# Particle Swarm Optimization (PSO)

## 1. Introduction

Particle Swarm Optimization (PSO) is a **swarm intelligence** method inspired by collective behaviors observed in nature, such as flocks of birds or schools of fish. It is widely applied in various optimization problems, including **course scheduling, function optimization, and machine learning hyperparameter tuning**. The algorithm operates by maintaining a population (or swarm) of candidate solutions, known as particles, that iteratively adjust their positions in search of an optimal solution.

This study implements two key variants of PSO:

1. **Global Best PSO (Gbest PSO):**
    - In this approach, each particle's movement is influenced by **the globally best solution** found within the swarm.
    - This structure allows **rapid convergence**, as particles are collectively guided toward the most promising solution.
    - However, it may increase the risk of **premature convergence**, where particles become trapped in suboptimal solutions due to lack of diversity.
2. **Local Best PSO (Lbest PSO):**
    - Instead of relying on a single global best, **each particle interacts only with its local neighbors**, forming dynamic neighborhoods.
    - This decentralized learning promotes **diversity** in the search process, reducing the likelihood of stagnation in suboptimal solutions.
    - The localized structure enables **gradual refinement** of solutions, improving robustness while maintaining adaptability.

## 2. Algorithm Design

### 2.1. Particle Representation

Each particle in the swarm represents a **possible scheduling solution**, structured as:

- **Course ID, Instructor ID, Room Assignment, Timeslot**
- This ensures each candidate solution is a **valid representation** within the scheduling constraints.

### 2.2. Swarm Initialization

- The swarm consists of **S particles**, each initialized with **random room and timeslot assignments**.

- Each particle maintains a velocity vector, which dictates **how it moves** in the scheduling space.

## 2.3. Neighborhood Formation

Unlike Global Best PSO (Gbest PSO), where all particles are influenced by the **global best**, this implementation:

- **Divides the swarm into N neighborhoods** (groups of particles).
- **Each particle selects k nearest neighbors** based on Manhattan distance (i.e., similarity of room and timeslot scheduling).
- The particle follows **the best neighbor**, rather than a global best solution.

## 2.4. Velocity Update Mechanism

Each particle updates its velocity using the equation:

$$v_i(t + 1) = \omega\, v_i(t) + c_1\, r_1(pbest - x_i) + c_2 r_2(nbest - x_i)\, v_i(t + 1)$$

**Where:**

- $v_i(t + 1)$ is the **updated velocity** of particle i at iteration (t + 1).
- $v_i(t)$ is the **current velocity** of particle i.
- $x_i$ is the **current position** of particle i.
- ω (inertia weight) controls how much of the previous velocity is retained.
- $c_1$ (cognitive coefficient) influences the particle's movement toward its own best solution.
- $c_2$(social coefficient) guides the particle toward the **best solution among its k-nearest neighbors**.
- $r_1$, $r_2$ are random values between [0,1] to introduce stochastic behavior.
- $pbest$ is the **personal best position** of the particle.
- $nbest$ is the **best position found by its local neighbors**.

## 2.5. Position Update

Once velocity is updated, the new **position** of the particle is computed as:

$$x_i(t + 1) = x_i(t) + v_i(t + 1)$$

This ensures the particle moves based on the learned velocity while exploring new solutions.

