EC Assignment #2 (PSO)

電機系(碩一)陳昕佑 61375017H

My assigned problems are f2(Sschwefel’s P2.22) in unimodal and f8(Schwefel function with dimension d=30) in multimodal. To make a comparison with GA, the key differences in PSO are:

1. **Particles Instead of Individuals:** Each particle has a position and velocity in the search space.
2. **Velocity Update:** Particle velocity is updated based on its best-known position and the global best position in the swarm.
3. **Position Update:** Particles move by updating their position based on their velocity.

For the PSO code implementation, the first thing to do is initialize particle positions and velocities:

```
class Particle:
    def __init__(self, dim, lower_bound, upper_bound):
        self.position = np.array([random.uniform(lower_bound, upper_bound)
                                  for _ in range(dim)])
        self.velocity = np.array([random.uniform(-1, 1) for _ in range(dim)])
        self.best_position = np.copy(self.position)
        self.best_fitness = float('inf')

    def update_velocity(self, global_best_position, inertia, cognitive_coeff, social_coeff):
       cognitive = cognitive_coeff * random.random() * (self.best_position - self.position)
        social = social_coeff * random.random() * (global_best_position - self.position)
       self.velocity = inertia * self.velocity + cognitive + social

    def update_position(self, lower_bound, upper_bound):
          self.position += self.velocity
          self.position = np.clip(self.position, lower_bound, upper_bound)
```

The Particle class defines initial positions randomly within the bounds and set initial velocities. The “update_velocity” method updates the particle’s velocity using the PSO formula based on inertia, cognitive, and social factos. The “update_position” method updates the particle’s position by adding the velocity and ensures it stays within bounds.

In PSO function, it would follow the steps bellow:

- There are three parameters need to adjust, inertia, cognitive and social coefficients.

  ```
  inertia = 0.7
  cognitive_coeff = 1.5
  social_coeff = 1.5
  ```

- Initializes a swarm of particles.

  ```
  swarm = [Particle(dim, lower_bound, upper_bound)
                   for _ in range(num_particles)]
  global_best_position = np.zeros(dim)
  global_best_fitness = float('inf')
  fitness_history = [ ]
  total_evaluations = 0
  ```

- loops through the ‘max_iterations’ or until ‘evaluation_limit’ is reached.

  ```
  for iteration in range(max_iterations):
    for particle in swarm:
      fitness = fitness_function(particle.position)
      total_evaluations += 1
  ```

- Updates each particle’s fitness, personal best, and global best.

  ```
   if fitness < particle.best_fitness:
       particle.best_fitness = fitness
       particle.best_position = np.copy(particle.position)
   if fitness < global_best_fitness:
       global_best_fitness = fitness
       global_best_position = np.copy(particle.position)
   if total_evaluations >= evaluation_limit:
       break
   fitness_history.append(global_best_fitness)
  ```

- Updates the particle velocities and positions based on global and personal bests.

  ```
  for particle in swarm:
    particle.update_velocity(global_best_position, inertia,
                                                  cognitive_coeff, social_coeff)
    particle.update_position(lower_bound, upper_bound)
  ```

- Records ‘global_best_fitness’ at each iteration for plotting.

For main execution:

- Runs PSO on both Schwefel functions (unimodal and multimodal) with the specified dimensionality, particle count, and iteration limits.

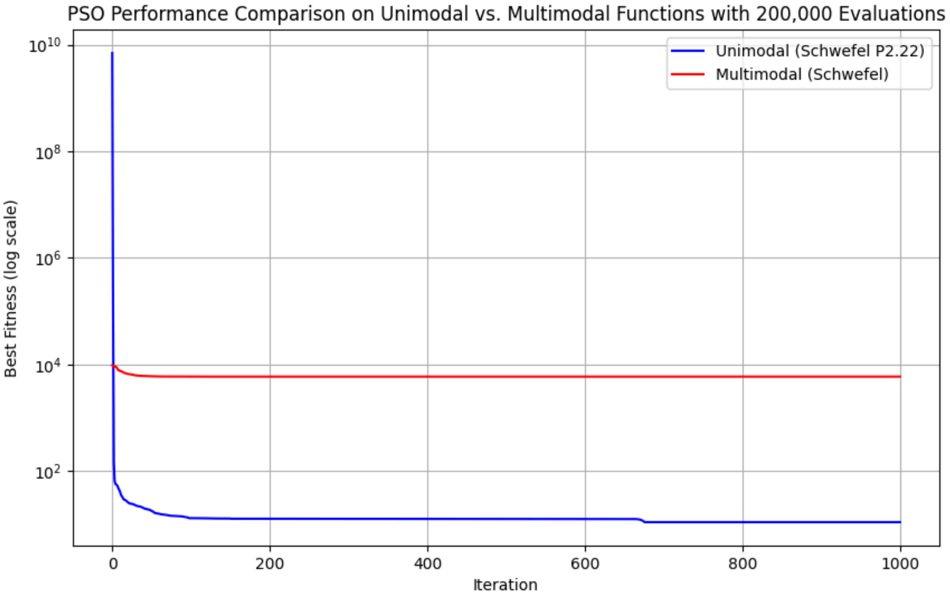
  ```
  num_dimensions = 30
  num_particles = 200
  max_iterations = 1000
  evaluation_limit = 200000
  ```

