Name: Morgan Sasha Rosca User-name: fnlz75
Algorithm A: Elitist Ant Colony
Algorithm B: Particle Swarm Optimisation
Description of enhancement of Algorithm A: (MMAS/ACS Hybrid + 2-Opt)
Description of enhancement of Algorithm B: ACO + PSO + 3-Opt

1. **Initialization (NN + 2-Opt):** Uses Nearest Neighbour heuristic followed by 2-Opt refinement to generate a high-quality initial tour, informing initial pheromone bounds. **Code Ref:** nearest_neighbour(), two_opt_first(), initialization of seed_tour, seed_len.

1. **MMAS Pheromone Bounds:** Enforces explicit upper (τmax) and lower (τmin) bounds on pheromone trails during updates to prevent stagnation and balance exploration/exploitation. Bounds are dynamically updated. **Code Ref:** Calculation of tau_max/tau_min; max()/min() calls during pheromone updates.

2. **MMAS Selective Pheromone Update:** Only the iteration-best (post 2-Opt) and global-best tours deposit pheromone, focusing reinforcement on elite solutions. **Code Ref:** Pheromone deposit section (comments # deposit: iteration best, # deposit: global best).

3. **ACS Decision Rule:** Employs a pseudo-random rule: with probability q0, ants greedily choose the best next city; otherwise, they use probabilistic selection, balancing exploitation and exploration. **Code Ref:** Tour construction loop (if random.random() < q0: block, comment # ACS decision rule).

5. **Candidate Lists:** Uses pre-computed lists (cand) of nearest neighbours for each city to accelerate the next-city selection process by prioritizing likely good moves. **Code Ref:** cand list initialization and usage in tour construction (comment # candidate list first).

6. **Integrated 2-Opt Local Search:** Applies 2-Opt (first-improvement) to the best tour found in *each iteration* before pheromone updates, combining ACO's global search with 2-Opt's local refinement. **Code Ref:** two_opt_first() definition; call in main loop (comment # light 2-opt on iteration best).

7. **Stagnation Handling:** Resets the pheromone matrix if the global best solution doesn't improve for stagnation_limititerations, helping escape local optima. **Code Ref: s**tagnation counter; if stagnation >= stagnation_limit: block.

1. **Hybrid Multi-Stage Structure:** Replaced the single-stage PSO solver with a three-stage pipeline: (1) PSO optimizes ACO parameters, (2) ACO solves the TSP using optimized parameters, (3) 3-Opt refines the ACO solution. This structure follows this paper's structure: *A new hybrid method based on Particle Swarm Optimization, Ant Colony Optimization and 3-Opt algorithms for Traveling Salesman Problem (Mostafa Mahi, Ömer Kaan Baykan, Halife Kodaz)* **Code Ref:** Overall structure of main execution block, calling pso_find_alpha_beta, ant_colony, and three_opt sequentially.

2. **PSO for ACO Parameter Optimization:** PSO is repurposed from a tour-finding algorithm to a parameter tuner. Particles now represent (α, β) pairs for ACO. The fitness of a particle is evaluated by running a short ACO instance with its parameters, aiming to find the (α, β) combination yielding the best ACO performance. **Code Ref:** pso_find_alpha_beta() function, where particle positions are [a, b] and fitness evaluation calls ant_colony().

3. **ACO as Primary TSP Solver** A standard Ant Colony Optimization algorithm is introduced as the main engine for constructing the TSP tour in Stage 2. It utilizes the pheromone (τ) and heuristic (η) information, guided by the α and β values found by the PSO stage. **Code Ref:** ant_colony() function implementing the ACO logic (tour construction, pheromone updates).

4. **3-Opt Local Search Refinement:** A deterministic 3-Opt local search is added as a final (Stage 3) post-processing step. It takes the best tour found by the ACO stage and attempts to improve it by systematically evaluating 3-edge exchanges. **Code Ref:** three_opt() function definition and its call at the end of the main script.