### 2.6. Adaptive Neighborhoods

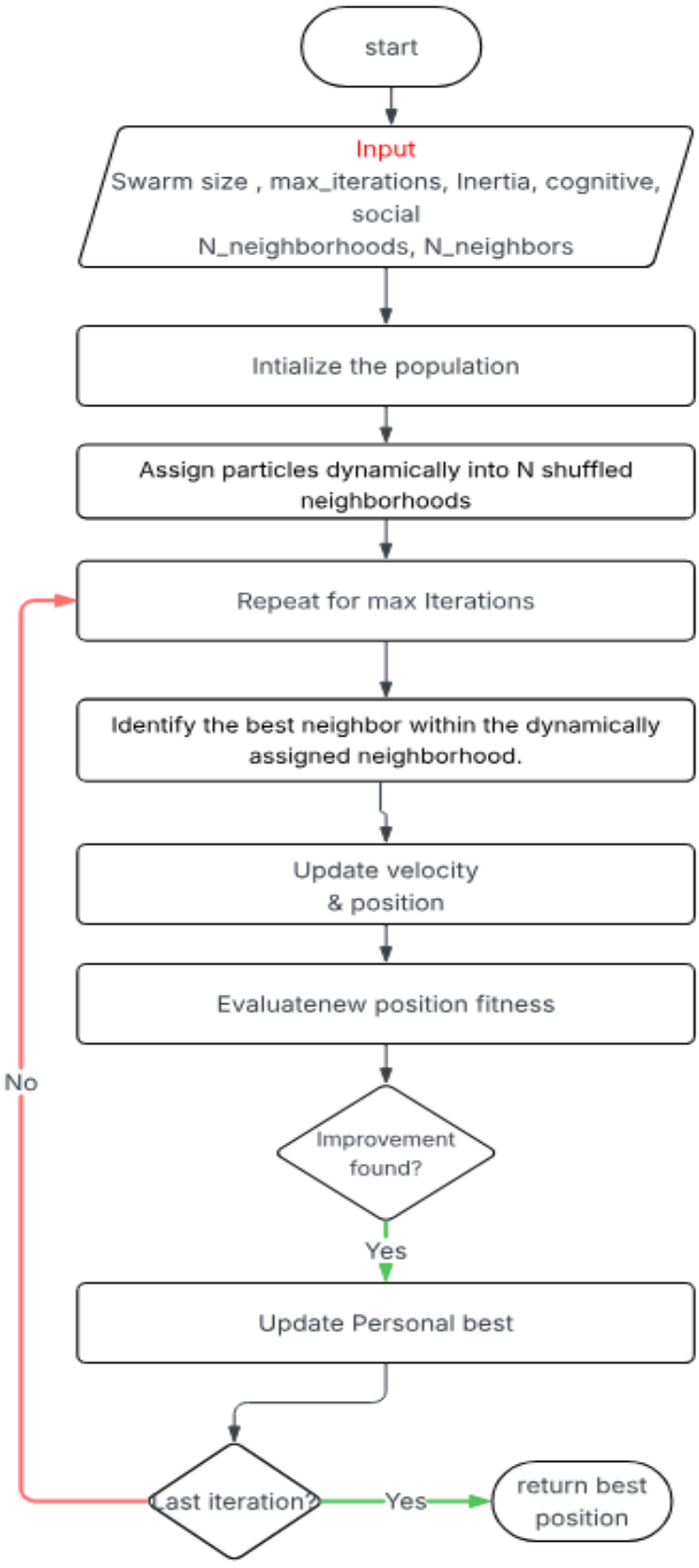
Neighborhoods are shuffled dynamically after initialization to **prevent stagnation** in local optima.

## 2.7. Algorithm parameters setup

| **Subset of Data (20%)** | |
|---|---|
| PSO Swarm size | 100 Particles |
| PSO Iterations Number | 1000 Iterations |
| Inertial weight | 0.5 |
| Cognitive Learning Rate | 1 |
| Social Learning Rate | 1 |
| Number of Neighborhoods | 10 |
| Neighbors per particle | 7 |

| **All Data** | |
|---|---|
| PSO Swarm size | 250 Particles |
| PSO Iterations Number | 1000 Iterations |
| Inertial weight | 0.5 |
| Cognitive Learning Rate | 2 |
| Social Learning Rate | 2 |
| Number of Neighborhoods | 10 |
| Neighbors per particle | 7 |

## 3) Lbest PSO Approach Flowchart



## 4) Lbest PSO Approach Pseudocode

**Algorithm: Local Best Particle Swarm Optimization (lbest PSO)**

**Input:**

- **S** = swarm size (number of particles)
- **max_iterations** = number of optimization cycles
- **N** = number of neighborhoods (subdivisions of swarm)
- **k** = number of neighbors per neighborhood (local best selection)
- **inertia, cognitive, social** = PSO hyperparameters controlling movement behavior

**Initialization:**

1. **Create a swarm of S particles**, each representing a potential scheduling solution.
2. Assign each particle a **random position** (course assignments to rooms and timeslots).
3. Assign each particle a **random velocity**, affecting room and timeslot adjustments.
4. Set each particle's **best personal position** to its initial state.
5. Compute **fitness** for each particle and update the best-known global solution.
6. Assign particles dynamically into **N shuffled neighborhoods** to ensure varied local interactions.