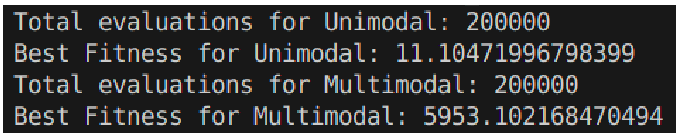
```
# Unimodal Schwefel P2.22
best_position_uni, best_fitness_uni, fitness_history_unimodal, evals_uni
= pso( schwefel_unimodal, num_dimensions, -10, 10, num_particles,
            max_iterations, evaluation_limit)

# Multimodal Schwefel
best_position_multi, best_fitness_multi, fitness_history_multimodal
, evals_multi
= pso( schwefel_multimodal, num_dimensions, -500, 500,
            num_particles, max_iterations, evaluation_limit)
```

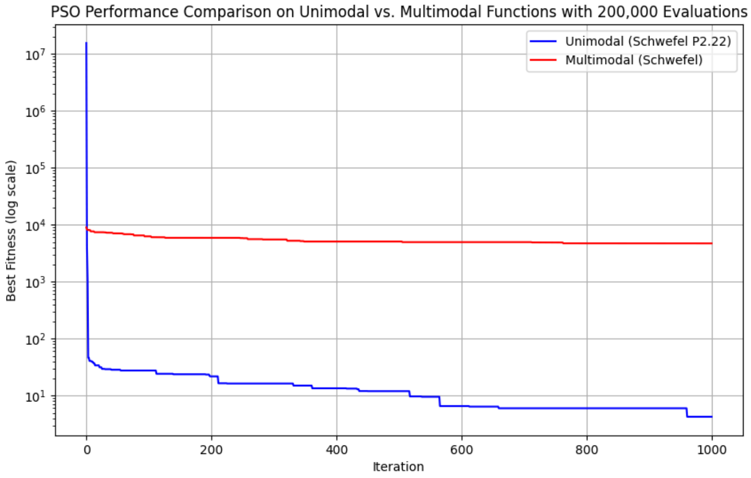
- Plots fitness histories of both functions to observe PSO’s performance.

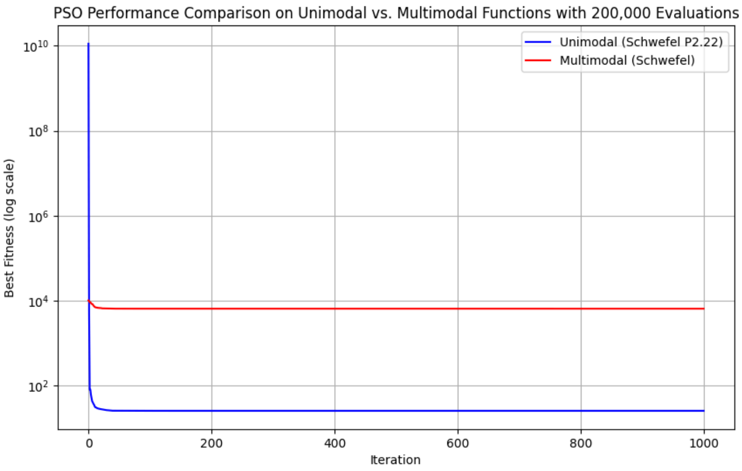
  ```
  print(f"Total evaluations for Unimodal: {evals_uni}")
  print(f"Best Fitness for Unimodal: {best_fitness_uni}")
  print(f"Total evaluations for Multimodal: {evals_multi}")
  print(f"Best Fitness for Multimodal: {best_fitness_multi}")
  ```





```
Total evaluations for Unimodal: 200000
Best Fitness for Unimodal: 11.10471996798399
Total evaluations for Multimodal: 200000
Best Fitness for Multimodal: 5953.102168470494
```





