Enhanced Artificial Bee Colony Algorithm with Tree-structured Parzen Estimator for Multi-Depot Multi-Satellite Vehicle Routing Problem: A Hyperparameter Optimization Study

Summary

• Problem: MDMS-MCVRP optimization using enhanced ABC algorithm

• Data: Excel files converted to weighted demand matrix with depot/satellite/customer classification

• Method: ABC algorithm with TPE-based hyperparameter optimization using Optuna

• Results: TPE shows superior convergence behavior compared to Grid Search

• Contribution: Enhanced ABC with intelligent parameter tuning for complex routing problems

Data Pre-processing :

Renaming-

Depot ID	Location	Product Type
D1	Nagpur	Vitamin B
D2	Cuttack	Diclofenac
D3	Kolkata	Amlodipine
D4	Bhubaneswar	Azithromycin
D5	Noida	Fluconazole
D6	Bhubaneswar	Metformin
D7	Noida	Levocetirizine
D8	Bhubaneswar	Metronidazole
D9	Cuttack	Ofloxacin
D10	Cuttack	Paracetamol

Satellite ID	Location
S1	Mayurbhanj
S2	Bhadrak

Customer ID	Location Name	Facility Type
C1	Baripada	District Head Hospital (DHH)
C2	Badasahi	Healthcare Center
C3	Bahalda	Healthcare Center
C4	Bangiriposi	Healthcare Center
C5	Badampahar	Healthcare Center
C6	Betnoti	Healthcare Center
C7	Bijatola	Healthcare Center
C8	Dukura	Healthcare Center
C9	Gorumahisani	Healthcare Center
C10	Jamda	Healthcare Center
C11	Jamukeswar	Healthcare Center
C12	Jashipur	Healthcare Center
C13	Jharadihi	Healthcare Center
C14	Kaptipada	Healthcare Center
C15	Karanjia	Sub-Divisional Hospital (SDH)
C16	Khunta	Healthcare Center
C17	Kisantandi	Community Health Centre (CHC)
C18	Kostha	Community Health Centre (CHC)

C19	Krushna Chandra Pur	Community Health Centre (CHC)
C20	Kuliana	Healthcare Center
C21	Manada	Healthcare Center
C22	Rairangapur	Sub-Divisional Hospital (SDH)
C23	Rangamatia	Healthcare Center
C24	Raruan	Healthcare Center
C25	Rasgovinda Pur	Healthcare Center
C26	Sirsa	Healthcare Center
C27	Sriram Ch.Pur	Healthcare Center
C28	Sukruli	Healthcare Center
C29	Tato	Healthcare Center
C30	Thakurmunda	Healthcare Center
C31	Tiringi	Healthcare Center
C32	Udala	Sub-Divisional Hospital (SDH)
C33	Murgabadi	Urban Primary Health Centre (UPHC)
C34	Agarpada	Healthcare Center
C35	Barapada	Healthcare Center
C36	Basudevpur	Healthcare Center
C37	Bhadrak	District Head Hospital (DHH)
C38	Bhandari Pokhari	Healthcare Center
C39	Chandabali	Healthcare Center
C40	Dhamnagar	Healthcare Center
C41	Tihidi	Healthcare Center

Demand matrix :

Customer	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
C1	3000	6000	0	0	0	0	0	6000	0	20000
C2	40000	0	0	0	2000	0	18000	0	0	40000
C3	0	0	0	0	0	0	0	2000	0	15000
C4	10000	9000	8820	4000	4000	0	4800	10000	0	20000
C5	100000	9900	50400	0	5000	50000	0	20000	0	50000
C6	20000	0	10080	10000	2000	0	4800	20000	5000	40000
C7	0	6000	0	4800	0	0	0	0	0	10000
C8	0	0	0	0	0	0	0	0	0	20000
C9	2000	0	6300	0	2000	0	4800	0	0	5000
C10	15000	9000	3780	4000	0	0	7800	5000	10000	10000
C11	0	0	0	0	0	0	0	0	0	5000
C12	20000	0	12600	0	0	0	0	0	0	20000
C13	2000	1800	6300	2000	1000	5000	900	1000	0	5000
C14	20000	0	0	0	2000	0	0	0	0	20000
C15	30000	0	18900	13000	0	0	0	0	0	20000
C16	10000	0	8820	0	0	0	0	20000	0	10000
C17	12000	0	8820	0	0	0	9900	10000	0	20000
C18	0	9000	3780	5000	0	6000	15000	10000	0	20000
C19	10000	6000	5040	0	0	3000	4800	2000	0	10000
C20	5000	9000	8820	0	0	5000	0	10000	0	20000
C21	0	0	7560	0	0	0	0	0	0	5000
C22	40000	4800	8820	14000	0	3000	4800	5000	10000	30000
C23	8000	3000	6300	0	300	7000	0	0	0	5000
C24	0	0	0	0	0	0	0	0	0	5000
C25	10000	0	0	0	0	0	9000	10000	0	10000
C26	8000	0	8820	0	0	0	0	20000	5000	20000
C27	5000	0	0	0	0	0	4800	0	0	10000
C28	0	0	0	0	0	0	0	0	0	5000
C29	10000	0	8820	2000	0	0	0	0	0	20000
C30	10000	0	0	5250	0	0	0	0	0	20000
C31	0	0	0	0	0	0	0	10000	5000	5000
C32	20000	3900	0	0	0	0	0	0	0	5000
C33	0	0	0	0	0	0	0	0	0	5000
C34	0	0	0	10000	0	0	10000	0	8000	20000
C35	0	0	5000	5000	600	0	15000	0	5000	20000
C36	0	0	0	0	0	0	0	0	5000	30000
C37	150000	2500	5000	31250	0	20000	91000	10000	0	128600
C38	15000	0	0	3000	1000	5000	15000	0	15000	20000
C39	0	0	5000	16750	1000	0	10000	0	5000	50000
C40	5000	30000	5000	16250	0	0	10000	0	5000	30000
C41	0	0	5000	5000	0	0	15000	0	0	30000