**Optimization Process (Repeat for max_iterations):**

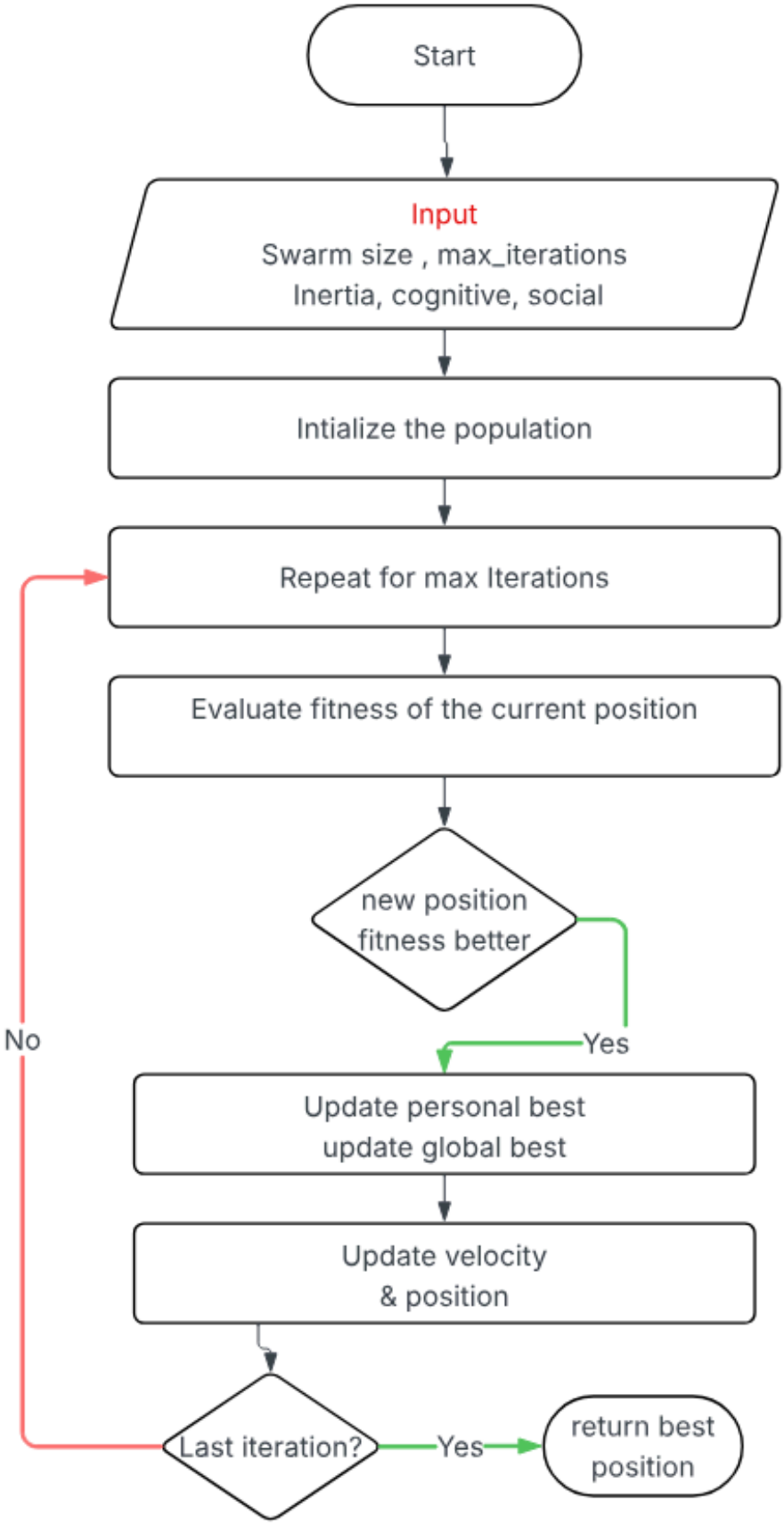
For each particle in the swarm:

1. **Identify the best neighbor** within the dynamically assigned neighborhood.
2. **Update velocity** using:
    - **Inertia component** (retains some influence from previous velocity).
    - **Cognitive component** (adjusts towards the particle's personal best).
    - **Social component** (adjusts towards the best-performing neighbor).
3. **Update position** based on new velocity, ensuring valid constraints (room availability, timeslot restrictions).
4. **Evaluate new position's fitness** and update personal best if an improvement is found.
5. Track **global best solution** across all particles while neighborhoods continue evolving.

**Return:**

- The **best scheduling solution** found during optimization.

## 5) Gbest PSO Approach Flowchart



## 6) Gbest PSO Approach Pseudocode

**Algorithm: Global Best Particle Swarm Optimization (gbest PSO)**

**Input:**

- **S** = swarm size (number of particles)
- **max_iterations** = number of optimization cycles
- **inertia, cognitive, social** = PSO hyperparameters controlling movement behavior

**Initialization:**

1. **Create a swarm of S particles**, each representing a potential scheduling solution.
2. Assign each particle a **random position** (course assignments to rooms and timeslots).
3. Assign each particle a **random velocity**, affecting room and timeslot adjustments.
4. Set each particle's **best personal position** to its initial state.
5. Compute **fitness** for each particle using a predefined fitness function.
6. Identify the best-performing particle and set it as the **global best solution**.

