Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Deep Learning Surrogate-Assisted Particle Swarm Optimization in Partitioned Search Spaces with Adaptive Neighborhood Topology for Continuous Optimization (PSA-PSO)

Hassan Moussa

hassan.moussa@st.ul.edu.lb

Lebanese University

Abbas El Dor

Lebanese University

Research Article

Keywords: Particle Swarm Optimization, Deep Learning Surrogates, Surrogate-Assisted Optimization, Partitioned Search Space, Adaptive Neighborhood Topology, Swarm Intelligence, Continuous Global Optimization

Posted Date: March 31st, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9236679/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Deep Learning Surrogate-Assisted Particle Swarm Optimization in Partitioned Search Spaces with Adaptive Neighborhood Topology for Continuous Optimization (PSA-PSO)

Hassan Moussa[1*] and Abbas El Dor[1*]

[1*]Department Of Applied Mathematics, Lebanese University, Beirut, Lebanon.

*Corresponding author(s). E-mail(s): hassan.moussa@st.ul.edu.lb; abbas.eldor@ul.edu.lb;

Abstract

In this paper, we propose Partitioned Surrogate Adaptive Particle Swarm Optimization (PSA-PSO), a hybrid optimization framework that integrates deep learning–based surrogate modeling, search space partitioning, and adaptive neighborhood topology to enhance performance in global continuous optimization. The proposed method first partitions the search space into multiple zones around the central search region. Within each zone, a deep neural network surrogate model is trained to approximate the objective function using sampled data. A gradient-based minimization procedure is then applied to the surrogate model to estimate promising local minima within each partition. These estimated minima are subsequently used to initialize the particles of the main PSO swarm, providing a diverse and informed starting population.

To further balance exploration and exploitation during the optimization process, PSA-PSO employs an adaptive ring topology in which the neighborhood size gradually expands over iterations, enabling controlled information sharing among particles.

The proposed framework is evaluated on a set of uni-modal and multi-modal continuous benchmark functions, including cases with coordinate rotation and shift transformations. Experimental results demonstrate that PSA-PSO achieves competitive performance and improved robustness compared to conventional PSO variants and several state-of-the-art continuous optimization algorithms.

Keywords: Particle Swarm Optimization, Deep Learning Surrogates, Surrogate-Assisted Optimization, Partitioned Search Space, Adaptive Neighborhood Topology, Swarm Intelligence, Continuous Global Optimization

1 Introduction

Particle Swarm Optimization (PSO) is a population-based metaheuristic widely applied to contiguous optimization problems due to its simplicity and competitive [1] .Despite its effectiveness, a fundamental challenge in PSO is achieving a proper balance between exploration and exploitation, particularly in high-dimensional search spaces where premature convergence to local minima may occur.Several variants of PSO have been proposed to address these limitations, including modifications to the swarm topology, adaptive parameter control, and hybridization with other optimization strategies. Among these approaches, adaptive neighborhood topologies have attracted considerable attention, as they regulate the flow of information among particles and help balance exploration and exploitation during the optimization process.

Another promising direction for improving optimization performance involves the use of surrogate models. Surrogate-assisted optimization techniques approximate the objective function using computational models such as regression methods, Gaussian processes, or neural networks. These approximations can guide the search process toward promising regions of the search space while reducing the number of expensive objective function evaluations (FEs) . With recent advances in deep learning, neural network-based surrogate models have demonstrated strong capabilities in capturing complex nonlinear relationships within optimization landscapes.

In addition, partitioning the search space into smaller regions can help improve exploration by allowing localized analysis of the objective function. By combining surrogate modeling with search space partitioning, it becomes possible to estimate promising candidate solutions in different regions of the search space before initializing the optimization algorithm.

Motivated by these observations, this paper proposes Partitioned Surrogate Adaptive Particle Swarm Optimization (PSA-PSO), a hybrid framework that integrates search space partitioning, deep learning-based surrogate modeling, and adaptive neighborhood topology. In the proposed approach, the search space is first divided into multiple zones around the central search region. For each zone, a deep neural network surrogate model is trained using sampled objective function evaluations. A gradient-based optimization procedure is then applied to the surrogate model to estimate promising local minima within each partition. These estimated solutions are used to initialize the particles of the PSO swarm, providing a diverse and informed starting population. Furthermore, PSA-PSO employs an adaptive ring topology in which the neighborhood size gradually increases during the optimization process, enabling controlled information exchange among particles and improving the balance

between exploration and exploitation.

The main contributions of this work can be summarized as follows:

- **Partitioned search space exploration**: A search space partitioning strategy is introduced to enable localized surrogate modeling and improve coverage of the optimization landscape.
- **Deep learning surrogate guidance**: Deep neural network surrogate models are trained within each partition to approximate the objective function and guide the estimation of promising candidate solutions.
- **Gradient-based particle initialization**: A gradient-based minimization procedure is applied to the surrogate models to generate high-quality initial particles for the PSO swarm.
- **Adaptive neighborhood topology**: An adaptive ring topology is employed to dynamically control information sharing among particles and enhance the balance between exploration and exploitation.

The proposed PSA-PSO framework is evaluated on a set of uni-modal and multi-modal continuous benchmark functions, including rotated and shifted variants, and compared against conventional PSO variants and state-of-the-art continuous optimization algorithms. In addition, the method is evaluated on the Lennard-Jones potential optimization problem, which is widely used as a challenging benchmark for testing global optimization algorithms in high-dimensional energy landscapes. The experimental results demonstrate that the proposed approach provides competitive performance and improved robustness in different optimization scenarios.

2 Related Work

2.1 Particle Swarm Optimization: Foundations

Particle swarm optimization was originally proposed by Kennedy and Eberhart in 1995 [1] as a population-based stochastic optimization algorithm inspired by the social behavior of flocking birds.PSO has since become one of the most widely applied swarm intelligence techniques for continuous global optimization due to its conceptual simplicity, low computation complexity, and competitive performance.

Consider a continuous optimization problem:

$$\min_{x \in \mathbf{R}^D} f(x)$$

Where D is the problem dimension. In PSO algorithm we define a swarm of N particle exploring the search space where each particle i has:

- Position vector $x_i = (x_{i,1}, \ldots, x_{i,D})$
- Velocity vector $v_i = (v_{i,1}, \ldots, v_{i,D})$
- Personal best position p_i
- Global best position g

The position and Velocity vector are randomly initialized at the beginning of the process, and the update equation is as follows:

$$v_i^{t+1} = wv_i^t + c_1 r_1 (p_i - x_i^t) + c_2 r_2 (g - x_i^t) \tag{1}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{2}$$

where:

- w is the inertia weight,
- c_1 and c_2 are cognitive and social acceleration coefficients,
- r_1 and $r_2 \sim \mathcal{U}(0,1)$ are random numbers,
- t denotes the iteration index.

Personal best of each particle p_i is updated as follows in case of minimization:

$$\mathbf{P}_i(t+1) = \begin{cases} \mathbf{x}_i(t+1), & \text{if } f(\mathbf{x}_i(t+1)) < f(\mathbf{P}_i(t)) \\ \mathbf{P}_i(t), & \text{otherwise} \end{cases} \tag{3}$$

The global best position Gbest, found by the evaluations of the particles during each generation, is defined as:

$$\mathbf{G}(t) = \arg \min_{i=1,\ldots,s} f(\mathbf{P}_i(t)) \tag{4}$$

Algorithm 1 Original Particle Swarm Optimization (PSO)

1: Initialize randomly the position $\mathbf{x}_i$ and velocity $\mathbf{v}_i$ of each particle; set iteration $= 1$
2: **for** $i = 1$ to s **do**
3: Set personal best $\mathbf{P}_{best_i} = \mathbf{x}_i$
4: **end for**
5: **while** stopping criterion not satisfied **do**
6: **for** $i = 1$ to s **do**
7: Update position $\mathbf{x}_i$ using equations (1) and (2)
8: Calculate particle fitness $f(\mathbf{x}_i)$
9: Update personal best $\mathbf{P}_{best_i}$ and global best $\mathbf{G}_{best}$ using equations (3) and (4)
10: **end for**
11: iteration = iteration +1
12: **end while**

2.2 Surrogate-Assisted Particle Swarm Optimization

Surrogate optimization - has been widely studied as a means to reduce the computational cost of population –based search algorithms, especially when evaluations of the

true objective function are expensive. In the context of particle swarm optimization (PSO), researchers have explored a variety of surrogate modeling strategies to guide the search and improve efficiency.

One of the early surrogate-assisted PSO algorithms is the two-layer surrogate-assisted PSO (TLSAPSO) [2], It uses traditional surrogate models such as global and local regression-type surrogates (e.g., radial basis functions or other inexpensive approximators) to approximate the objective function to approximate the objective function and accelerate convergence. In TLSAPSO, a global surrogate smooths the overall landscape while local surrogates provide more accurate local estimations around particles, demonstrating improved performance on multi-modal benchmarks compared to standard PSO.