Conversion Methodology:

Assumption: Each demand unit represents a pharmaceutical bundle

Bundle Specification: 10 tablets per bundle

Individual Tablet Weight: 1.5 grams

Bundle Weight: 10 tablets × 1.5 grams = 15 grams = 0.015 kg

Conversion Formula: Demand Weight (kg) = Number of Bundles × 0.015 kg/bundle

Demand in kgs:

Customer	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
C1	45.0	90.0	0.0	0.0	0.0	0.0	0.0	90.0	0.0	300.0
C2	600.0	0.0	0.0	0.0	30.0	0.0	270.0	0.0	0.0	600.0
C3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	30.0	0.0	225.0
C4	150.0	135.0	132.3	60.0	60.0	0.0	72.0	150.0	0.0	300.0
C5	1500.0	148.5	756.0	0.0	75.0	750.0	0.0	300.0	0.0	750.0
C6	300.0	0.0	151.2	150.0	30.0	0.0	72.0	300.0	75.0	600.0
C7	0.0	90.0	0.0	72.0	0.0	0.0	0.0	0.0	0.0	150.0
C8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	300.0
C9	30.0	0.0	94.5	0.0	30.0	0.0	72.0	0.0	0.0	75.0
C10	225.0	135.0	56.7	60.0	0.0	0.0	117.0	75.0	150.0	150.0
C11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	75.0
C12	300.0	0.0	189.0	0.0	0.0	0.0	0.0	0.0	0.0	300.0
C13	30.0	27.0	94.5	30.0	15.0	75.0	13.5	15.0	0.0	75.0
C14	300.0	0.0	0.0	0.0	30.0	0.0	0.0	0.0	0.0	300.0
C15	450.0	0.0	283.5	195.0	0.0	0.0	0.0	0.0	0.0	300.0
C16	150.0	0.0	132.3	0.0	0.0	0.0	0.0	300.0	0.0	150.0
C17	180.0	0.0	132.3	0.0	0.0	0.0	148.5	150.0	0.0	300.0
C18	0.0	135.0	56.7	75.0	0.0	90.0	225.0	150.0	0.0	300.0
C19	150.0	90.0	75.6	0.0	0.0	45.0	72.0	30.0	0.0	150.0
C20	75.0	135.0	132.3	0.0	0.0	75.0	0.0	150.0	0.0	300.0
C21	0.0	0.0	113.4	0.0	0.0	0.0	0.0	0.0	0.0	75.0
C22	600.0	72.0	132.3	210.0	0.0	45.0	72.0	75.0	150.0	450.0
C23	120.0	45.0	94.5	0.0	4.5	105.0	0.0	0.0	0.0	75.0
C24	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	75.0
C25	150.0	0.0	0.0	0.0	0.0	0.0	135.0	150.0	0.0	150.0
C26	120.0	0.0	132.3	0.0	0.0	0.0	0.0	300.0	75.0	300.0
C27	75.0	0.0	0.0	0.0	0.0	0.0	72.0	0.0	0.0	150.0
C28	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	75.0
C29	150.0	0.0	132.3	30.0	0.0	0.0	0.0	0.0	0.0	300.0
C30	150.0	0.0	0.0	78.75	0.0	0.0	0.0	0.0	0.0	300.0
C31	0.0	0.0	0.0	0.0	0.0	0.0	0.0	150.0	75.0	75.0
C32	300.0	58.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	75.0
C33	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	75.0
C34	0.0	0.0	0.0	150.0	0.0	0.0	150.0	0.0	120.0	300.0
C35	0.0	0.0	75.0	75.0	9.0	0.0	225.0	0.0	75.0	300.0
C36	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	75.0	450.0
C37	2250.0	37.5	75.0	468.75	0.0	300.0	1365.0	150.0	0.0	1929.0
C38	225.0	0.0	0.0	45.0	15.0	75.0	225.0	0.0	225.0	300.0
C39	0.0	0.0	75.0	251.25	15.0	0.0	150.0	0.0	75.0	750.0
C40	75.0	450.0	75.0	243.75	0.0	0.0	150.0	0.0	75.0	450.0
C41	0.0	0.0	75.0	75.0	0.0	0.0	225.0	0.0	0.0	450.0