**Optimization Process (Repeat for max_iterations):**

For each particle in the swarm:

1. **Evaluate fitness** to determine how good the current schedule is.
2. **Update personal best** if the new position provides a better fitness score.
3. **Update global best** if the particle's personal best is better than the current global best.
4. **Update velocity** using:
    - **Inertia component** (retains some influence from previous velocity).
    - **Cognitive component** (adjusts towards the particle's personal best).
    - **Social component** (adjusts towards the global best solution).
5. **Update position** based on new velocity, ensuring constraints (room availability, valid timeslots).
6. Repeat the process across all particles until convergence or reaching **max_iterations**.

**Return:**

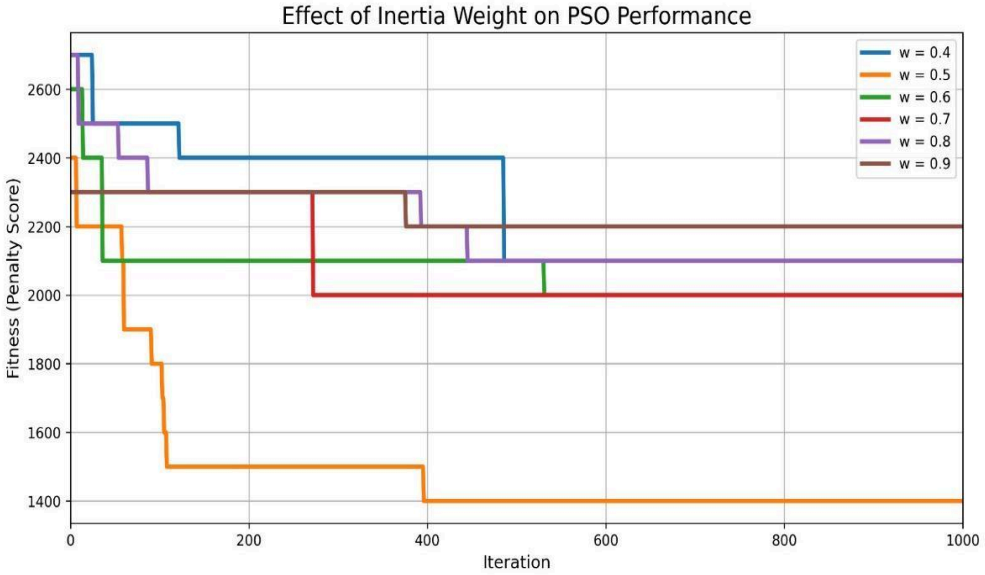
- The **best scheduling solution** found during optimization.

# 7. Experimental Setup & Results

The results you obtained are based on only 20% of the available data. This limitation occurred due to constraints in computational power and the excessively long processing time required to analyze the full dataset. As a result, the findings might not fully represent the entire dataset and could be impacted by the reduced sample size.

## 3.1. Effect of Inertia weight on PSO Performance

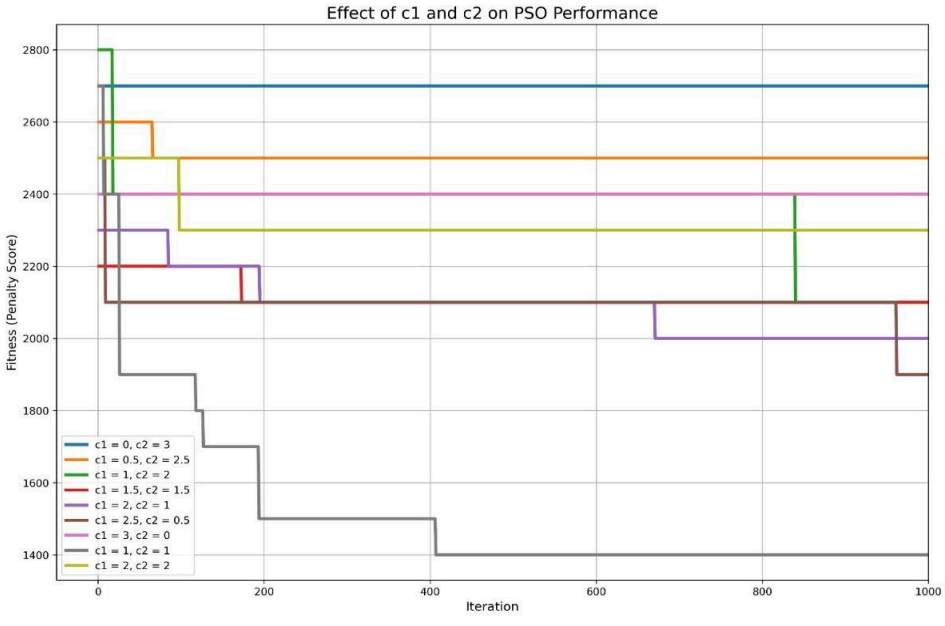
The inertia weight (ω) plays a significant role in determining the performance of Particle Swarm Optimization (PSO) fitness, though its behavior is not entirely fixed or predictable. However, a general trend emerges: fitness improvements tend to increase as ω decreases.