Here's a comparison of the two PSO performance figures, focusing on the effects of different inertia values while keeping cognitive and social coefficients constant:

1. **High Inertia (The figure on the left)**

   For Unimodal (Schwefel P2.22):

   The fitness drops quickly in the initial iterations, but the high inertia causes the particles to retain a higher velocity, resulting in more exploration rather than converging rapidly.

   For Multimodal (Schwefel):

   As a result, the fitness value for the multimodal function remains flat over the iterations, indicating that particles are unable to effectively refine their positions near good solutions.

2. **Low Inertia (The figure on the right)**

   For Unimodal (Schwefel P2.22):

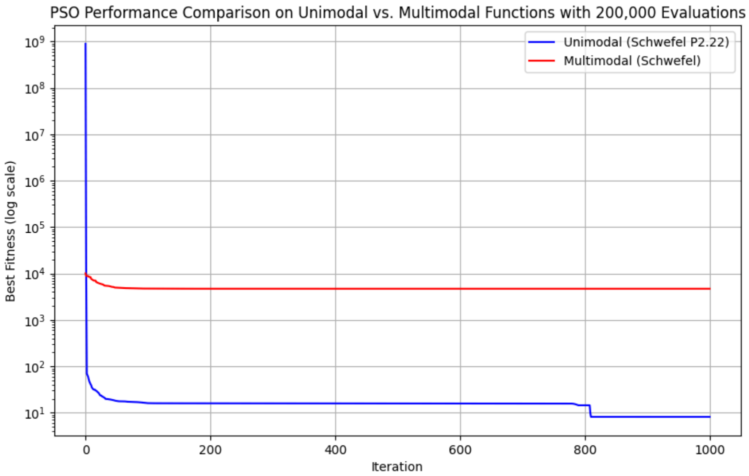
   This leads to faster convergence since the particles are encouraged to focus more on exploitation (fine-tuning around the current best-known solutions) rather than exploration.

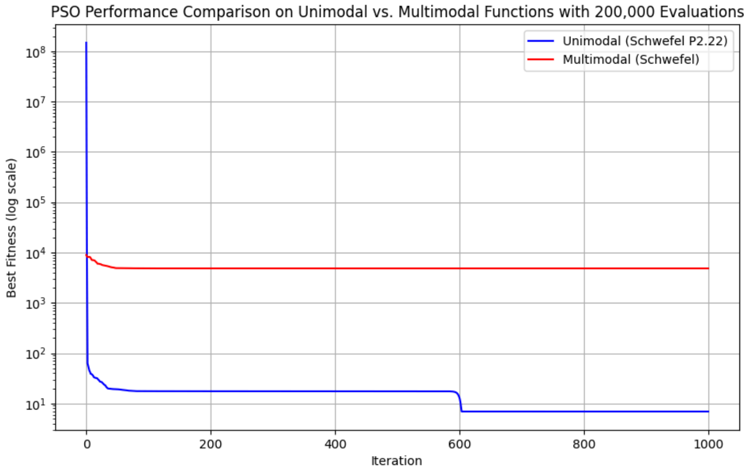
   For Multimodal (Schwefel):

   This figure shows that the lower inertia does not benefit the multimodal function because the particles lack the necessary exploration capability to search widely in the complex landscape.

Summary of the Impact of Inertia:

High inertia is beneficial for exploration, while low inertia helps with convergence. For a complex landscape with multiple optima, adaptive inertia or dynamic tuning could be more effective in balancing exploration and exploitation over time.





Here's a comparison of the two PSO performance figures with inertia fixed at 0.7 and different cognitive coefficients (2.0 vs. 0.5) while keeping the social coefficient constant at 1.5.

1. **Higher Cognitive Coefficient**

   For Unimodal (Schwefel P2.22):

   The blue line shows rapid convergence, reaching a low fitness value by around iteration 200. After iteration 600, the fitness remains stable, indicating that the swarm has effectively converged to a good solution.