Algorithm Explanations

By Steps:

Complete Algorithm Steps for Enhanced ABC with TPE Optimization

Step 1: Problem Setup and Data Preparation

- Load the MDMS-MCVRP problem data from Excel files
- Convert demand matrix from tablet bundles to weight (bundles × 0.015 kg)
- Define network structure: 10 depots, 2 satellites, 41 customers
- Set up distance matrix and identify forbidden routes (distance ≥ 9999)
- Define vehicle capacities and satellite storage limits

Step 2: Initialize TPE Hyperparameter Optimization

- Define search space for ABC parameters:
- swarm_size:
- max_iterations:
- employed_ratio: 0.4, 0.5, 0.6
- onlooker_ratio: 0.2, 0.3, 0.4
- limit:
- Start with empty observation history
- Set number of trials (e.g., 30 for quick mode)

Step 3: Begin TPE Trial Loop

- For each trial from 1 to total trials:
- If trial ≤ 10: randomly select parameters from search space
- If trial > 10: use TPE to intelligently suggest next parameters
- Record suggested parameters for this trial

Step 4: Run ABC Algorithm with Suggested Parameters

- Use the parameters suggested by TPE to run ABC optimization
- Initialize bee swarm with specified swarm_size
- Set maximum iterations from suggested parameters

Step 5: ABC Phase 1 - Initialize Bee Population

- Create random routing solutions for each bee in the swarm
- Each solution includes:
- Routes from depots to satellites (primary vehicles)
- Routes from satellites to customers (secondary vehicles)
- Check that all solutions satisfy capacity constraints
- Calculate initial fitness for each solution (total route cost + penalties)

Step 6: ABC Phase 2 - Employed Bee Search

- For each employed bee (based on employed_ratio):
- Create a neighbor solution by modifying current routes
- Apply local search operators (swap customers, relocate, exchange)
- Ensure the new solution is feasible
- Calculate fitness of neighbor solution
- If neighbor is better, replace current solution
- If not improved, increase trial counter for this solution

Step 7: ABC Phase 3 - Onlooker Bee Selection

- Calculate selection probability for each solution based on fitness
- Better solutions get higher probability of being selected
- For each onlooker bee (based on onlooker_ratio):
- Select a solution using roulette wheel selection
- Perform intensive local search on selected solution
- Update solution if improvement is found

Step 8: ABC Phase 4 - Scout Bee Replacement

- Check each solution's trial counter
- If any solution hasn't improved for 'limit' iterations:
- Replace it with a completely new random solution
- Reset its trial counter to zero
- This prevents getting stuck in local optima

Step 9: ABC Phase 5 - Update Best Solution

- Find the best solution in current swarm

- Update global best if a better solution is found
- Record improvement details

Step 10: ABC Termination Check

- If maximum iterations reached OR convergence achieved:
- Stop ABC algorithm
- Return best solution fitness
- Otherwise, go back to Step 6 (next ABC iteration)

Step 11: Record TPE Trial Results

- Store the ABC result (fitness value) with corresponding parameters
- Add this (parameters, fitness) pair to TPE observation history
- TPE learns which parameter combinations work better

Step 12: Update TPE Model

- Separate observations into “good” and “bad” performance groups
- Fit probability models for good and bad parameter combinations
- Prepare TPE to suggest better parameters for next trial

Step 13: TPE Trial Loop Continue

- If more trials remaining, go back to Step 3
- TPE will now suggest parameters more intelligently based on learning

Step 14: Final Hyperparameter Selection

- After all trials completed, identify best performing parameters
- These are the parameters that achieved lowest fitness (best routing cost)

Step 15: Final Solution Generation

- Run ABC algorithm one more time with best found parameters
- Generate the final optimized vehicle routing solution
- Validate solution satisfies all constraints

Step 16: Results Analysis and Output

- Calculate final route costs and efficiency metrics
- Generate routing assignments:
- Which depot serves which satellites

- Which satellite serves which customers
- Vehicle utilization and load distribution
- Create visualization and performance reports

Step 17: Solution Validation

- Verify all 41 customers are served
- Check vehicle capacity constraints are met
- Ensure satellite capacity limits are respected
- Confirm no forbidden routes are used

Step 18: Performance Comparison

- Compare TPE results with Grid Search baseline
- Analyze convergence behavior and parameter importance
- Generate graphs showing optimization progress

Step 1: Problem Setup and Data Preparation

The algorithm begins by loading the Multi-Depot Multi-Satellite Vehicle Routing Problem data from the provided Excel files containing demand information and distance matrices. The original demand data, which represents pharmaceutical bundles, is converted to weight-based measurements using the conversion factor of 0.015 kg per bundle (10 tablets × 1.5 grams each). The network structure is established with 10 pharmaceutical depots, 2 satellite distribution centers, and 41 customer healthcare facilities. The distance matrix is processed to identify feasible routes, marking connections with distances ≥ 9999 as forbidden routes that cannot be used in the routing solution. Vehicle capacities for both primary vehicles (depot-to-satellite) and secondary vehicles (satellite-to-customer) are defined, along with storage capacity limits for each satellite facility.