In our experiment, most ww values exhibited early stabilization or convergence, typically reaching that state around the halfway mark of the maximum iterations. Based on the results, setting ww around **0.5** appears to offer an optimal balance. This value demonstrated the best overall performance—both in terms of final fitness score and the level of improvement observed throughout the iterations.

## 3.2. Effect of C1 and C2 combinations on PSO Performance

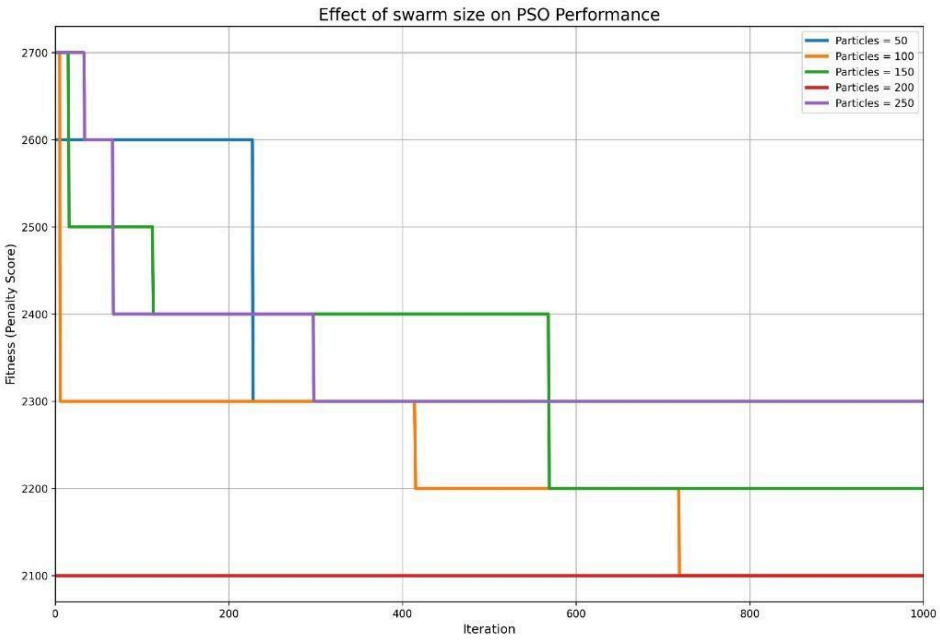
The impact of varying the coefficients $c_1$ and $c_2$ on fitness performance in Particle Swarm Optimization (PSO) appears to be relatively minor, as most parameter combinations exhibit similar effects. However, assigning a value of **3** to either $c_1$ or $c_2$ tends to degrade performance, with fitness values remaining constant across iterations, indicating stagnation.



Conversely, parameter combinations where **one of the values is set to 1** result in noticeably improved fitness performance. These configurations demonstrate both higher fitness improvements and better final scores across generations. Based on the experimental results, setting $c_1 = 1$ and $c_2 = 1$ emerges as the most effective configuration, yielding the best overall performance in terms of both convergence behavior and final fitness values.

### 3.3. Effect of Swarm Size on PSO Performance

Among the different swarm sizes tested, **100 particles consistently provided the best optimization**, demonstrating strong initial fitness and efficient convergence to a competitive final score. The **150 and 250 particle configurations** also showed steady improvement throughout the iterations, maintaining consistent progress before stabilizing.




On the other hand, the **50-particle swarm reached convergence early**, with only minimal improvement, leading to a higher penalty score. The **200-particle swarm exhibited the weakest performance**, showing limited optimization and stagnation at a higher fitness value.

For this particular experiment, a **swarm size of 100 particles** emerged as the most effective configuration, offering the best balance between performance and convergence behavior. **Swarm sizes of 150 and 250 also performed reasonably well**, whereas **50 and 200 particles showed weaker results due to early convergence or stagnation**.