Similarly, Yang et al. proposed an improved surrogate-assisted PSO (ISAPSO) that combines global and local surrogate models. The global surrogate guides the coarse search across the space, while the local surrogate performs refined searches near promising solutions, effectively balancing exploration and exploitation in expensive black-box problems [3]. Related efforts to handle high-dimensional expensive problems use surrogate models in conjunction with PSO to guide the search toward promising regions while controlling computational cost. [4]

In addition to traditional surrogate-assisted PSO algorithms using simple regression or Gaussian process models, there is a growing interest in integrating **deep neural network-based surrogates** with PSO to capture complex nonlinear mappings and reduce expensive evaluations. One example is an inverse design framework that employs a deep convolutional neural network as a surrogate solver to model the simulation process and assists a PSO search to efficiently generate optimal designs, significantly reducing the need for costly simulations [5].

Similarly, artificial neural network (ANN) surrogate models have been paired with PSO in engineering optimization applications, where the ANN is trained to approximate the objective function based on data from expensive simulations, and PSO is then used to search the approximated landscape to locate optimal solutions with fewer true evaluations [6]. While these approaches demonstrate the potential of neural network surrogates in guiding PSO, they are often developed for specific applications rather than general continuous optimization benchmarking, highlighting an opportunity for further methodological development.

The proposed PSA-PSO framework builds on this trend by using deep learning surrogates not only to approximate objective functions but also to perform gradient-based minimization within partitioned regions to systematically initialize the swarm for improved global search performance.

2.3 Partitioned Search Space And Adaptive Topology

Methods that partition the search space or adapt the swarm topology have been proposed to enhance the exploration and diversity of PSO in complex optimization landscapes. One notable example is the multi-swarm PSO variant by ElDor etal., which divides the search space into multiple regions and employs charged particles within each zone to increase diversity. In this approach, auxiliary sub-swarms are initialized in different areas of the search space and interact via an electrostatic repulsion heuristic, while a main swarm consolidates the best particles from these sub-swarms, leading to improved performance on unimodal and multimodal benchmark functions as well as the Lennard-Jones potential problem [7]

In addition to explicit search space partitioning, adaptive topology strategies focus on dynamically modifying the communication structure among particles to balance exploration and exploitation. For instance, Wu et al. propose incorporating a Euclidean distance-based adaptive topology into PSO, where each particle connects only to its closest neighbors at each iteration. This adaptive network structure, combined with a Surprisingly Popular Algorithm metric, allows the swarm to maintain diversity while accelerating convergence across problems of varying dimensionality [8].

Other research has explored adaptive neighborhood and clustering mechanisms to further improve PSO performance. Liang et al. introduce a clustering-based adaptive PSO (APSO-C) that dynamically groups particles using k-means and uses ring neighborhoods for information exchange, adjusting inertia and control parameters based on cluster states to enhance search balance [9].

These works illustrate distinct but related strategies — explicitly dividing the search space or adapting swarm connectivity — to guide particle interactions and improve search effectiveness in complex continuous optimization tasks. The proposed PSA-PSO framework builds on these ideas by combining partitioned initialization with an adaptive ring topology that gradually expands neighborhood connections throughout the optimization process.

3 The Proposed Method

In this work, we propose the **Partitioned Surrogate Adaptive Particle Swarm Optimization (PSA-PSO)**, a hybrid framework that integrates *partitioned search spaces*, *deep learning surrogate modeling*, and *adaptive neighborhood topology* to improve the efficiency and robustness of continuous global optimization.

The proposed method begins by dividing the original search space into multiple **subspaces** centered around regions of interest. Within each subspace, a deep learning surrogate model is trained to approximate the objective function using sampled data points. Once the surrogate model is trained, a gradient-based minimization procedure is applied to the surrogate within that subspace to identify promising local minima. Each of these minima is then used to initialize a particle in the main PSO swarm,

effectively providing a diverse and informed starting population.

After initialization, the main PSO is executed using an **adaptive neighborhood topology**. The neighborhood of each particle begins with a minimal size to encourage local exploration and gradually expands up to a maximum neighborhood size, enabling controlled information sharing and facilitating a balance between exploration and exploitation throughout the optimization process.

The proposed framework is modular, allowing each component of PSA-PSO to be studied and optimized individually. The structure of the method is described in details in the following sections.

3.1 Search Space Partitioning

The first stage of our proposed algorithm involves partitioning the search space into multiple subspaces. This partitioning increases the diversity of candidate particles, enhancing the exploration capability of the algorithm and providing a structured framework for training deep learning-based surrogate neural networks within each subspace. The details of the partitioning methodology are described in the following subsection.

We consider a search space defined as $[min, max]^N$, where N represents the dimensionality of the problem. This search space is partitioned into max_{zone} subspaces (or zones) centered around a central point. The center of the search space is computed using Equation (5), and a step size for partitioning is determined based on half of the search range, as shown in Equations (6) and (7):

$$center = (max + min)/2 \tag{5}$$

$$half_{range} = 0.5 * (max - min) \tag{6}$$

$$step = half_{range}/max_{zone} \tag{7}$$

Each subspace, denoted as $zone^i = [zone^i_{min}, zone^i_{max}]^N$, is defined according to Equations (8) and (9), where $i = 1, 2, \ldots, max_{zone}$ indicates the index of the current zone:

$$zone^i_{min} = center - i * step \tag{8}$$

$$zone^i_{max} = center + i * step \tag{9}$$

As shown in Figure 1, the search space is divided into multiple zones...

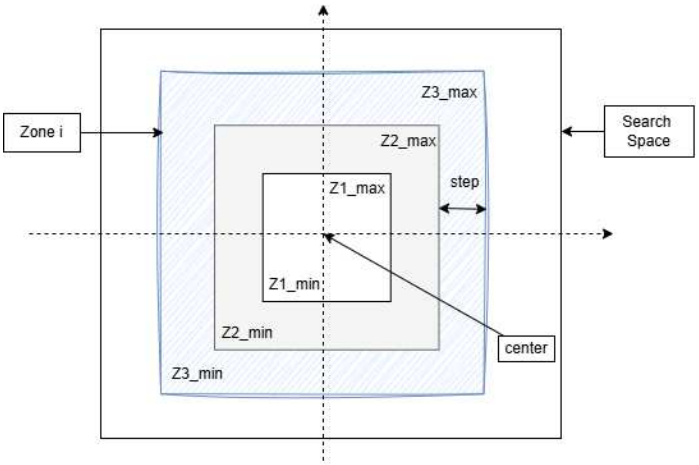


Fig. 1 Illustration of the search space partitioning into multiple subspaces.

The size of the main swarm is determined by max_{zone}, with each particle corresponding to the local minimum obtained from gradient-based optimization of the deep learning surrogate within its respective zone i.

Algorithm 2 Search Space Partitioning Algorithm for PSA-PSO

Input: $bounds = [min, max]$, max_zone
$center \leftarrow (bounds[1] + bounds[0])/2$
$half_range \leftarrow 0.5 \times (bounds[1] - bounds[0])$
$step \leftarrow half_range/max_zone$
for $i = 1$ to max_zone **do**
    $zone_min \leftarrow center - i \times step$
    $zone_max \leftarrow center + i \times step$
    Store $zone^i = [zone_min, zone_max]^N$
end for

3.2 Deep Learning Neural Network Surrogate Training

Once the search space has been partitioned into subspaces as described in the previous section, a deep learning neural network surrogate is trained within each zone. The surrogate model is designed to approximate the true objective function using a set of sampled points from the corresponding subspace. By learning the underlying mapping between input variables and objective values, the surrogate enables efficient evaluation of candidate solutions within the zone, reducing the need for expensive direct function evaluations. This stage provides the foundation for subsequent gradient-based minimization, allowing the algorithm to identify promising local minima in each subspace to initialize the main PSO swarm.

For each subspace, we employ a *feedforward multi-layer perceptron (MLP)* as the deep learning surrogate to approximate the objective function. The MLP takes as input a vector of dimension D, corresponding to the number of decision variables in the search space, and outputs a scalar value representing the predicted objective function.

The architecture of the MLP consists of four fully connected layers. The first three layers each have 256 neurons and are followed by a Rectified Linear Unit (ReLU) (equation 10) activation function to introduce nonlinearity. The final layer has a single neuron that outputs the predicted objective value without activation.

$$\sigma(x) = \max(0, x) \tag{10}$$

This deep architecture allows the surrogate to capture complex nonlinear mappings within each subspace, providing accurate approximations of the objective function that are used for subsequent gradient-based minimization to locate local minima.