Step 2: Initialize TPE Hyperparameter Optimization

The Tree-structured Parzen Estimator framework is initialized to optimize the ABC algorithm's hyperparameters. A comprehensive search space is defined for five critical parameters: swarm_size ranging from 30 to 80 bees, max_iterations from 100 to 300, employed_ratio from 0.4 to 0.6, onlooker_ratio from 0.2 to 0.4, and abandonment limit from 10 to 20. The TPE system starts with an empty observation history that will store the relationship between parameter combinations and their resulting performance. The total number of optimization trials is set based on the desired exploration depth, typically 30 trials for quick mode or up to 100+ for comprehensive optimization.

Step 3: Begin TPE Trial Loop

The main optimization loop initiates where each trial represents one complete run of the ABC algorithm with a specific parameter configuration. For the initial trials (typically the first 10), the TPE system randomly samples parameters from the defined search space to gather diverse initial observations. After this exploration phase, TPE switches to intelligent parameter suggestion mode, using its learned probabilistic models to propose parameter combinations that are more likely to yield superior performance based on the accumulated historical data from previous trials.

Step 4: Run ABC Algorithm with Suggested Parameters

Once TPE suggests a parameter configuration for the current trial, these parameters are passed to the ABC algorithm as its operational settings. The ABC solver is instantiated with the specified swarm_size determining how many bee solutions will be maintained in the population, max_iterations controlling how long the optimization will run, and the ratio parameters determining the distribution of computational effort among different bee types. This creates a complete ABC instance ready to solve the MDMS-MCVRP with the TPE-suggested configuration.

Step 5: ABC Phase 1 - Initialize Bee Population

The ABC algorithm creates its initial population by generating random routing solutions for each bee in the swarm. Each solution represents a complete assignment of vehicles to routes, including primary vehicle paths from depots to satellites and secondary vehicle paths from satellites to customers. The algorithm ensures that each randomly generated solution is feasible by checking vehicle capacity constraints, satellite storage limits, and route connectivity requirements. Initial fitness values are calculated for every solution by summing total transportation costs and adding penalty terms for any constraint violations that might exist.

Step 6: ABC Phase 2 - Employed Bee Search

In this phase, employed bees perform local search operations on their assigned solutions. Each employed bee generates a neighboring solution by applying various modification operators such as swapping customers between routes, relocating customers to different vehicles, or exchanging route segments between vehicles. The algorithm ensures that any newly generated solution maintains feasibility by repairing constraint violations and validating route connectivity. If the neighboring solution achieves better fitness than the current solution, the employed bee adopts the new solution; otherwise, it increments a trial counter tracking how long the solution has remained unimproved.

Step 7: ABC Phase 3 - Onlooker Bee Selection

Onlooker bees implement the exploitation phase by focusing computational effort on the most promising solutions discovered by employed bees. The algorithm calculates selection

probabilities for each solution based on their relative fitness quality, ensuring that better solutions receive higher probability of being selected for further improvement. Each onlooker bee uses roulette wheel selection to choose a solution probabilistically, then performs intensive local search operations on the selected solution. This concentrated search effort often leads to significant improvements in already good solutions, accelerating convergence toward optimal routing configurations.

Step 8: ABC Phase 4 - Scout Bee Replacement

The scout bee mechanism maintains population diversity by identifying and replacing stagnant solutions that have not improved for a specified number of iterations (the limit parameter). The algorithm examines each solution's trial counter, and when any solution exceeds the abandonment threshold, it is completely replaced with a newly generated random solution. This replacement strategy prevents premature convergence to local optima by continuously introducing fresh genetic material into the population, ensuring that the search can escape from regions of the solution space that may trap the algorithm.

Step 9: ABC Phase 5 - Update Best Solution

After completing all bee phases in an iteration, the algorithm evaluates the entire swarm to identify the current best solution. The global best solution is updated if any bee has discovered a routing configuration with lower total cost than previously found solutions. The algorithm maintains detailed records of improvements, including the iteration number, fitness improvement amount, and the specific route modifications that led to the enhancement. This tracking provides valuable insights into the algorithm's convergence behavior and optimization progress.

Step 10: ABC Termination Check

The ABC algorithm evaluates its termination criteria by checking whether the maximum number of iterations has been reached or if convergence has been achieved based on predefined criteria such as lack of improvement over consecutive iterations. If termination conditions are met, the algorithm concludes its search and returns the fitness value of the best solution found. Otherwise, the algorithm continues to the next iteration by returning to the employed bee phase, maintaining the evolutionary search process until optimal or near-optimal solutions are discovered.

Step 11: Record TPE Trial Results

Upon completion of the ABC algorithm run, the resulting best fitness value is recorded along with the parameter configuration that produced it. This creates a new observation pair that

links specific hyperparameter values to their corresponding performance outcomes. The TPE system stores this information in its growing database of trial results, which forms the foundation for learning which parameter combinations tend to produce superior routing solutions and which combinations should be avoided in future trials.