   For Multimodal (Schwefel):

   The higher cognitive coefficient causes particles to focus on their individual bests, which can limit exploration in a multimodal landscape, leading to early convergence around local minima.

2. **Lower Cognitive Coefficient**

   For Unimodal (Schwefel P2.22):

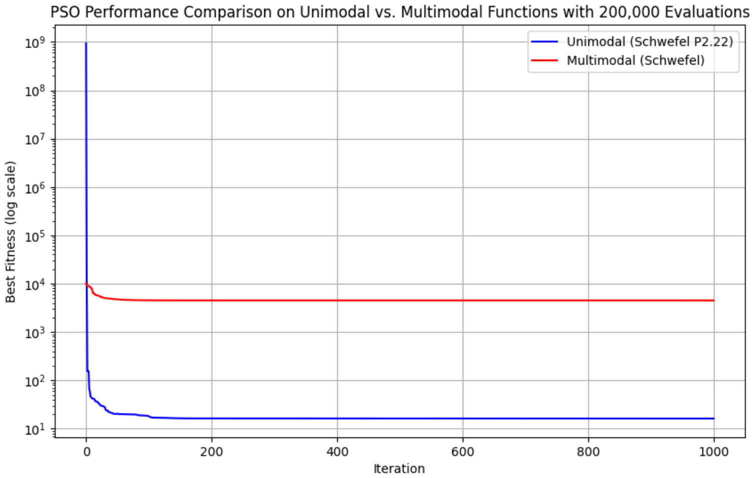
   The setup allows the swarm to balance exploration and exploitation, though it converges slightly slower than the first figure where particles had a stronger personal pull.

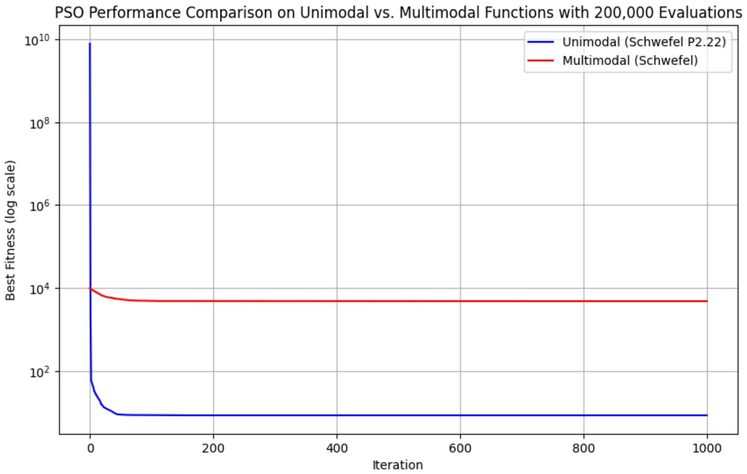
   For Multimodal (Schwefel):

   The lower cognitive coefficient reduces the likelihood of particles being trapped in their individual best positions, but it still doesn't overcome the local minima problem for the multimodal function, as particles gravitate toward the global best.

Summary of the Impact of Cognitive Coefficient:

In general, higher cognitive values favor convergence speed, while lower values allow for broader search.





Here's a comparison of the two PSO performance figures with different social coefficients while keeping inertia and cognitive coefficients constant(inertia fixed at 0.7, cognitive fixed at 1.5):

1. **Higher Social Coefficient(Social coefficient = 2.0)**

   For Unimodal (Schwefel P2.22):

   This results in faster convergence because particles are more inclined to exploit the best-known solution in the swarm.

   For Multimodal (Schwefel):

   The red line initially drops but then stabilizes without significant further improvement.

2. **Lower Social Coefficient(Social coefficient = 0.5)**

   For Unimodal (Schwefel P2.22):

   The blue line shows a slightly slower convergence compared to the first figure, but it still reaches a low fitness value relatively early.