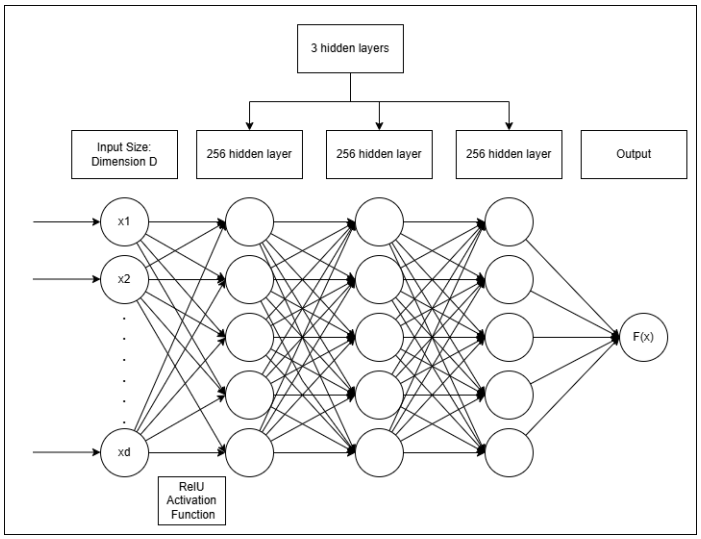


Fig. 2 The architecture of Neural Network Surrogate used in estimating objective function

For each subspace, the MLP surrogate is trained to approximate the objective function using a set of sampled points. The training procedure consists of the following steps:

1. Generate n random input samples uniformly within the bounds of the current subspace.
2. Evaluate the objective function on these samples to obtain the corresponding target values. Non-finite values are replaced with a large constant to maintain numerical stability.

3. Standardize both the input samples and the target values by subtracting the mean and dividing by the standard deviation to stabilize training.
4. Train the MLP using the mean squared error (MSE) loss function and the Adam optimizer over a fixed number of epochs with mini-batches of size bs.
5. After training, the model and the scaling parameters (mean and standard deviation of inputs and outputs) are stored for later use in gradient-based minimization.

This procedure allows the surrogate to learn an accurate approximation of the objective function within each subspace, enabling efficient local optimization without repeated evaluations of the true objective.

The neural network surrogate learns an approximation of the objective function by minimizing the mean squared error (MSE) between its predictions and the true function values. Specifically, given a set of input samples $X = \{x_1, x_2, \ldots, x_n\}$ drawn from a subspace and their corresponding objective values $F(X) = \{f(x_1), f(x_2), \ldots, f(x_n)\}$, the MLP adjusts its weights and biases to minimize the following loss function:

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \Big(f_\theta(x_i) - f(x_i) \Big)^2, \tag{11}$$

where $f_\theta(x_i)$ denotes the network's prediction for input x_i and θ represents all trainable parameters of the network. By iteratively updating θ using the Adam optimizer on mini-batches of the training data, the network learns the mapping from input vectors to objective function values, effectively capturing the local behavior of the objective within the subspace. Once trained, the surrogate can provide accurate predictions for unseen inputs in the subspace, enabling efficient gradient-based optimization.

Algorithm 3 Training Deep Learning Surrogate in a Subspace

Input: MLP model f_θ, input dimension D, number of samples n, number of epochs *epochs*, learning rate lr, subspace bounds $[min, max]$, objective function F
Output: Trained surrogate f_θ and scalers $(X_{mean}, X_{std}, y_{mean}, y_{std})$
Generate n random samples X uniformly in $[min, max]^D$
Evaluate $y = F(X)$, replace non-finite values with a large constant
Compute $X_{mean}, X_{std}, y_{mean}, y_{std}$ for standardization
Standardize: $X_s = (X - X_{mean})/X_{std}, \quad y_s = (y - y_{mean})/y_{std}$
Initialize Adam optimizer with learning rate lr and MSE loss function
for epoch = 1 to *epochs* **do**
    Shuffle the samples
    for each mini-batch b in X_s, y_s **do**
        Compute predictions: $\hat{y} = f_\theta(X_s[b])$
        Compute loss: $L = \text{MSE}(\hat{y}, y_s[b])$
        Update model parameters using Adam optimizer
    end for
end for
return trained model f_θ and scalers $(X_{mean}, X_{std}, y_{mean}, y_{std})$

The output of this stage is the trained neural network surrogate along with the scaling parameters for the input and output variables. In the next stage, this surrogate will be used in a gradient-based minimization procedure to identify a local minimum within the corresponding subspace. Each minimum obtained in this way is then used to initialize a particle in the main PSO swarm, ensuring that the swarm starts with informed candidate solutions derived from the surrogate models.

3.3 Minimizing The Trained Neural Network Surrogate

After training the deep learning surrogate for each subspace, we employ a gradient-based minimization procedure to identify local minima within the subspace. The surrogate provides a differentiable approximation of the objective function, allowing the use of gradient descent to efficiently locate promising points.

To improve robustness and avoid local optima, multiple random restarts are performed within the bounds of the subspace. At each restart, a candidate solution is initialized randomly and iteratively updated using the Adam optimizer, with gradients computed on the standardized surrogate output. After the optimization steps, the solution is transformed back to the original scale of the objective function. The minimum across all restarts is selected as the representative local minimum of the subspace.

Let $f_\theta(x)$ denote the trained surrogate model with parameters θ, and let $F(x)$ be the original objective function. The goal is to find a local minimum x^* of the surrogate within a given subspace $[min, max]^D$. The minimization problem can be formulated as:

$$x^* = \arg \min_{x \in [min,max]^D} f_\theta(x) \tag{12}$$

Gradient-based optimization is applied to solve this problem. Starting from an initial point x_0, we iteratively update the candidate solution using:

$$x_{t+1} = \text{Proj}_{[min,max]}\Big(x_t - \eta \nabla_x f_\theta(x_t)\Big), \tag{13}$$

where η is the learning rate, $\nabla_x f_\theta(x_t)$ is the gradient of the surrogate output with respect to x_t, and $\text{Proj}_{[min,max]}(\cdot)$ denotes projection of the updated solution back into the subspace bounds to ensure feasibility.

To improve robustness and reduce the chance of poor local minima, the procedure is repeated for multiple random restarts $\{x_0^1, x_0^2, \ldots, x_0^{n_{restarts}}\}$. The best solution across all restarts is selected as the surrogate minimum:

$$x^* = \arg \min_{i=1,\ldots,n_{restarts}} f_\theta(x_T^i), \tag{14}$$

where x_T^i is the final solution of the i-th restart after T gradient steps.

Finally, the quality of the identified minimum is evaluated against the original objective function $F(x)$:

$$F^* = F(x^*), \tag{15}$$

ensuring that the surrogate-guided solution is indeed promising in terms of the true objective. Each such minimum x^* is then used as a particle in the main PSO swarm, providing an informed initialization for the global optimization stage.

Algorithm 4 Gradient-Based Minimization of the Trained Surrogate

Input: Trained surrogate f_θ, scalers $(X_{mean}, X_{std}, y_{mean}, y_{std})$, input dimension D, bounds $[min, max]$, learning rate lr, number of steps $steps$, number of restarts $n_{restarts}$
Output: Local minimum x^* and corresponding objective value $f(x^*)$
Initialize $best_x \leftarrow None$, $best_val \leftarrow \infty$
for restart $= 1$ to $n_{restarts}$ **do**
    Initialize x_0 randomly in $[min, max]^D$ with gradients enabled
    Initialize Adam optimizer on x_0 with learning rate lr
    for step $= 1$ to $steps$ **do**
        Normalize input: $x_{norm} = (x_0 - X_{mean})/X_{std}$
        Compute surrogate prediction: $\hat{y} = f_\theta(x_{norm})$
        Set loss $L = \hat{y}$
        Update x_0 using Adam optimizer to minimize L
        Clamp x_0 within bounds $[min, max]$
    end for
    Transform x_0 back to original scale and evaluate $f(x_0)$
    if $f(x_0) < best_val$ **then**
        $best_val \leftarrow f(x_0)$, $best_x \leftarrow x_0$
    end if
end for
return $best_x, best_val$

3.4 PSA-PSO Main Algorithm

After obtaining the local minima from the gradient-based minimization of the surrogate models in each subspace, these minima are used to initialize the particles of the main Particle Swarm Optimization (PSO) swarm. Let the set of minima be $\{x_1^*, x_2^*, \ldots, x_{max_{zone}}^*\}$, where each x_i^* corresponds to the local minimum from subspace i. The main swarm thus has a size equal to the number of subspaces max_{zone}, and all particles are positioned in promising regions of the search space based on surrogate guidance.

To further enhance exploration and exploitation, the main PSO employs an **adaptive ring topology**. At the beginning of the optimization, each particle communicates only with a minimal number of neighbors, promoting independent exploration. As iterations progress, the neighborhood size gradually increases up to a maximum, potentially reaching a fully connected topology. This adaptive mechanism allows controlled information sharing among particles, balancing global exploration and local exploitation.