Step 12: Update TPE Model

The TPE framework processes the accumulated observation history to update its probabilistic models that guide future parameter selection. The algorithm partitions the historical data into "good" and "bad" performance groups based on a threshold (typically the 25th percentile of observed fitness values). Separate probability density functions are fitted to model the distribution of parameters that led to good performance versus those that resulted in poor performance. These models enable TPE to mathematically reason about which unexplored parameter combinations are most likely to yield superior results.

Step 13: TPE Trial Loop Continue

If additional trials remain in the optimization budget, the algorithm returns to the trial selection phase where TPE now leverages its updated models to make intelligent parameter suggestions. The system balances exploration of potentially promising but untested parameter regions with exploitation of parameter neighborhoods that have previously demonstrated good performance. This adaptive learning process enables TPE to become increasingly effective at proposing high-quality parameter configurations as more trials are completed.

Step 14: Final Hyperparameter Selection

After exhausting the trial budget or achieving satisfactory convergence, the TPE system analyzes all completed trials to identify the parameter configuration that achieved the best overall performance. This selection process considers not only the single best trial but may also account for consistency and robustness across multiple similar parameter configurations. The selected hyperparameters represent the algorithm's learned optimal settings for solving MDMS-MCVRP instances with characteristics similar to the training problem.

Step 15: Final Solution Generation

Using the best discovered hyperparameters, the algorithm executes one final run of the ABC optimization to generate the definitive routing solution. This final execution ensures that the returned solution represents the algorithm's best effort using optimal parameter settings rather than just the best solution encountered during the hyperparameter search process. The resulting solution provides complete vehicle route assignments, load distributions, and

operational details necessary for implementing the optimized pharmaceutical distribution plan.

Step 16: Results Analysis and Output

The algorithm generates comprehensive output including detailed route assignments specifying which primary vehicles travel between which depots and satellites, and which secondary vehicles serve customers from each satellite. Vehicle utilization metrics, load distribution analysis, and total transportation costs are calculated and reported. The solution is formatted into user-friendly representations such as route lists, cost breakdowns, and efficiency measures that facilitate practical implementation and performance evaluation of the optimized distribution network.

Step 17: Solution Validation

A thorough validation process verifies that the final solution satisfies all problem constraints including complete customer coverage, vehicle capacity limits, satellite storage restrictions, and route feasibility requirements. The algorithm confirms that all 41 customer locations receive their required pharmaceutical deliveries, no vehicle exceeds its weight capacity, satellite facilities operate within their storage limits, and no forbidden route connections are utilized in the final routing plan.

Step 18: Performance Comparison

The algorithm concludes by comparing the TPE-optimized results against baseline methods such as Grid Search to demonstrate the effectiveness of the intelligent hyperparameter optimization approach. Convergence analysis reveals how TPE's adaptive learning leads to superior parameter discovery compared to exhaustive or random search methods. Performance metrics, optimization trajectories, and parameter sensitivity analysis provide insights into the algorithm's behavior and validate the benefits of combining Bayesian optimization with metaheuristic routing algorithms for solving complex pharmaceutical distribution problems.

Write in research paper

The Artificial Bee Colony (ABC) algorithm is a powerful nature-inspired metaheuristic based on the intelligent foraging behavior of honey bee swarms, originally proposed by Karaboga (2005). It effectively balances exploration and exploitation to solve complex combinatorial optimization problems. In this research, the ABC algorithm is adapted and enhanced to solve the Multi-Depot Multi-Satellite Vehicle Routing Problem (MDMS-MCVRP) efficiently, with automated hyperparameter tuning via methods such as TPE and Grid Search.

The ABC process begins by conceptualizing potential solutions as "food sources," where each food source represents a candidate vehicle routing plan across multiple distribution echelons: from depots to satellites (first echelon) and satellites to customers (second echelon). Employed bees exploit these existing food sources by refining the corresponding solutions, onlooker bees select food sources based on information shared by employed bees, and scout bees explore new areas by randomly generating new solutions to maintain the swarm's diversity.

Initially, the algorithm generates a population, or swarm, of solutions. This involves defining a three-dimensional solution structure, where the first dimension represents the two echelons of the distribution network, the second dimension indexes the vehicles in each echelon, and the third dimension represents the sequence of nodes (depots, satellites, customers) a vehicle visits. For the second echelon, vehicles are randomly assigned customers under feasibility and capacity constraints until all customers are allocated. If any customer remains unallocated after exhausting available satellites, the problem is deemed infeasible. Similarly, for the first echelon, vehicles are assigned satellites. The combined routes from both echelons form initial feasible solutions that constitute the swarm.

Once the swarm is initialized, it is divided into employed and unemployed bees in proportions controlled by a parameter α. The unemployed bees are further split into onlookers and scouts via parameter β. The algorithm iteratively updates the solutions through the behaviors modeled by these bees:

- **Employed Bee Phase**: Each employed bee modifies its current solution by exploring neighboring solutions (through local search or mutation-like operations). If an improved solution is found, the bee updates its position accordingly.

