossRef]
5. Maučec, M.S.; Brest, J. A Review of the Recent Use of Differential Evolution for Large-Scale Global Optimization: An Analysis of Selected Algorithms on the CEC 2013 LSGO Benchmark Suite. *Swarm Evol. Comput.* **2018**, *50*, 100428. [CrossRef]
6. Sun, G.; Han, R.; Deng, L.; Li, C.; Yang, G. Hierarchical Structure-Based Joint Operations Algorithm for Global Optimization. *Swarm Evol. Comput.* **2023**, *79*, 101311. [CrossRef]
7. Glorieux, E.; Svensson, B.; Danielsson, F.; Lennartson, B. Constructive Cooperative Coevolution for Large-Scale Global Optimization. *J. Heuristics* **2017**, *23*, 449–469. [CrossRef]
8. Jiang, R.; Shankaran, R.; Wang, S.; Chao, T. A Proportional, Integral and Derivative Differential Evolution Algorithm for Global Optimization. *Expert Syst. Appl.* **2022**, *206*, 117669. [CrossRef]
9. Marcelino, C.; Almeida, P.; Pedreira, C.; Caroalha, L.; Wanner, E. Applying C-DEEPSO to Solve Large Scale Global Optimization Problems. In Proceedings of the 2018 IEEE Congress on Evolutionary Computation, CEC, Rio de Janeiro, Brazil, 8–13 July 2018. [CrossRef]
10. Yang, S.; Jiang, J.; Yan, G. A Dolphin Partner Optimization. In Proceedings of the 2009 WRI Global Congress on Intelligent Systems, GCIS 2009, Xiamen, China, 19–21 May 2009; Volume 1, pp. 124–128. [CrossRef]
11. Koçer, H.G.; Uymaz, S.A. A Novel Local Search Method for LSGO with Golden Ratio and Dynamic Search Step. *Soft Comput.* **2021**, *25*, 2115–2130. [CrossRef]
12. Molina, D.; Latorre, A.; Herrera, F. SHADE with Iterative Local Search for Large-Scale Global Optimization. In Proceedings of the 2018 IEEE Congress on Evolutionary Computation, CEC 2018—Proceedings, Rio de Janeiro, Brazil, 8–13 July 2018. [CrossRef]
13. López, E.D.; Puris, A.; Bello, R.R. Vmode: A Hybrid Metaheuristic for the Solution of Large Scale Optimization Problems. *Investig. Oper.* **2015**, *36*, 232–239.
14. LaTorre, A.; Pena, J.M. A Comparison of Three Large-Scale Global Optimizers on the CEC 2017 Single Objective Real Parameter Numerical Optimization Benchmark. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation, CEC 2017—Proceedings, Donostia, Spain, 5–8 June 2017; pp. 1063–1070. [CrossRef]
15. 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]

16. Pluhacek, M.; Senkerik, R.; Viktorin, A.; Kadavy, T. PSO with Attractive Search Space Border Points. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Cham, Switzerland, 2017; Volume 10246, pp. 665–675. [CrossRef]
17. Xu, Y.; Pi, D. A Reinforcement Learning-Based Communication Topology in Particle Swarm Optimization. *Neural Comput. Appl.* **2020**, *32*, 10007–10032. [CrossRef]
18. Pluhacek, M.; Senkerik, R.; Viktorin, A.; Zelinka, I. Multi-Chaotic Approach for Particle Acceleration in PSO. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Cham, Switzerland, 2016; Volume 9668, pp. 75–86. [CrossRef]
19. Nobile, M.S.; Cazzaniga, P.; Besozzi, D.; Colombo, R.; Mauri, G.; Pasi, G. Fuzzy Self-Tuning PSO: A Settings-Free Algorithm for Global Optimization. *Swarm Evol. Comput.* **2018**, *39*, 70–85. [CrossRef]
20. Pradhan, C.; Senapati, M.K.; Malla, S.G.; Nayak, P.K.; Gjengedal, T. Coordinated Power Management and Control of Standalone PV-Hybrid System with Modified IWO-Based MPPT. *IEEE Syst. J.* **2020**, *15*, 3585–3596. [CrossRef]
21. Lin, A.; Sun, W.; Yu, H.; Wu, G.; Tang, H. Global Genetic Learning Particle Swarm Optimization with Diversity Enhancement by Ring Topology. *Swarm Evol. Comput.* **2018**, *44*, 571–583. [CrossRef]