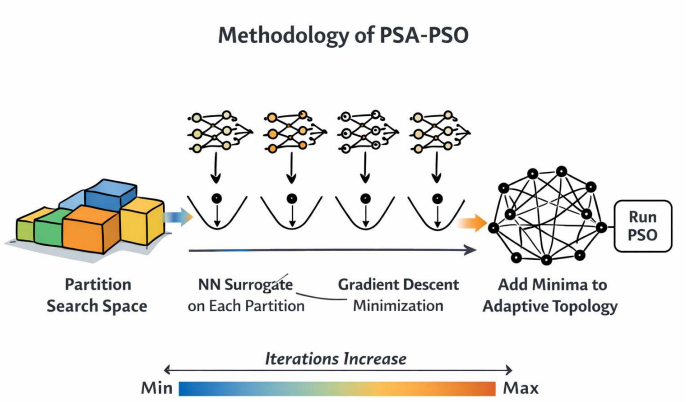


Fig. 3 Methodology of PSA-PSO

To enhance the balance between exploration and exploitation, the main swarm in PSA-PSO employs an **adaptive ring topology**. In the first iteration, each particle interacts only with a minimal number of neighbors, defined by a ring radius $k_{\min}$. This allows particles to explore locally and independently, preventing premature convergence. As the optimization progresses, the neighborhood radius increases gradually until it reaches a maximum value $k_{\max}$, at which point the topology can approach full connectivity. The adaptive growth of the neighborhood promotes controlled information sharing among particles, allowing the swarm to transition from local exploration to coordinated global search.

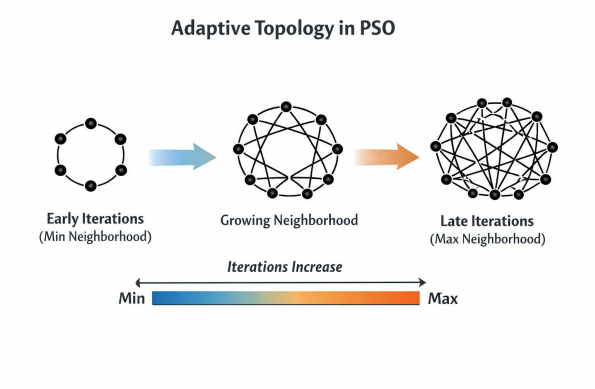


Fig. 4 Adaptive Ring topology used on PSA-PSO

Formally, let the neighborhood radius at iteration t be $k(t)$:

$$k(t) = k_{\min} + (k_{\max} - k_{\min}) \left(\frac{t}{T} \right)^{\alpha} \tag{16}$$

where T is the maximum number of iterations, and α is a scaling exponent (e.g., $\alpha = 2$) to control the growth rate. Each particle i identifies the best personal best among its neighbors:

$$Lbest_i = \arg \min_{j \in \mathcal{N}_i(k(t))} f(Pbest_j) \tag{17}$$

where $\mathcal{N}_i(k(t))$ denotes the set of particle indices within the ring neighborhood of radius $k(t)$ around particle i. The velocity and position updates then follow the standard PSO equations, but using $Lbest_i$ instead of the global best.

Algorithm 5 Adaptive Topology Particle Swarm Optimization (AT-PSO)

Input: Objective function F, initial particle positions $\{x_i\}_{i=1}^{N}$, bounds $[min, max]^D$, PSO parameters w, c_1, c_2, maximum iterations T, $k_{\min}, k_{\max}$
Output: Best position x^* and corresponding objective $F(x^*)$
Initialize particle velocities $v_i = 0$, personal best $Pbest_i = x_i$, and compute $F(Pbest_i)$
for iteration $t = 1$ to T **do**
    Update neighborhood radius: $k(t) = k_{\min} + (k_{\max} - k_{\min})(t/T)^{\alpha}$
    for each particle $i = 1$ to N **do**
        Identify neighborhood $\mathcal{N}_i(k(t))$ (indices within ring of radius $k(t)$)
        Determine local best: $Lbest_i = \arg\min_{j \in \mathcal{N}_i(k(t))} F(Pbest_j)$
        Generate random vectors $r_1, r_2 \sim U[0,1]^D$
        Update velocity:

$$v_i \leftarrow w v_i + c_1 r_1 (Pbest_i - x_i) + c_2 r_2 (Lbest_i - x_i)$$

        Update position: $x_i \leftarrow x_i + v_i$
        Apply bounds: $x_i \leftarrow \text{clip}(x_i, min, max)$
        Evaluate $F(x_i)$
        if $F(x_i) < F(Pbest_i)$ **then**
            Update personal best: $Pbest_i \leftarrow x_i$
        end if
    end for
end for
return $x^* = \arg\min_i F(Pbest_i), \quad F(x^*)$

The complete PSA-PSO algorithm integrates subspace partitioning, deep learning surrogate modeling, gradient-based minimization, and adaptive topology PSO to efficiently solve high-dimensional continuous optimization problems. The procedure consists of the following steps:

1. **Subspace Partitioning:** The search space $[min, max]^D$ is divided into max_{zone} subspaces.
2. **Surrogate Training:** For each subspace, a deep neural network surrogate (MLP) is trained on n randomly sampled points using the mean squared error loss between the surrogate output and the true objective function.
3. **Gradient-Based Minimization:** Each trained surrogate is minimized using gradient descent with multiple random restarts to identify a local minimum within the subspace. These minima form the initial particle positions for the main PSO swarm.
4. **Main PSO Optimization:** The main swarm is initialized with the surrogate minima and evolves according to the PSO update equations. The algorithm employs an adaptive ring topology where the neighborhood radius starts small and gradually increases, allowing controlled information sharing among particles.
5. **Performance Evaluation:** The best solution obtained by the main swarm is evaluated against the original objective function. Multiple independent runs are performed to assess robustness and convergence.

Algorithm 6 Partitioned Surrogate Adaptive PSO (PSA-PSO)

Input: Objective function F, search bounds $[min, max]^D$, dimension D, number of subspaces max_{zone}, PSO parameters (w, c_1, c_2), max PSO iterations T, number of runs R
Output: Best position x^* and objective value $F(x^*)$
Initialize empty particle set: $\mathcal{P} \leftarrow \emptyset$
for subspace $i = 1$ to max_{zone} **do**
    Define subspace bounds: $zone^i_{min}, zone^i_{max}$
    Train MLP surrogate f_θ in $[zone^i_{min}, zone^i_{max}]$
    Minimize surrogate via gradient descent: $x_i^* = \arg\min f_\theta(x)$
    Add x_i^* to particle set: $\mathcal{P} \leftarrow \mathcal{P} \cup \{x_i^*\}$
end for
Initialize AT-PSO swarm with particles $\mathcal{P}$ and bounds $[min, max]^D$
Run AT-PSO based on algorithm (5) and particles $\mathcal{P}$.
return overall best position x^* and value $F(x^*)$

4 Experiments and discussion

4.1 Benchmark Functions

In this section, we present the benchmark functions used in our experiments. These functions are categorized into six main groups and are widely used in the optimization literature. They serve as standard test problems to evaluate and compare the performance of the proposed PSA-PSO algorithm against other state-of-the-art optimization methods.

4.1.1 Group A: Unimodal Functions

1. Sphere Function

$$F_1(x) = \sum_{i=1}^{D} x_i^2 \tag{18}$$

4.1.2 Group B: Unrotated multimodal problems

2. Rosenbrock's function

$$F_2(x) = \sum_{i=1}^{D-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right] \tag{19}$$

3. Ackley's function

$$F_3(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e \tag{20}$$

4. Rastrigin's function

$$F_4(x) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right) \tag{21}$$

5. Griewank's function

$$F_5(x) = 1 + \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) \tag{22}$$

4.1.3 Group C: Rotated multimodal problems

To increase the difficulty of benchmark problems and avoid axis-aligned bias, benchmark functions can be rotated using an orthogonal transformation. According to the method proposed by Salomon [10], an orthogonal rotation matrix M is generated and applied to the decision variable vector x. The rotated variable is defined as

$$y = Mx \tag{23}$$

where x represents the original decision vector and M is an orthogonal matrix satisfying $M^T M = I$. The fitness value of the rotated function is then computed using the transformed variable y.

In many classical benchmark problems, the objective function is separable, meaning that it can be optimized through independent one-dimensional searches along each coordinate axis. However, when the coordinate system is rotated, the variables become interdependent and the function is no longer separable. As a result, the optimization problem becomes significantly more challenging, since algorithms can no longer optimize each dimension independently.

The rotated benchmark functions are therefore widely used to evaluate the robustness and effectiveness of optimization algorithms under non-separable and more complex search landscapes. The rotated functions used in this work are defined as follows.