- **Onlooker Bee Phase**: Onlooker bees observe the dances (information sharing) of employed bees and probabilistically select a food source to exploit. Using this information, they perform intensive local search (akin to crossover) on promising solutions, replacing their current solution only if an improvement is found.

- **Scout Bee Phase:** Scouts explore for new, unexplored food sources by generating random solutions. When a scout finds a better solution than its current one, it replaces the latter to enhance the swarm's diversity and avoid premature convergence.

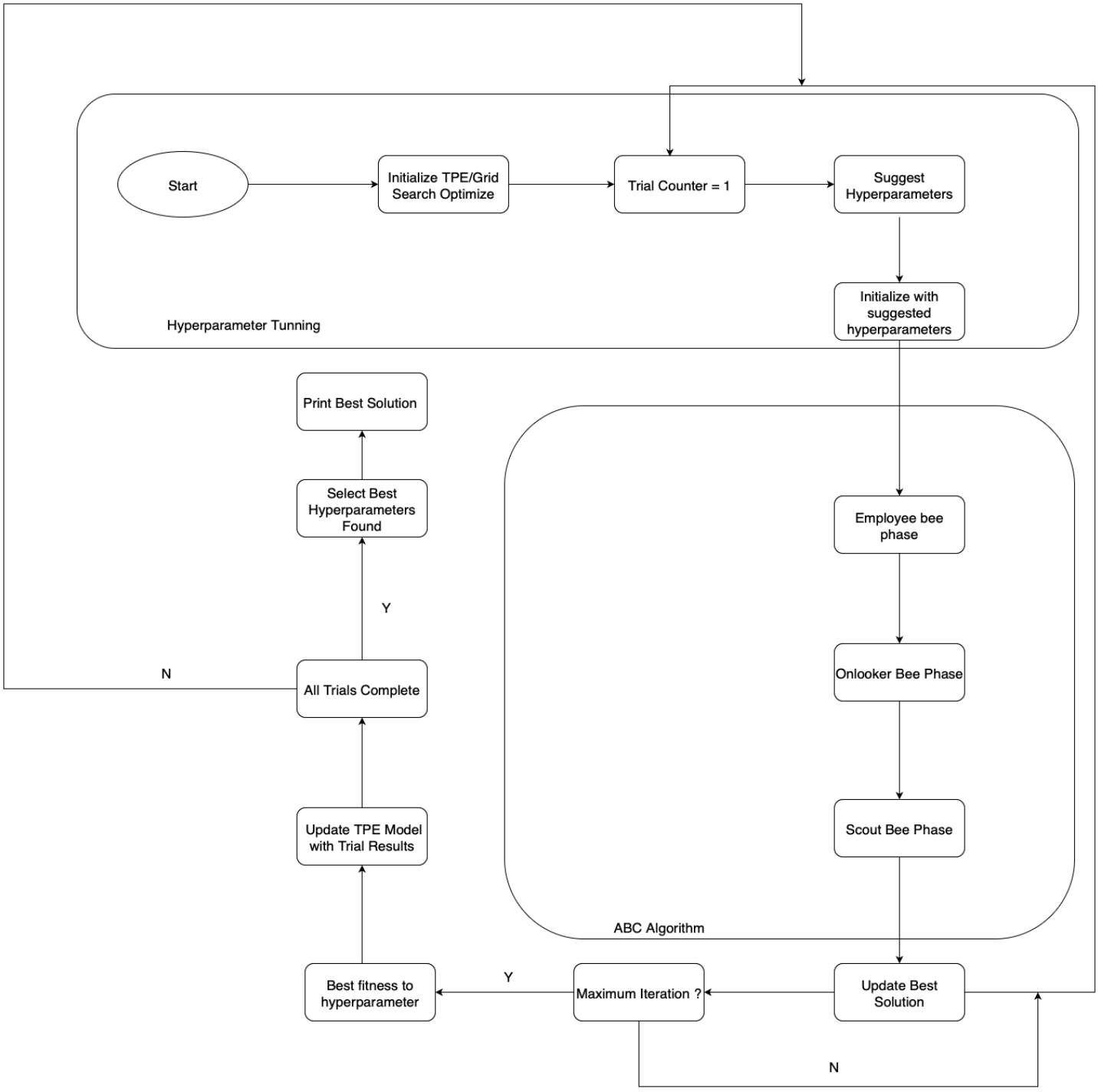
These phases repeat iteratively until the stopping criteria (typically the maximum number of iterations or convergence thresholds) are met, with the employed and unemployed swarms combined at each step to form an updated population.

The algorithm maintains rigorous constraint handling, including vehicle and satellite capacity restrictions, and ensures route feasibility by penalizing violations in the fitness evaluation. This fitness function, combining route cost and penalty terms, guides the search toward optimal and feasible routing solutions.

In this work, the ABC algorithm is further augmented with advanced hyperparameter tuning using Tree-structured Parzen Estimator (TPE) and Grid Search methods to intelligently select algorithm control parameters such as swarm size, the number of iterations, employed and onlooker bee ratios, and abandonment limits. This adaptive hyperparameter selection improves convergence speed and solution quality.