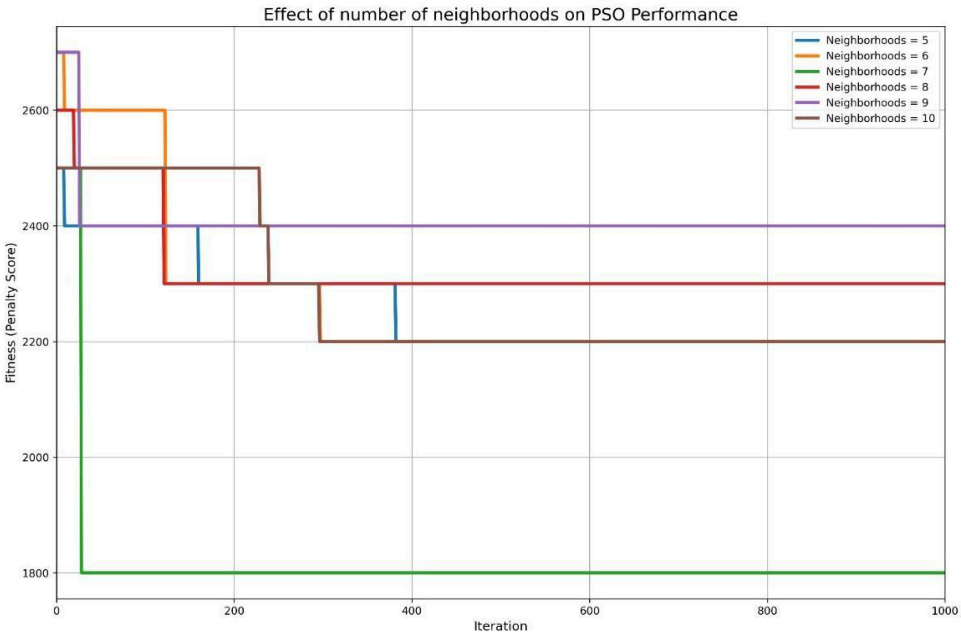
### 3.4. Effect of Number of neighborhood on PSO Performance

All neighborhood configurations began with high fitness scores and exhibited a consistent decrease in penalty throughout the generations. However, **most configurations experienced early convergence**, suggesting limited exploration in the later stages of the optimization process.

**Neighborhood Size = 7:** Achieved the **fastest convergence** while maintaining the **best fitness score**, demonstrating a strong balance between **exploration** (global search) and **exploitation** (local refinement).

**Neighborhood Size = 10:** Showed the **most substantial fitness improvement before convergence**, suggesting **better initial exploration** but **weaker exploitation** toward the final solution.

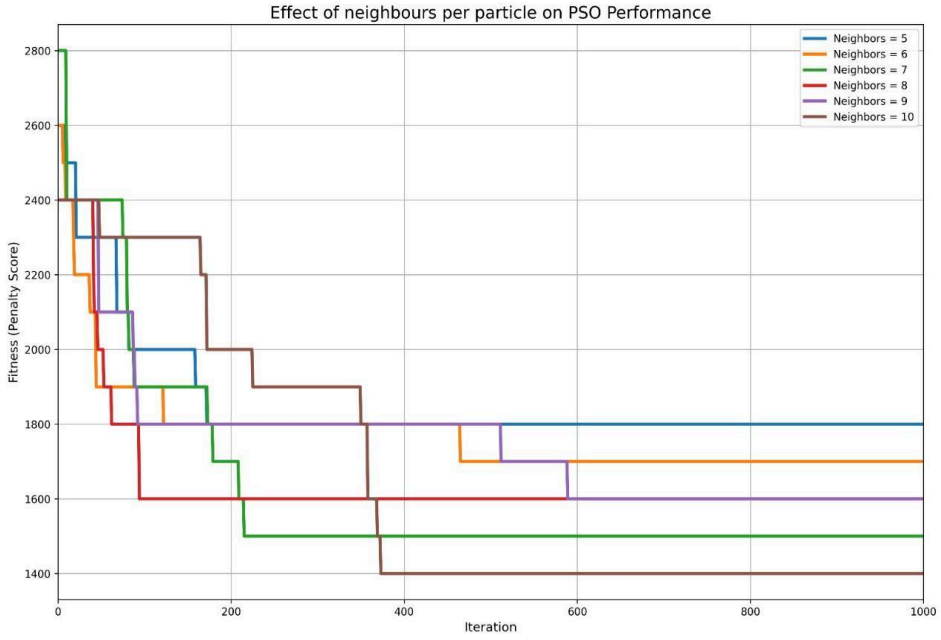
**Neighborhood Sizes = 5, 6, 8, and 9:** Converged relatively early without reaching particularly low fitness values. These configurations appeared to stagnate after a short period of improvement, indicating **either weak exploration or premature convergence**.