6. Rotated Rosenbrock's function

$$F_6(x) = \sum_{i=1}^{D-1} \left[100 \left(y_{i+1} - y_i^2 \right)^2 + (y_i - 1)^2 \right] \tag{24}$$

7.Rotated Ackley's function

$$F_7(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} y_i^2} \right) - \exp\left(\frac{1}{D} \sum_{i=1}^{D} \cos(2\pi y_i) \right) + 20 + e \tag{25}$$

8. Rotated Rastrigin's function

$$F_8(x) = \sum_{i=1}^{D} \left(y_i^2 - 10 \cos(2\pi y_i) + 10 \right) \tag{26}$$

9. Rotated Griewank's function

$$F_9(x) = 1 + \frac{1}{4000} \sum_{i=1}^{D} y_i^2 - \prod_{i=1}^{D} \cos\left(\frac{y_i}{\sqrt{i}} \right) \tag{27}$$

4.1.4 Group D: Shifted multimodal problems

In this group of benchmark functions, the original decision variables are translated by a shift vector

$$O = (o_1, o_2, \ldots, o_D)$$

and a bias term f_{bias} is added to the function values. This transformation moves the global optimum to a new location, making the problem more challenging for optimization algorithms, as the search cannot exploit any inherent symmetry or origin-centered structure of the original function. The use of shifted functions prevents algorithms from prematurely converging toward the origin and encourages more robust exploration of the search space. The shift vectors used in this study are obtained from the CEC 2005 benchmark suite [11], which is a standard source for shifted and rotated benchmark problems in evolutionary computation. Examples of functions in this group are listed below.

10. Shifted Rosenbrock's function

$$F_{10}(x) = \sum_{i=1}^{D-1} \left[100(z_{i+1} - z_i^2)^2 + (z_i - 1)^2\right] \tag{28}$$

$$z_i = x_i - o_{i,j}$$

11. Shifted Ackley's function

$$F_{11}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)\right) + 20 + e \tag{29}$$

$$z_i = x_i - o_{i,j}$$

12. Shifted Rastrigin's function

$$F_{12}(x) = \sum_{i=1}^{D} \left(z_i^2 - 10\cos(2\pi z_i) + 10\right) \tag{30}$$

$$z_i = x_i - o_{i,j}$$

13. Shifted Griewank's function

$$F_{13}(x) = 1 + \frac{1}{4000}\sum_{i=1}^{D} z_i^2 - \prod_{i=1}^{D}\cos\left(\frac{z_i}{\sqrt{i}}\right) \tag{31}$$

$$z_i = x_i - o_{i,j}$$

4.1.5 Group E: Shifted unimodal problem

14. Shifted sphere function

$$F_{14}(x) = \sum_{i=1}^{D} z_i^2 \tag{32}$$

$$z_i = x_i - o_{i,j}$$

4.1.6 Group F : The Lennard-Jones atomic cluster problem

The Lennard-Jones (LJ) atomic cluster problem is one of the simplest and most widely studied molecular conformation problems. Its objective is to find a spatial arrangement of atoms in a cluster or molecule that minimizes the total potential energy [12].

Let K denote the number of atoms in the cluster, and let $x_i = (x_{i1}, x_{i2}, x_{i3}) \in \mathbb{R}^3$ represent the coordinates of atom i in three-dimensional space. The complete configuration of the cluster is then represented by

$$X = (x_1, x_2, \ldots, x_K).$$

The interaction between each pair of atoms (i, j) is modeled using the Lennard-Jones potential function:

$$v(r_{ij}) = \frac{1}{r_{ij}^{12}} - \frac{1}{r_{ij}^{6}}, \quad 1 \leq i < j \leq K \tag{33}$$

where $r_{ij} = \|x_i - x_j\|$ is the Euclidean distance between atoms i and j.

For a cluster of K atoms, the total potential energy to be minimized is the sum of the pairwise LJ potentials:

$$V_K(X) = \sum_{1 \leq i < j \leq K} v(r_{ij}) = \sum_{1 \leq i < j \leq K} \left(\frac{1}{\|x_i - x_j\|^{12}} - \frac{1}{\|x_i - x_j\|^{6}} \right) \tag{34}$$

Although the LJ potential for a single pair of atoms is unimodal, the problem becomes highly complex for clusters with a large number of atoms. The interactions among all pairs produce a **rugged energy landscape** with many local minima, making the global optimization of $V_K(X)$ a challenging task.

This function is commonly used to evaluate global optimization algorithms in molecular modeling and physics simulations due to its **non-separability, multi-modality, and high dimensionality**.

The function name, parameters of the functions, the global optimum x*, the corresponding fitness value f (x*), and the search space [Xmin, Xmax] are given in Table 1.

We present also the shape of some benchmark functions in the figure Fig 5.

Table 1 Parameters of the benchmark functions

Function	Global Optimum x^*	$f(x^*)$	Search Space
F_1 Sphere	$(0, 0, \ldots, 0)$	0	$[-100, 100]^n$
F_2 Rosenbrock's	$(1, 1, \ldots, 1)$	0	$[-5, 10]^n$
F_3 Ackley's	$(0, 0, \ldots, 0)$	0	$[-32.768, 32.768]^n$
F_4 Rastrigin's	$(0, 0, \ldots, 0)$	0	$[-5.12, 5.12]^n$
F_5 Griewank's	$(0, 0, \ldots, 0)$	0	$[-600, 600]^n$
F_6 Rotated Rosenbrock's function	$(1, 1, \ldots, 1)$	0	$[-5, 10]^n$
F_7 Rotated Ackley's function	$(0, 0, \ldots, 0)$	0	$[-32.768, 32.768]^n$
F_8 Rotated Rastrigin's function	$(0, 0, \ldots, 0)$	0	$[-5.12, 5.12]^n$
F_9 Rotated Griewank's function	$(0, 0, \ldots, 0)$	0	$[-600, 600]^n$
F_10 Shifted Rosenbrock's function	$(o11, o12, \ldots, o_n)$	390	$[-5, 10]^n$
F_11 Shifted Ackley's function	$(o11, o12, \ldots, o_n)$	-140	$[-32.768, 32.768]^n$
F_12 Shifted Rastrigin's function	$(o11, o12, \ldots, o_n)$	-330	$[-5.12, 5.12]^n$
F_13 Shifted Griewank's function	$(o11, o12, \ldots, o_n)$	-180	$[-600, 600]^n$
F_14 Shifted Sphere function	$(o11, o12, \ldots, o_n)$	-450	$[-100, 100]^n$
V_8 Clustered Lennard Jones	$(x1, x2, \ldots, x24)$	-19.821489	$[-2, 2]^{24}$
V_9 Clustered Lennard Jones	$(x1, x2, \ldots, x27)$	-24.113360	$[-2, 2]^{27}$
V_{10} Clustered Lennard Jones	$(x1, x2, \ldots, x30)$	-28.422532	$[-2, 2]^{30}$

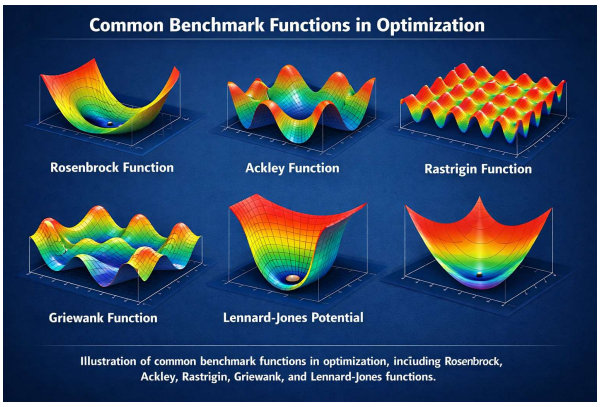


Fig. 5 Illustration of common benchmark functions used in optimization, including Rosenbrock, Ackley, Rastrigin, Griewank, and Lennard-Jones potential. Each surface highlights the characteristic landscape of the function, demonstrating valleys, peaks, and multi-modal features relevant for testing optimization algorithms.

4.2 Algorithm Parameter Settings

All experiments were conducted on a virtual machine equipped with 4 vCPUs and 16 GB of RAM. The computational environment ensures consistent and reproducible performance evaluation across all benchmark functions and algorithms.

The proposed PSA-PSO algorithm was evaluated against several state-of-the-art population-based optimization algorithms, with comparisons varying depending on the function groups:

- **Group A (Unimodal), Group B (Multimodal), and Group C (Rotated Functions):** PSA-PSO was compared with CLPSO, CPSO, UPSO, standard PSO 2007, standard PSO 2011, Differential Evolution (DE), and APSO. These algorithms represent a range of swarm intelligence strategies, including classical and modern PSO variants. Specifically:
  - Std. PSO 2007 utilize a random *lbest* topology to exchange information between particles.
  - UPSO employs a combination of *lbest* and *gbest* topologies.
  - CLPSO uses the comprehensive learning strategy, which abandons the global best information and updates particle velocities using other particles' past best information.
- **Group B (Multimodal) and Group D (Shifted Functions):** PSA-PSO was compared with standard PSO 2007 and standard PSO 2011. This comparison evaluates the algorithm's ability to handle shifted landscapes, where global optima are displaced from the origin.
- **Lennard-Jones benchmark:** PSA-PSO was tested against standard PSO 2007 and standard PSO 2011 to evaluate its performance on molecular potential optimization problems.