Overall, the enhanced ABC with automated tuning offers a flexible, scalable, and effective framework for solving the complex MDMS-MCVRP, producing near-optimal routing in reasonable computational time suitable for real-world pharmaceutical distribution applications.

This narrative reframes your original explanation into a smooth, research-appropriate description highlighting your algorithm's logic, operation phases, and enhancement via hyperparameter tuning. You may adjust or expand specific details as needed.


Start
Initialize TPE/Grid Search Optimize
Trial Counter = 1
Suggest Hyperparameters
Initialize with suggested hyperparameters
Hyperparameter Tunning
Print Best Solution
Select Best Hyperparameters Found
Y
N
All Trials Complete
Update TPE Model with Trial Results
Best fitness to hyperparameter
Y
Maximum Iteration ?
Update Best Solution
N
Employee bee phase
Onlooker Bee Phase
Scout Bee Phase
ABC Algorithm

Flow Chart

Pseudocode

Algorithm 1: Enhanced ABC Algorithm with Hyperparameter Optimization for MDMS-MCVRP

Requires: D, S, C, K_1, K_2, demand_matrix, distance_matrix

Ensure: Vehicle capacity constraints, Satellite capacity constraints, Customer coverage

Input: Problem instance P, Hyperparameter trials n_trials

Output: Best solution S*, Best hyperparameters θ*

▷ Hyperparameter Optimization Phase

Initialize TPE optimizer

best_fitness ← ∞

best_params ← ∅

observation_history ← ∅

for trial ← 1 to n_trials do

  if trial ≤ startup_trials then

    params ← random_sample(parameter_space)

  else

    params ← tpe.suggest_parameters(observation_history)

  end if

  fitness ← ABC_SOLVE(P, params)

  observation_history ← observation_history ∪ {(params, fitness)}

  if fitness < best_fitness then

    best_fitness ← fitness

```
        best_params ← params
    end if

    tpe.update_model(observation_history)
end for

θ* ← best_params
S* ← ABC_SOLVE(P, θ*)

Function ABC_SOLVE(problem P, parameters θ):
▷ Swarm Initialization Phase
swarm ← ∅
while |swarm| < θ.swarm_size do
    E₁ ← ∅  // Primary routes (depot to satellite)
    E₂ ← ∅  // Secondary routes (satellite to customer)

    for i in D do
        E₁[i] ← [i, i]  // Initialize depot route
    end for

    for j in S do
        E₂[j] ← [j, j]  // Initialize satellite route
    end for

    C_copy ← copy(C)
    while |C_copy| > 0 do
```

```
        sat ← random_choice(S)
        if capacity_check(sat, C_copy) then
            customer ← random_choice(C_copy)
            E₂[sat].insert(-1, customer)
            C_copy.remove(customer)
        end if
    end while

    swarm.insert([E₁, E₂])
end while

richest_food ← min(swarm, key=fitness)

▷ Main ABC Optimization Loop
for iteration ← 1 to θ.max_iterations do

    ▷ Employed Bee Phase
    employed ← swarm[:⌊α × |swarm|⌋]
    for bee in employed do
        new_bee ← modify_solution(bee)
        if fitness(new_bee) < fitness(bee) then
            bee ← new_bee
        end if
    end for

    ▷ Onlooker Bee Phase
```

```
unemployed ← swarm[⌊α × |swarm|⌋:]
onlookers ← unemployed[:⌊β × |unemployed|⌋]

probabilities ← calculate_selection_prob(employed)
for bee in onlookers do
   selected_bee ← roulette_wheel_selection(employed, probabilities)
   new_bee ← intensive_search(selected_bee)
   if fitness(new_bee) < fitness(bee) then
      bee ← new_bee
   end if
end for

▷ Scout Bee Phase
scouts ← unemployed[⌊β × |unemployed|⌋:]
for bee in scouts do
   if stagnation_check(bee, θ.limit) then
      bee ← generate_new_solution(P)
   end if
end for

swarm ← employed + onlookers + scouts
current_best ← min(swarm, key=fitness)
if fitness(current_best) < fitness(richest_food) then
   richest_food ← current_best
end if
```

```
end for

return fitness(richest_food)

Function modify_solution(solution):
    operation ← random_choice([swap, relocate, cross_exchange])
    return apply_operation(solution, operation)

Function calculate_selection_prob(employed_bees):
    total_fitness ← sum(1/fitness(bee) for bee in employed_bees)
    return [(1/fitness(bee))/total_fitness for bee in employed_bees]

Function intensive_search(solution):
    best ← solution
    for attempt ← 1 to local_search_iterations do
        candidate ← modify_solution(best)
        if fitness(candidate) < fitness(best) then
            best ← candidate
        end if
    end for
    return best

Function stagnation_check(solution, limit):
    return solution.trial_counter > limit

Function fitness(solution):
```