22. Xia, X.; Gui, L.; Zhan, Z.-H. A Multi-Swarm Particle Swarm Optimization Algorithm Based on Dynamical Topology and Purposeful Detecting. *Appl. Soft Comput.* **2018**, *67*, 126–140. [CrossRef]
23. Erskine, A.; Joyce, T.; Herrmann, J.M. Stochastic Stability of Particle Swarm Optimisation. *Swarm Intell.* **2017**, *11*, 295–315. [CrossRef]
24. Al-Bahrani, L.T.; Patra, J.C. Multi-Gradient PSO Algorithm for Optimization of Multimodal, Discontinuous and Non-Convex Fuel Cost Function of Thermal Generating Units under Various Power Constraints in Smart Power Grid. *Energy* **2018**, *147*, 1070–1091. [CrossRef]
25. Huang, C.; Zhou, X.; Ran, X.; Liu, Y.; Deng, W.; Deng, W. Co-Evolutionary Competitive Swarm Optimizer with Three-Phase for Large-Scale Complex Optimization Problem. *Inf. Sci.* **2023**, *619*, 2–18. [CrossRef]
26. Liu, H.; Wang, Y.; Tu, L.; Ding, G.; Hu, Y. A Modified Particle Swarm Optimization for Large-Scale Numerical Optimizations and Engineering Design Problems. *J. Intell. Manuf.* **2018**, *30*, 2407–2433. [CrossRef]
27. Wang, H.; Liang, M.; Sun, C.; Zhang, G.; Xie, L. Multiple-Strategy Learning Particle Swarm Optimization for Large-Scale Optimization Problems. *Complex. Intell. Syst.* **2021**, *7*, 1–16. [CrossRef]
28. Al-Bahrani, L.T.; Patra, J.C. A Novel Orthogonal PSO Algorithm Based on Orthogonal Diagonalization. *Swarm Evol. Comput.* **2018**, *40*, 1–23. [CrossRef]
29. 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; pp. 69–73.
30. Evers, G.I.; Ghalia, M. Ben Regrouping Particle Swarm Optimization: A New Global Optimization Algorithm with Improved Performance Consistency across Benchmarks. In Proceedings of the 2009 IEEE International Conference on Systems, Man and Cybernetics, San Antonio, TX, USA, 11–14 October 2009; pp. 3901–3908. [CrossRef]
31. Li, F.; Yue, Q.; Liu, Y.; Ouyang, H.; Gu, F. A Fast Density Peak Clustering Based Particle Swarm Optimizer for Dynamic Optimization. *Expert. Syst. Appl.* **2024**, *236*, 121254. [CrossRef]
32. Akan, Y.Y.; Herrmann, J.M. Stability, Entropy and Performance in PSO. In Proceedings of the GECCO 2023 Companion—Proceedings of the 2023 Genetic and Evolutionary Computation Conference Companion, Lisbon, Portugal, 15–19 July 2023; pp. 811–814. [CrossRef]
33. Sun, L.; Yang, Y.; Wei, W. A Three-Stage Gene Selection Algorithm Based on Intrinsic Dimension and the Concise Particle Swarm Optimization. *SSRN* 2023. [CrossRef]
34. Tsujimoto, T.; Shindo, T.; Jin'no, K. The Neighborhood of Canonical Deterministic PSO. In Proceedings of the Evolutionary Computation (CEC), New Orleans, LA, USA, 5–8 June 2011; pp. 1811–1817.
35. Kennedy, J. Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Washington, DC, USA, 6–9 July 1999; Volume 3, pp. 1931–1938.
36. Yang, X.; Li, H.; Huang, Y. An Adaptive Dynamic Multi-Swarm Particle Swarm Optimization with Stagnation Detection and Spatial Exclusion for Solving Continuous Optimization Problems. *Eng. Appl. Artif. Intell.* **2023**, *123*, 106215. [CrossRef]
37. Jiang, J.J.; Wei, W.X.; Shao, W.L.; Liang, Y.F.; Qu, Y.Y. Research on Large-Scale Bi-Level Particle Swarm Optimization Algorithm. *IEEE Access* **2021**, *9*, 56364–56375. [CrossRef]
38. Zhao, Q.; Li, C. Two-Stage Multi-Swarm Particle Swarm Optimizer for Unconstrained and Constrained Global Optimization. *IEEE Access* **2020**, *8*, 124905–124927. [CrossRef]
39. Balavalikar, S.; Nayak, P.; Shenoy, N.; Nayak, K. Particle Swarm Optimization Based Artificial Neural Network Model for Forecasting Groundwater Level in Udupi District. *Proc. AIP Conf. Proc.* **2018**, *1952*, 20021.