All algorithms, including PSA-PSO, were run for multiple independent trials to account for stochastic variability. Performance metrics such as best-found fitness, mean fitness, and standard deviation were recorded to ensure statistical reliability.

4.2.1 Parameter Of PSO variant algorithms

In this section, we present the parameters of the PSO variant algorithms against which our proposed PSA-PSO was compared. The parameters include the inertia weight (w) and the acceleration coefficients (c_1 and c_2), which play a crucial role in controlling the exploration and exploitation balance in particle swarm optimization. These settings are based on standard values reported in the literature for each algorithm.

Table 2 Parameter settings of PSO variant algorithms used in the experiments.

Algorithm	**Inertia Weight (w)**	**Cognitive Coefficient (c_1)**	**Social Coefficient (c_2)**
Std. PSO 2007	$\frac{1}{2\ln 2}$	$0.5 + ln2$	$0.5 + ln2$
Std. PSO 2011	$\frac{1}{2\ln 2}$	$0.5 + ln2$	$0.5 + ln2$
CLPSO	$0.9 - \frac{0.5*g}{max_{gen}}$	1.49	1.49
CPSO	$0.9 - \frac{0.5*g}{max_{gen}}$	1.49	1.49
UPSO	0.72	1.49	1.49

4.2.2 Parameter Of Differential Evolution

In this section, we present the parameters used for the Differential Evolution (DE) algorithm in our comparisons. The DE algorithm was configured with standard settings commonly reported in the literature to ensure fair comparison with the PSO variants.

Table 3 Parameter settings of Differential Evolution Algorithm .

Algorithm	Pop Size	Max FEs	CR
Differential Evolution	200	40k	0.9

4.2.3 Parameter Of PSA-PSO

In this section, we present the parameters used for our PSA-PSO algorithm. The parameters include both the surrogate model configuration and the main PSO algorithm settings.

Deep Neural Network Surrogate Parameters

The neural network surrogate was trained using the following settings:

Parameter	Value
Number of Sample Points	500
Number of Epochs	50
Learning Rate	1×10^{-3}

Table 4 Parameters of the deep neural network surrogate used in PSA-PSO.

Gradient Minimizer Parameters

The gradient-based minimization on each surrogate model was performed with:

Main PSA-PSO Algorithm Parameters

For the main PSO algorithm, the parameters were set as follows:

4.3 Experimental results

4.3.1 Results on the 10-D problems

In this section, we present the results of our experiments on the 10-dimensional (10-D) problems for test function groups A, B, and C. The performance of our PSA-PSO

Parameter	Value
Number of Steps	50
Learning Rate	0.05
Number of Restarts	3

Table 5 Parameters of the gradient minimizer used on each surrogate.

Parameter	Value
Inertia Weight (w)	$\frac{1}{2\ln 2}$
Acceleration Coefficients ($c_1 = c_2$)	$0.5 + \ln 2$
Minimum Neighborhood ($k_{\min}$)	1
Maximum Neighborhood ($k_{\max}$)	Number of zones
Number of Particles	Number of zones

Table 6 Main PSA-PSO algorithm parameters used in the experiments.

algorithm is compared against other algorithms, including CLPSO, CPSO, UPSO, Std. PSO 2007, Std. PSO 2011, and Differential Evolution, depending on the group.

Each experiment was repeated for 30 independent runs to ensure statistical reliability. For our algorithm, a maximum of 40,000 function evaluations (FEs) was used for each run. The best results among all compared algorithms are highlighted in **bold** in the tables.

The results of the algorithms on the 10-D benchmark functions are summarized in Table 8.

As shown in Table 7, PSA-PSO achieved the best performance on 6 out of 9 benchmark functions, second-best on 2 functions, and third-best on 1 function. The Friedman ranking further confirms its efficiency, with the lowest average rank of 1.44 among all compared algorithms.

Table 7 Friedman Test Ranking of the Algorithms on 10-D Problems (lower rank = better performance)

Algorithm	Friedman Rank
PSA-PSO	**1.44**
STD PSO 2011	3.22
CLPSO	3.89
UPSO	4.22
Std. PSO2007	5.22
CPSO	5.44
APSO	5.56
DE	7.00

Table 8 Results on the 10-D problems

Function	PSA - PSO	CLPSO	CPSO	UPSO	Std. PSO2007	STD PSO 2011	DE	APSO
f1	**1.118744E-31** ± **1.012089E-31**	4.020000E-21 ± 6.490000E-21	9.010000E-10 ± 1.380000E-09	3.410000E-10 ± 1.580000E-09	7.410000E-10 ± 2.580000E-09	9.773290E-22 ± 9.089458E-22	1.322586E-05 ± 6.059815E-06	1.172055E-23 ± 8.290345E-24
F2	**3.589113E-02** ± **6.831873E-02**	2.840000E+00 ± 1.050000E+00	1.080000E+00 ± 1.180000E+00	2.890000E-01 ± 7.360000E-02	3.100000E-01 ± 9.950000E-01	2.236918E+00 ± 6.497812E-01	2.930333E+00 ± 5.319324E-01	4.010175E+01 ± 1.808692E+02
F3	3.996803E-15 ± 7.450000E-17	1.780000E-11 ± 1.630000E-11	7.020000E-06 ± 5.770000E-06	**3.320000E-15** ± **8.860000E-16**	3.850000E-01 ± 2.070000E-01	1.208050E-11 ± 3.700179E-12	2.171306E-03 ± 5.953712E-04	1.509755E-12 ± 1.355719E-12
F4	**1.346803E-17** ± **1.450000E-18**	2.350000E-09 ± 2.720000E-09	3.830000E-10 ± 7.360000E-10	9.280000E+00 ± 2.970000E+00	5.110000E+00 ± 2.290000E+00	7.417245E+00 ± 2.500943E+00	2.877694E+01 ± 5.177916E+00	2.463242E+00 ± 1.537768E+00
F5	1.975511E-02 ± 1.436385E-02	**4.560000E-03** ± **4.810000E-03**	4.070000E-02 ± 2.800000E-02	1.040000E-01 ± 7.100000E-02	4.780000E-02 ± 2.880000E-02	1.863168E-02 ± 2.186083E-02	4.502755E-01 ± 7.315302E-02	8.280319E-02 ± 3.043519E-02
F6	**4.231256E-01** ± **1.978116E-01**	6.440000E+00 ± 4.560000E+00	3.010000E+00 ± 2.520000E+00	1.110000E+00 ± 1.180000E+00	5.730000E-01 ± 1.050000E+00	1.590809E+00 ± 3.474197E-01	2.450773E+00 ± 5.887838E-01	2.126210E+01 ± 3.156376E+01
F7	**3.996803E-15** ± **1.996803E-15**	8.450000E-08 ± 2.970000E-07	1.310000E+00 ± 9.680000E-01	7.700000E-02 ± 2.880000E-01	2.030000E+01 ± 8.540000E-02	1.191482E-11 ± 3.069150E-12	2.670355E-03 ± 6.578756E-04	4.927146E-01 ± 8.201009E-01
F8	**2.118744E-31** ± **3.012089E-31**	8.850000E+00 ± 3.070000E+00	2.810000E+01 ± 1.390000E+01	1.180000E+01 ± 5.660000E+00	1.260000E+01 ± 7.000000E+00	6.278032E+00 ± 3.200551E+00	3.502026E+01 ± 3.866962E+00	1.621780E+01 ± 6.989113E+00
F9	2.263763E-02 ± 1.305243E-02	4.500000E-02 ± 3.080000E-02	1.200000E-01 ± 8.070000E-02	7.760000E-02 ± 6.400000E-02	1.170000E-01 ± 7.400000E-02	**1.505576E-02** ± **1.649220E-02**	4.975656E-01 ± 5.881158E-02	2.292702E-01 ± 1.626881E-01

4.3.2 Results on the 30-D problems

In this section, we present the results of the three groups of algorithms (Group A, B, and C) on the 30-dimensional benchmark functions. For all experiments, a maximum of 100,000 function evaluations (FEs) was used, and each algorithm was independently run 30 times to ensure statistical reliability. The performance of each algorithm is evaluated in terms of the mean, and standard deviation of the objective values obtained over the 30 runs. The results of the algorithms on the 30-D benchmark functions are summarized in Table 10.

In our experiments on the 30-dimensional benchmark functions, PSA-PSO achieved the best performance on 3 out of 9 functions, ranked second on 4 out of 9 functions, and third on 2 out of 9 functions. The results of the Friedman test further confirm the efficiency of PSA-PSO over other compared algorithms. The average ranks obtained from the Friedman test are presented in Table9.

