README - Particle Swarm Optimization for Mixed Variable Optimization

Liora Feldblum & Meron Sabag

Objective

This project presents the process of optimizing a vector of length n=64, where the first half of the vector consists of floating-point values, and the second half consists of integers. The optimization is performed within the search space of [-100,100] for each dimension. Having recently been introduced to Particle Swarm Optimization (PSO), we implemented a PSO-based approach to find the optimal solution to this problem.

Input

The program takes three objective functions from the files provided:

- from MixedVariableObjectiveFunctions import setC
- import MixedVariableObjectiveFunctions as f_mixed
- import ellipsoidFunctions as Efunc

Output

For each function, the results of all N runs are displayed, along with the vector corresponding to the best result across all runs.

Each vector consists of 32 float values and 32 integer values.

How to Run

1. Place the objective function file in the working directory.
2. Execute the Python script from the main function:
   python HW3_PSO.py
3. The program will output the three results sequentially.

Code Breakdown

Particle Class

Represents a single particle in the swarm.

<u>Attributes:</u>

- position_floats, position_ints: Continuous & integer positions.
- velocity_floats, velocity_ints: Continuous & integer velocities.
- x_pbest, f_pbest: Best position & function value found by this particle.

<u>Methods</u>**:**

- **comb_pos()**: Combines float & integer positions.
- **evaluate(objFunc)**: Computes function value & updates personal best.

- **update_position(lb, ub)**: Updates position within bounds.
- **update_floats_velocity(pos_best_g, c1, c2, omega, max_step_size)**: Adjusts velocity for continuous variables.
- **update_ints_velocity(pos_best_g)**: Adjusts velocity for integer variables.

ParticleSwarm Class

Manages the swarm of particles and runs the PSO algorithm.

Attributes**:**

- swarm: List of particles.
- f_best_g, x_best_g: Best global function value & position.
- omega, c1, c2: Hyperparameters.

Methods:

- **get_gbest_from_neighbors(k)**: Identifies the vector that achieves the lowest personal best value among neighboring particles.
- **run(seed)**: Executes PSO, updates velocities, evaluates particles.

Helper Functions (all not used in the final version of the code)

- **weighted_random_choice(max_value)**: Chooses a random integer with a bias towards larger values.
- **small_step(sign, p_sign, p_not_sign)**: Generates a small integer step in a specified direction.
- **large_step(max_step_size, dist)**: Generates a large step, weighted toward larger values.

Main Execution

- Runs PSO for multiple functions (e.g., genHadamardHellipse).
- Optimizes an objective function (MixedVarsEllipsoid).
- Performs multiple runs to find the best solution.

Tuning Parameters

- **Nruns = 5** → Number of attempts to find the best solution for each function *(Line 329)*.
- **Budget = 10,000** → Number of iterations per attempt *(Line 331)*.
- **Population = 100** → Number of particles in the swarm *(Line 332)*.
- **Omega = 0.7298** → Controls the balance between exploration and exploitation *(Line 252)*.
- **C1, C2 = 3, 1.49618** → Determine the impact of personal best and global best positions on velocity updates *(Line 253, 254)*.
- **max_step_size = 5** → Maximum limit for float values *(Line 255)*.