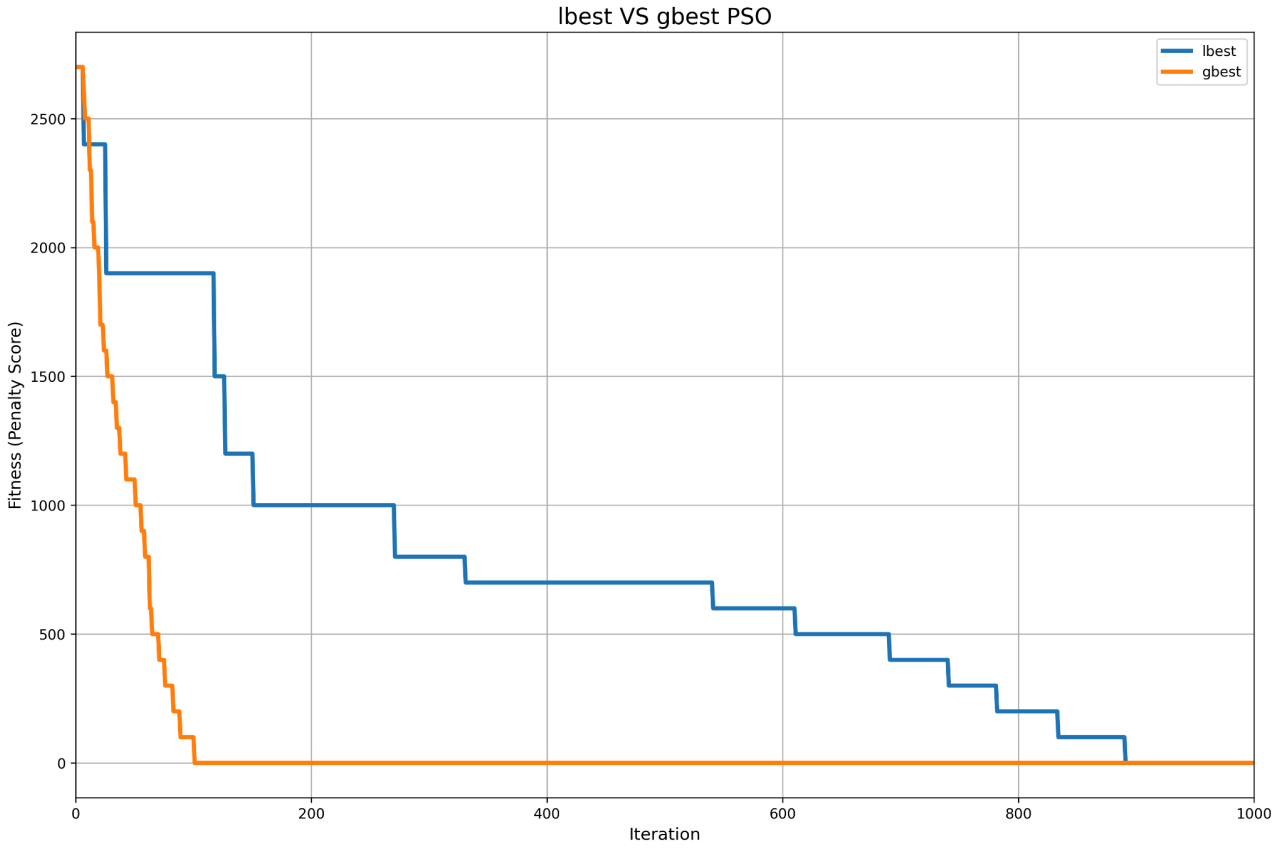

Among the tested configurations, a **neighborhood size of 7** provided the best trade-off between exploration and exploitation, resulting in **efficient optimization**. Meanwhile, a **neighborhood size of 10** exhibited strong initial improvements but weaker refinement in later stages. Other configurations displayed premature convergence, limiting their overall effectiveness.