Table 9 Friedman Test Results on 30-D Problems (Average Ranks, D=30)

Algorithm	Average Rank
PSA-PSO	**1.89**
STD PSO 2011	2.56
CLPSO	3.67
UPSO	4.33
CPSO	4.89
Std. PSO2007	5.11
APSO	6.22
DE	7.33

Table 10 Results on the 30-D problems

Function	PSA - PSO	CLPSO	CPSO	UPSO	Std. PSO2007	STD PSO 2011	DE	APSO
f1	1.904841E-22 ± 1.394758E-22	2.420000E-21 ± 2.490000E-21	1.570000E-10 ± 2.990000E-10	1.410000E-10 ± 2.380000E-09	2.410000E-10 ± 1.480000E-09	9.403072E-15 ± 3.263683E-15	9.045199E-06 ± 4.239498E-06	**9.805693E-24 ± 1.608405E-23**
F2	**2.138083E-02 ± 2.972990E-02**	2.010000E+01 ± 3.370000E+00	1.030000E+01 ± 1.370000E+01	1.310000E+01 ± 2.980000E+00	1.230000E+01 ± 1.470000E+00	6.964345E+01 ± 4.497944E+01	1.140418E+10 ± 4.796185E+09	1.595346E+04 ± 3.145017E+04
F3	3.647305E-12 ± 1.994534E-12	**9.900000E-14 ± 3.800000E-14**	3.900000E-06 ± 5.980000E-06	1.330000E+00 ± 8.580000E-01	1.360000E+00 ± 1.130000E+00	1.810121E-05 ± 3.082619E-05	7.082576E+00 ± 4.564420E-01	3.542315E-02 ± 1.695650E-01
F4	**6.821210E-14 ± 5.870763E-14**	1.870000E-09 ± 5.340000E-09	3.150000E-10 ± 1.050000E-09	7.630000E+01 ± 1.450000E+01	4.880000E+01 ± 1.440000E+01	8.955241E+00 ± 2.744369E+01	2.389961E+02 ± 1.471031E+01	8.350649E+01 ± 2.203542E+01
F5	2.217336E-03 ± 4.710342E-03	**3.140000E-10 ± 4.640000E-10**	3.630000E-02 ± 3.600000E-02	1.660000E-03 ± 3.070000E-03	1.830000E-02 ± 2.200000E-02	4.769498E-03 ± 5.676690E-03	5.634902E+00 ± 1.356271E+00	7.414576E-03 ± 2.203542E+01
F6	2.272669E+01 ± 5.227555E-01	3.040000E+01 ± 1.320000E+01	3.030000E+01 ± 8.300000E-01	**2.150000E+01 ± 1.890000E+01**	2.770000E+01 ± 1.310000E+01	2.701624E+01 ± 4.877317E-01	9.195368E+01 ± 2.110315E+01	7.657490E+01 ± 1.390118E+02
F7	**3.222045E-12 ± 1.680285E-12**	2.220000E-06 ± 8.290000E-06	1.520000E+00 ± 8.300000E-01	1.630000E+00 ± 9.490000E-01	2.090000E+00 ± 1.310000E+00	2.645193E-08 ± 9.621469E-09	7.004070E+00 ± 5.353673E-01	1.546661E+00 ± 9.950837E-01
F8	**9.663381E-14 ± 1.631053E-13**	4.700000E+01 ± 6.850000E+00	9.520000E+01 ± 2.930000E+01	7.700000E+01 ± 1.700000E+01	9.040000E+01 ± 4.030000E+01	8.518430E+01 ± 1.545875E+01	2.473329E+02 ± 1.176240E+01	1.254878E+02 ± 4.633676E+01
F9	1.971457E-03 ± 4.156196E-03	**7.040000E-10 ± 1.250000E-11**	5.540000E-02 ± 3.970000E-02	1.480000E-03 ± 3.120000E-03	1.510000E-02 ± 1.240000E-02	1.971222E-03 ± 4.314457E-03	5.214408E+00 ± 7.929083E-01	8.624827E-03 ± 6.386197E-03

4.3.3 Comparison with Standard PSO

In this section, we compare the performance of PSA-PSO on functions from Group B (multi-modal) and Group D (shifted multi-modal) with the standard PSO algorithms, namely Std. PSO 2007 and Std. PSO 2011.

The parameters of the test functions used in this comparison are listed in Table 11.

Table 11 Parameters of Test Functions Used in the Comparison

Function	Search Space	Dimension	Acceptable Error	Max. FEs
Rosenbrock	$[-10, 10]$	30	0.0001	40000
Ackley	$[-32, 32]$	30	0.0001	40000
Rastrigin	$[-10, 10]$	30	0.0001	40000
Griewank	$[-100, 100]$	30	0.0001	40000
Shifted Rosenbrock	$[-100, 100]$	10	0.01	100000
Shifted Ackley	$[-32, 32]$	30	0.0001	100000
Shifted Rastrigin	$[-5, 5]$	30	0.0001	100000
Shifted Griewank	$[-600, 600]$	30	0.0001	100000
Shifted Sphere	$[-100, 100]$	30	0.0001	100000

As per the results presented in Table 12 PSA-PSO demonstrates superior performance compared to Std. PSO 2007 and Std. PSO 2011 across most of the benchmark functions from Group B (multi-modal) and Group D (shifted multi-modal).

Table 12 Comparison of PSA-PSO with Std. PSO 2007 and Std. PSO 2011 on Group B and D Functions

		PSA - PSO			STD PSO 2007			STD PSO 2011		
Function	Dimension	Mean best	Min. error	Suc. Rate	Mean best	Min. error	Suc. Rate	Mean best	Min. error	Suc. Rate
F2 Rosenbrock	30	**2.14E-02**	**1.62E-05**	**20%**	2.28E+01	2.14E+01	0%	6.96E+01	4.91E-01	0%
F3 Ackley	30	**3.65E-12**	**1.68E-12**	**100%**	9.94E-01	7.11E-01	33%	1.81E-05	5.25E-06	93%
F4 Rastrigin	30	**6.82E-14**	**0.00E+00**	**100%**	5.71E+01	2.49E+01	0%	8.96E+00	8.49E+01	0%
F5 Griewank's	30	**2.22E-03**	**0.00E+00**	**80%**	1.31E-02	4.90E-05	43%	4.77E-03	1.38E-06	53%
F 10 Shif ted Rosenbrock	10	**5.07E-01**	**3.50E-03**	**70%**	2.81E+04	9.65E-03	67%	1.45E+01	1.05E+00	0%
F 11Shif ted Ackley	30	2.30E-05	9.95E-13	100%	1.18E+02	1.18E+02	0%	**3.20E-06**	**1.20E-06**	**100%**
F12 Shif ted Rastrigin	30	**6.35E+01**	**5.47E+01**	**0%**	5.44E+01	2.92E+01	0%	7.14E+01	2.30E+01	0%
F13 Shif ted Griewank	30	3.70E-03	3.70E-03	100%	4.77E-05	4.93E-07	100%	**3.00E-06**	**3.20E-07**	**100%**
F 14 Shif ted Sphere	30	**2.30E-06**	**3.00E-07**	**100%**	5.87E-05	6.55E-06	100%	3.20E-04	3.00E-05	100%

4.3.4 Results on Lennard-Jones Cluster Optimization

In this section, we present the results of the minimization of the Lennard-Jones (LJ) potential for clusters containing 8, 9, and 10 atoms. The performance of PSA-PSO is compared with the standard PSO algorithms, Std. PSO 2007 and Std. PSO 2011, to evaluate its effectiveness in finding the global minimum energy configurations of these small atomic clusters.

As shown in Table 13, PSA-PSO outperforms both Std. PSO 2007 and Std. PSO 2011 in minimizing the Lennard-Jones potential for clusters of 8, 9, and 10 atoms (corresponding to dimensions 24, 27, and 30, respectively).

Table 13 Comparison of PSA-PSO with Std. PSO 2007 and Std. PSO 2011 on Lennard-Jones Clusters (8–10 Atoms)

		PSA - PSO			STD PSO 2007			STD PSO 2011		
Function	Dimension	Mean best	Min. error	Suc. Rate	Mean best	Min. error	Suc. Rate	Mean best	Min. error	Suc. Rate
Lennard Jones	24	**7.528990E-01 ± 7.221058E-01**	**1.00E-06**	**40%**	1.170000E+00 ± 1.800000E+00	4.80E-05	34%	6.033029E+00 ± 9.447478E-01	4.55E+00	0%
Lennard Jones	27	**1.278720E+00 ± 9.970934E-01**	**0.00E+00**	**20%**	2.570000E+00 ± 2.990000E+00	8.00E-05	14%	9.348850E+00 ± 6.642258E-01	8.73E+00	0%
Lennard Jones	30	**2.080812E+00 ± 2.159459E+00**	**1.20E-05**	**18%**	4.660000E+00 ± 4.220000E+00	9.20E-05	2%	1.233805E+01 ± 4.416605E-01	1.16E+01	0%

4.4 Results Discussion

In this section, we analyze and discuss the performance of PSA-PSO on various benchmark functions, comparing it to standard PSO algorithms and other state-of-the-art methods. The discussion is organized by problem dimension and function type.