40. Yang, X.; Li, H. Evolutionary-State-Driven Multi-Swarm Cooperation Particle Swarm Optimization for Complex Optimization Problem. *Inf. Sci.* **2023**, *646*, 119302. [CrossRef]
41. Nagra, A.A.; Han, F.; Ling, Q.H.; Mehta, S. An Improved Hybrid Method Combining Gravitational Search Algorithm with Dynamic Multi Swarm Particle Swarm Optimization. *IEEE Access* **2019**, *7*, 50388–50399. [CrossRef]

42. Nagra, A.A.; Han, F.; Ling, Q.H. An Improved Hybrid Self-Inertia Weight Adaptive Particle Swarm Optimization Algorithm with Local Search. *Eng. Optim.* **2018**, *51*, 1115–1132. [CrossRef]
43. Vakhnin, A.V.; Sopov, E.A.; Panfilov, I.A.; Polyakova, A.S.; Kustov, D.V. A Problem Decomposition Approach for Large-Scale Global Optimization Problems. *IOP Conf. Ser. Mater. Sci. Eng.* **2019**, *537*, 052031. [CrossRef]
44. Liao, T.; Stutzle, T. Benchmark Results for a Simple Hybrid Algorithm on the CEC 2013 Benchmark Set for Real-Parameter Optimization. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation, CEC 2013, Cancun, Mexico, 20–23 June 2013; pp. 1938–1944. [CrossRef]
45. Qu, L.; Zheng, R.; Shi, Y. BSO-CMA-ES: Brain Storm Optimization Based Covariance Matrix Adaptation Evolution Strategy for Multimodal Optimization. In *Communications in Computer and Information Science*; Springer: Cham, Switzerland, 2021; Volume 1454, pp. 167–174. [CrossRef]
46. Lan, R.; Zhang, L.; Tang, Z.; Liu, Z.; Luo, X. A Hierarchical Sorting Swarm Optimizer for Large-Scale Optimization. *IEEE Access* **2019**, *7*, 40625–40635. [CrossRef]
47. Diep, Q.B.; Zelinka, I.; Das, S. Self-Organizing Migrating Algorithm Pareto. *Mendel* **2019**, *25*, 111–120. [CrossRef]
48. Zhao, D.; Yi, J.; Liu, D. Particle Swarn Optimized Adaptive Dynamic Programming. In Proceedings of the Approximate Dynamic Programming and Reinforcement Learning, ADPRL 2007, Honolulu, HI, USA, 1–5 April 2007; pp. 32–37.
49. Strang, G. Calculus. In *Open Textbook Library*; Wellesley-Cambridge Press: Wellesley, MA, USA, 1991; ISBN 9780961408824.
50. Parsopoulos, K.E.; Vrahatis, M.N. *Particle Swarm Optimization and Intelligence*; IGI Global: Hershey, PA, USA, 2010. [CrossRef]
51. Olorunda, O.; Engelbrecht, A.P. Measuring Exploration/Exploitation in Particle Swarms Using Swarm Diversity. In Proceedings of the Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence), Hong Kong, China, 1–6 June 2008; pp. 1128–1134.
52. Li, X.; Tang, K.; Omidvar, M.N.; Yang, Z.; Qin, K.; China, H. Benchmark Functions for the CEC 2013 Special Session and Competition on Large-Scale Global Optimization. *Gene* **2013**, *7*, 8.
53. Molina, D.; Latorre, A. Toolkit for the Automatic Comparison of Optimizers: Comparing Large-Scale Global Optimizers Made Easy. In Proceedings of the 2018 IEEE Congress on Evolutionary Computation, CEC 2018—Proceedings, Rio de Janeiro, Brazil, 8–13 July 2018. [CrossRef]
54. Clerc, M. *Particle Swarm Optimization*; ISTE: London, UK, 2006; ISBN 9780470612163.
55. Kennedy, J.; Eberhart, R. Particle Swarm Optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November 1995–1 December 1995; Volume 4, pp. 1942–1948. [CrossRef]
56. Ferreira, S.L.C.; Caires, A.O.; Borges, T.d.S.; Lima, A.M.D.S.; Silva, L.O.B.; dos Santos, W.N.L. Robustness Evaluation in Analytical Methods Optimized Using Experimental Designs. *Microchem. J.* **2017**, *131*, 163–169. [CrossRef]
57. Bonnini, S.; Chesneau, C.; Ghosh, I.; Fleming, K. On the Robustness and Sensitivity of Several Nonparametric Estimators via the Influence Curve Measure: A Brief Study. *Mathematics* **2022**, *10*, 3100. [CrossRef]