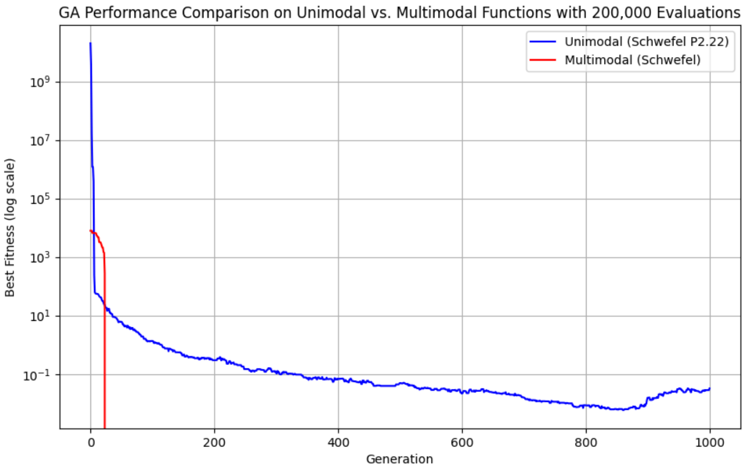
   For Multimodal (Schwefel):

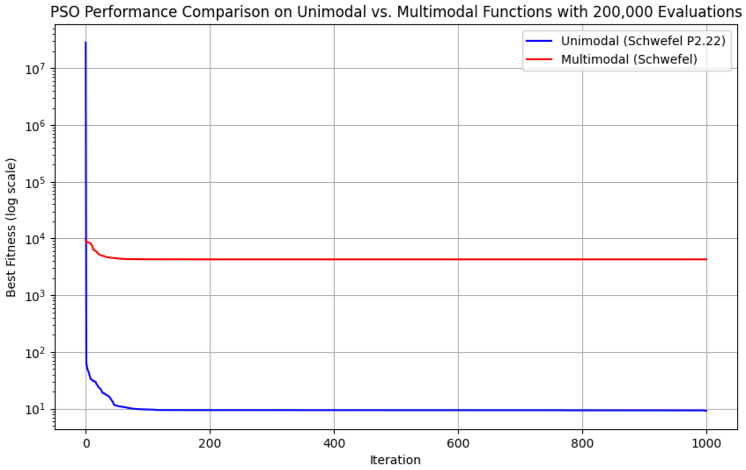
   The red line stabilizes early and does not show significant improvement after the initial drop. The lower social coefficient alone is not sufficient to overcome the challenge of local minima in the multimodal landscape.

Conclusion:

For unimodal functions, a higher social coefficient (2.0) promotes faster convergence by aligning particles toward the global best.

For multimodal functions, a lower social coefficient (0.5) encourages particle independence, but additional strategies (such as adaptive coefficients or randomized initialization) may be necessary to improve exploration.





Here's a comparison of the two optimization algorithms, Genetic Algorithm (GA, the left figure) and Particle Swarm Optimization (PSO, the right figure):

1. **Convergence Speed**

   By using GA, the unimodal function shows a fast initial drop in fitness, but then the convergence slows down and continues to gradually improve over the 1000 generations. The multimodal function converges relatively quickly in the first few generations but then stagnates. There is some improvement early on, but it doesn't reach as low a fitness as the unimodal function.

   By using PSO, the unimodal function achieves rapid convergence to a very low fitness value within the first 100 iterations and then remains stable. The multimodal function reaches a low fitness early on but then plateaus without further improvement.

2. **Final Fitness Achieved**

   By using GA, the unimodal function achieves a very low fitness by the end of the 1000 generations, but it took significantly more generations to reach this level compared to PSO. The multimodal function achieves moderate fitness but does not improve as much after the initial generations, indicating that GA had difficulty escaping local minima or finding better regions in the search space for the multimodal landscape.

   By using PSO, the unimodal function achieves very low fitness within a few iterations, stabilizing around a much lower value compared to GA. The multimodal function reaches a plateau at a moderate fitness level similar to GA but stabilizes faster without additional improvements.

3. **Stability of Convergence**

   GA’s convergence is gradual, with small fluctuations as it continues to explore potential solutions. This is typical in GA, as mutation introduces some randomness that can help with exploration but may slow down final convergence.

   PSO shows a very stable convergence for the unimodal function after the initial rapid improvement, maintaining a low fitness value with little to no fluctuation.

Conclusion:

For Unimodal Functions: PSO is the better choice due to its rapid convergence and stable behavior, reaching low fitness values quickly.

For Multimodal Functions: Both GA and PSO struggle with local minima, but GA offers more consistent exploration, which may lead to better outcomes with additional tuning (e.g., adaptive mutation rates or crossover strategies). However, neither algorithm achieved significant improvements for the multimodal function in this case, suggesting that further strategies (such as hybrid or adaptive approaches) may be required.

Overall, PSO is better suited for unimodal functions, while GA may offer a slight advantage in multimodal landscapes due to its genetic diversity and mutation, which promote exploration.