4.4.1 10-D Results Discussion

For the 10-dimensional benchmark functions, including Group A (unimodal), Group B (multi-modal), and Group C (rotated multi-modal), PSA-PSO demonstrates superior performance across the majority of functions. Key observations include:

- **Mean Best Values:** PSA-PSO achieves the lowest mean values on most functions, indicating that it consistently converges closer to the global optimum compared to the other algorithms.
- **Standard Deviation:** The standard deviation values are generally lower than those of other methods, reflecting high stability and robustness of the algorithm across independent runs.
- **Statistical Validation:** PSA-PSO attains a Friedman test score of 1.44, the lowest among all compared algorithms, which confirms its statistical superiority in 10-D function minimization.
- **Overall Efficiency:** The combination of low mean, low standard deviation, and top Friedman ranking demonstrates that PSA-PSO is both accurate and reliable for low-dimensional optimization problems.

4.4.2 30-D Results Discussion

When applied to the higher-dimensional versions of the same functions (30-D), PSA-PSO maintains its strong performance. The main findings are:

- **Consistency in Performance:** PSA-PSO achieves the best mean and standard deviation on 3 out of 9 functions, demonstrating that it scales effectively to higher-dimensional search spaces.
- **Stability:** The algorithm retains low standard deviation values across most functions, indicating stable convergence even in the presence of increased complexity.
- **Friedman Test Results:** With an average rank of 1.89, PSA-PSO statistically outperforms the other algorithms, confirming its effectiveness and robustness on higher-dimensional problems.
- **Scalability:** These results suggest that the adaptive topology and surrogate-assisted strategies embedded in PSA-PSO enable efficient exploration and exploitation even as problem dimensionality increases.

4.4.3 Discussion of Results Compared to Standard PSO 2007 and 2011

In this study, we compared the performance of PSA-PSO with Std. PSO 2007 and Std. PSO 2011 on benchmark functions from Group B (multi-modal) and Group D

(shifted multi-modal). Table 12 summarizes the results of this comparison. Our analysis focused on three key metrics: mean best values, minimum error, and success rate, to evaluate both convergence accuracy and reliability across multiple independent runs.

For Group B functions, including F2 Rosenbrock, F3 Ackley, F4 Rastrigin, and F5 Griewank, PSA-PSO consistently achieved:

- **Lower mean best values:** indicating convergence closer to the global optimum compared to the standard PSO algorithms.
- **Higher success rates:** for example, on F3 Ackley, PSA-PSO reached a mean best of 3.65×10^{-12} with 100% success, whereas Std. PSO 2007 achieved only 9.94×10^{-1} with 33% success, and Std. PSO 2011 reached 1.81×10^{-5} with 93% success.
- **Robust performance across multiple runs:** PSA-PSO demonstrates low variability and high reliability in exploring complex multi-modal landscapes.

For Shifted Multi-Modal functions, including F10 Shifed Rosenbrock, F11 Shifed Ackley, F12 Shifed Rastrigin, F13 Shifed Griewank, and F14 Shifed Sphere, PSA-PSO either matched or exceeded the performance of standard PSO algorithms:

- **High convergence accuracy:** For F11 Shifed Ackley, PSA-PSO achieved a mean error of 2.30×10^{-5} with 100% success, while Std. PSO 2007 failed completely (0% success), and Std. PSO 2011 performed comparably.
- **Fast and reliable convergence:** On F10 Shifed Rosenbrock, PSA-PSO obtained a lower mean best value than Std. PSO 2007, despite a slightly higher success rate for the latter (67% vs. 70%).

Across all Group B and D functions, PSA-PSO demonstrates:

- Lower mean errors and higher success rates.
- Consistently robust performance across multiple independent runs.
- Superior ability to navigate both standard multi-modal and shifted complex landscapes.

These results collectively confirm the efficiency and effectiveness of PSA-PSO over standard PSO variants in both multi-modal and shifted multi-modal benchmark functions.

4.4.4 Lennard Jones Results Discussion

In this study, we compared the performance of PSA-PSO with Std. PSO 2007 and Std. PSO 2011 for minimizing the Lennard-Jones potential for small atomic clusters containing 8, 9, and 10 atoms (corresponding to dimensions 24, 27, and 30, respectively). The results, summarized in Table 13, demonstrate that PSA-PSO outperforms both standard PSO variants across all evaluated metrics.

- **Mean Best Values:** PSA-PSO consistently achieves the lowest mean best energies across all cluster sizes, indicating more effective convergence toward the global minimum. For instance, for the 30-atom cluster, PSA-PSO reached a mean best of 2.08 with a standard deviation of 2.16, while Std. PSO 2007 and Std. PSO 2011 obtained 4.66 and 12.34, respectively.

- **Minimum Error:** PSA-PSO achieves very low minimum errors, demonstrating high precision in locating near-optimal configurations. For the 27-atom cluster, the minimum error is 0.00 for PSA-PSO, compared to 8.00×10^{-5} for Std. PSO 2007 and 8.73 for Std. PSO 2011.
- **Success Rate:** PSA-PSO exhibits higher success rates than the standard PSO variants. For example, for the 24-atom cluster, PSA-PSO achieved a 40% success rate, compared to 34% for Std. PSO 2007 and 0% for Std. PSO 2011.

Overall, these results indicate that PSA-PSO is both more efficient and robust in finding low-energy configurations for small Lennard-Jones clusters. Its superior performance across mean best values, minimum error, and success rates demonstrates the algorithm's effectiveness over the standard PSO algorithms for these challenging optimization problems.

5 Conclusion

In this work, we proposed a variant of Particle Swarm Optimization (PSO) enhanced with a surrogate-assisted deep learning neural network, designed to improve optimization efficiency and stability. The algorithm, PSA-PSO , was rigorously tested on a variety of benchmark functions, including unimodal, multi-modal, rotated multi-modal, shifted multi-modal, and Lennard-Jones cluster problems. The experimental results demonstrate that PSA-PSO consistently outperforms standard PSO algorithms in terms of mean best values, convergence stability, minimum error, and success rate. Statistical validation via the Friedman test further confirms its effectiveness across low- and high-dimensional optimization tasks.

As a future direction, we plan to evaluate the performance of PSA-PSO on real-world application problems, to further assess its practical efficiency and robustness in complex optimization scenarios.

References

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proceedings of IEEE International Conference on Neural Networks*, pp. 1942–1948, 1995.

[2] C. Sun, Y. Jin, J. Zeng, and Y. Yu, "A two-layer surrogate-assisted particle swarm optimization algorithm," *Soft Computing*, vol. 19, no. 6, pp. 1461–1475, 2015.

[3] Z. Yang, H. Qiu, and L. Gao, "A surrogate-assisted particle swarm optimization algorithm based on efficient global optimization for expensive black-box problems," *Engineering Optimization*, vol. 51, no. 4, pp. 568–588, 2019.

[4] F. Li, X. Cai, L. Gao, and W. Shen, "A surrogate-assisted multiswarm optimization algorithm for high-dimensional computationally expensive problems," *IEEE Transactions on Cybernetics*, vol. 51, no. 3, pp. 1390–1402, 2021.

[5] S. Hemayat *et al.*, "Integrating deep convolutional surrogate solvers and particle swarm optimization for inverse design," *arXiv preprint*, 2024, arXiv:PMC11501072. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC11501072/

[6] B. A. Khuwaileh, M. Al-Shabi, and M. E. H. Assad, "Artificial neural network based particle swarm optimization solution approach for the inverse depletion of used nuclear fuel," *Annals of Nuclear Energy*, 2021.

[7] A. E. Dor, M. Clerc, and P. Siarry, "A multi-swarm pso using charged particles in a partitioned search space for continuous optimization," *Computational Optimization and Applications*, vol. 53, no. 1, pp. 271–295, 2012.

[8] X. Wu, J. Han, D. Wang, P. Gao, Q. Cui, L. Chen, Y. Liang, H. Huang, H. P. Lee, C. Miao, Y. Zhou, and C. Wu, "Incorporating surprisingly popular algorithm and euclidean distance-based adaptive topology into pso," *Swarm and Evolutionary Computation*, vol. 76, p. 101222, 2023.

[9] X. Liang, W. Li, Y. Zhang, and M. C. Zhou, "An adaptive particle swarm optimization method based on clustering," *Soft Computing*, vol. 19, no. 2, pp. 431–448, 2014.

[10] R. Salomon, "Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions," *BioSystems*, vol. 39, pp. 263–278, 1996.

[11] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*. IEEE, 2005, pp. 1–12.

[12] J. P. K. Doye and D. J. Wales, "Theoretical studies of lennard-jones clusters: Structure and optimization," *Journal of Chemical Physics*, vol. 109, pp. 8143–8153, 1998.