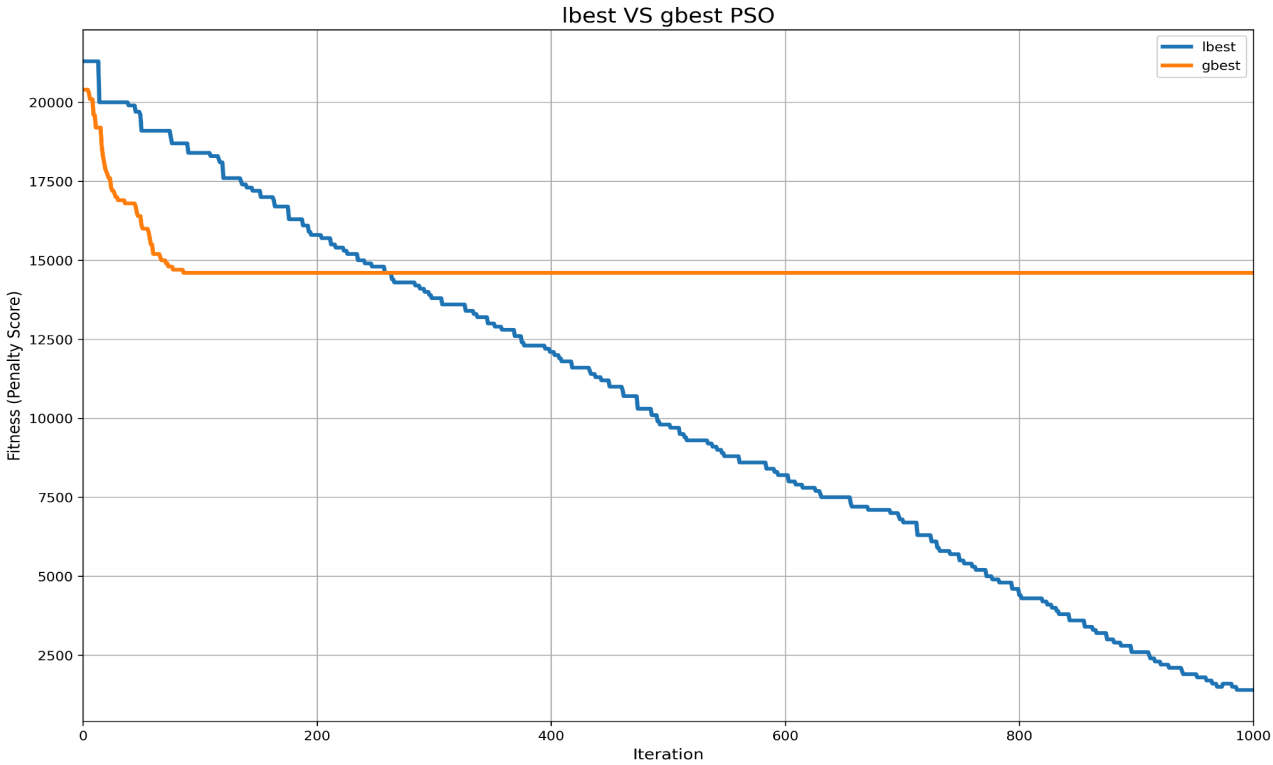
## 3.5. Effect of neighbors per particle on PSO Performance

The number of neighbors per particle influences the balance between exploration and exploitation in PSO. Configurations with **5, 6, 8, and 9 neighbors** exhibited early convergence but stagnated at higher fitness values, suggesting limited exploration. **A neighbors size of 7** achieved the fastest convergence and the lowest fitness score, demonstrating an optimal balance between search diversity and refinement. In contrast, **10 neighbors showed prolonged improvement before settling**, indicating strong initial exploration but weaker exploitation. A **neighborhood size of 7** provided the most efficient optimization, offering a trade-off between global search and local refinement. Smaller and larger neighborhood configurations either stagnated prematurely or took longer to converge, affecting overall performance.



## 3.5. Gbest VS Lbest approaches



Particle Swarm Optimization (PSO) is known for its ability to balance exploration and exploitation in search spaces. This study compares **Global Best PSO (Gbest PSO)** and **Local Best PSO (Lbest PSO)** based on their optimization behavior, analyzing how each approach influences convergence, search diversity, and final fitness values.

# 8. Observations from the Results

## 8.1. Global Best PSO (Gbest PSO)

- **Initial Performance:** Gbest PSO starts with **high fitness values**, indicating strong global guidance toward optimal solutions.
- **Convergence Behavior:** The rapid improvement in early iterations suggests **efficient exploitation**, allowing particles to quickly refine solutions.
- **Stagnation Risks:** While Gbest PSO achieves significant gains early, its reliance on a global best leader can lead to **premature convergence**, limiting exploration diversity.
- **Final Fitness Values:** The final optimization reaches competitive fitness levels, but the steady decline suggests that the swarm loses some adaptability after early improvements.

## 8.2. Local Best PSO (Lbest PSO)

- **Initial Performance:** Lbest PSO exhibits **gradual fitness improvement** over a longer period, emphasizing **localized learning** rather than immediate global influence.
- **Convergence Behavior:** The extended improvement phases suggest **better exploration**, allowing particles to search effectively before settling.
- **Adaptability and Diversity:** Unlike Gbest PSO, Lbest PSO maintains **higher variance in fitness values throughout iterations**, promoting flexibility in solution refinement.
- **Final Fitness Values:** The final solutions demonstrate a **smoother convergence curve**, indicating stronger adaptability across diverse search areas.

## 8.3 Behavioral Analysis

- **Exploration vs. Exploitation:** Gbest PSO prioritizes exploitation early, leading to faster but sometimes restrictive convergence. Lbest PSO, in contrast, ensures **wider search coverage** due to decentralized learning, reducing the risk of stagnation.
- **Search Stability:** Lbest PSO maintains adaptability, benefiting complex search spaces where premature convergence is undesirable.
- **Optimization Efficiency:** Gbest PSO is **ideal for problems requiring rapid convergence**, while Lbest PSO is **better suited for scenarios needing a broader search before refinement**.