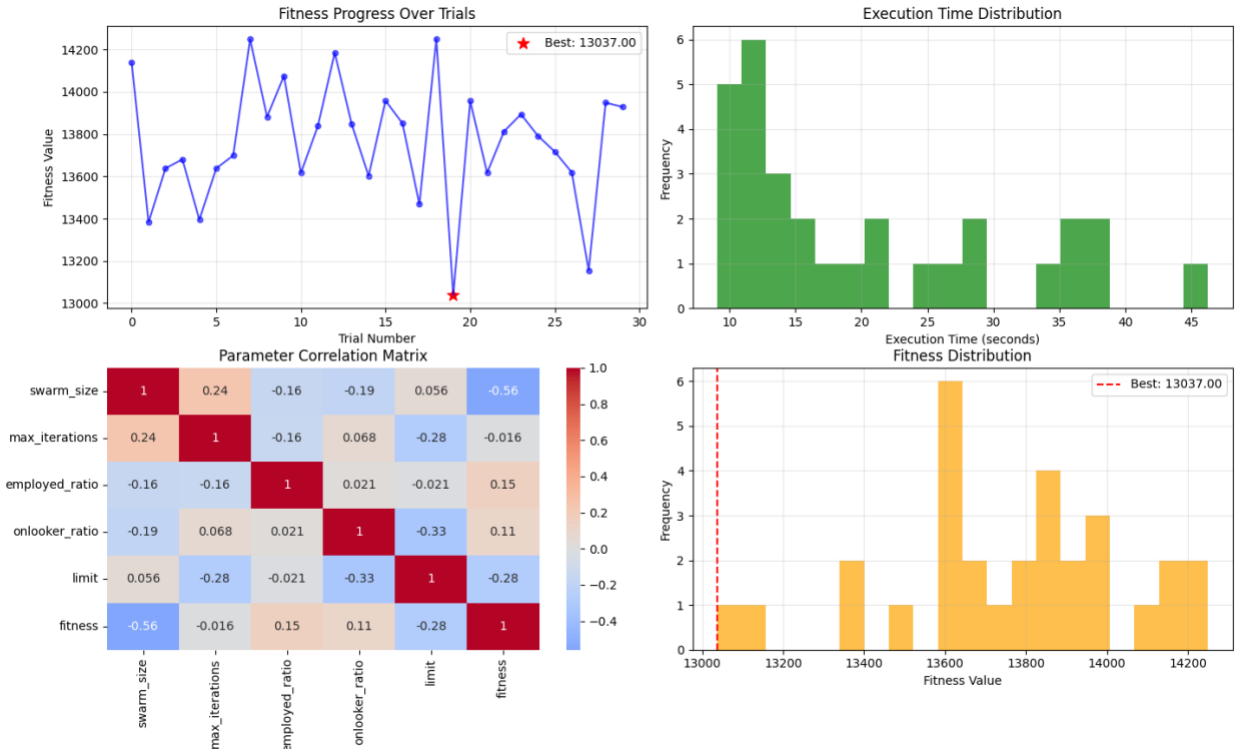
routing_cost ← calculate_total_distance(solution)

penalty ← calculate_constraint_violations(solution)

return routing_cost + λ × penalty

Total Network Demand: 3,812.82 kg

- Average Bundle Weight: 0.015 kg (15 grams)
- Total Bundles Across Network: 254,188 bundles
- Total Tablets: 2,541,880 tablets



Sampler: GRID

Search Mode: quick

Trials: 30

Runs per trial: 1

Grid Search: Testing up to 2400 parameter combinations

Total time: 606.89 seconds

Best fitness: 13037.00

Best parameters:

swarm_size: 100

max_iterations: 550

employed_ratio: 0.4

onlooker_ratio: 0.2

limit: 35

Best trial: 19. Best value: 13037: 100%|██████████| 30/30 [10:06<00:00, 20.23s/it]

Parameter Importance:

swarm_size: 0.3574

limit: 0.2543

employed_ratio: 0.1376

onlooker_ratio: 0.1330

max_iterations: 0.1178

FINAL BEST SOLUTION ROUTES:

🚚 Primary Vehicle Routes (Depots → Satellites):

Vehicle D1: D1 → S1 → D1 (1 satellites)

Vehicle D2: D2 → S2 → D2 (1 satellites)

Vehicle D3: D3 → S1 → D3 (1 satellites)

Vehicle D4: D4 → S1 → S2 → D4 (2 satellites)

Vehicle D5: D5 → S2 → S1 → D5 (2 satellites)

Vehicle D6: D6 → S2 → D6 (1 satellites)

Vehicle D7: D7 → S1 → D7 (1 satellites)

Vehicle D8: D8 → S2 → D8 (1 satellites)

Vehicle D9: D9 → S2 → D9 (1 satellites)

Vehicle D10: D10 → S2 → D10 (1 satellites)

🚐 Secondary Vehicle Routes (Satellites → Customers):

Vehicle S1: S1 → C10 → C27 → C4 → C15 → C32 → C23 → C20 → C22 → C5 → C11 → C2 → C40 → C19 → C1 → C6 → C33 → C17 → S1 (17 customers)

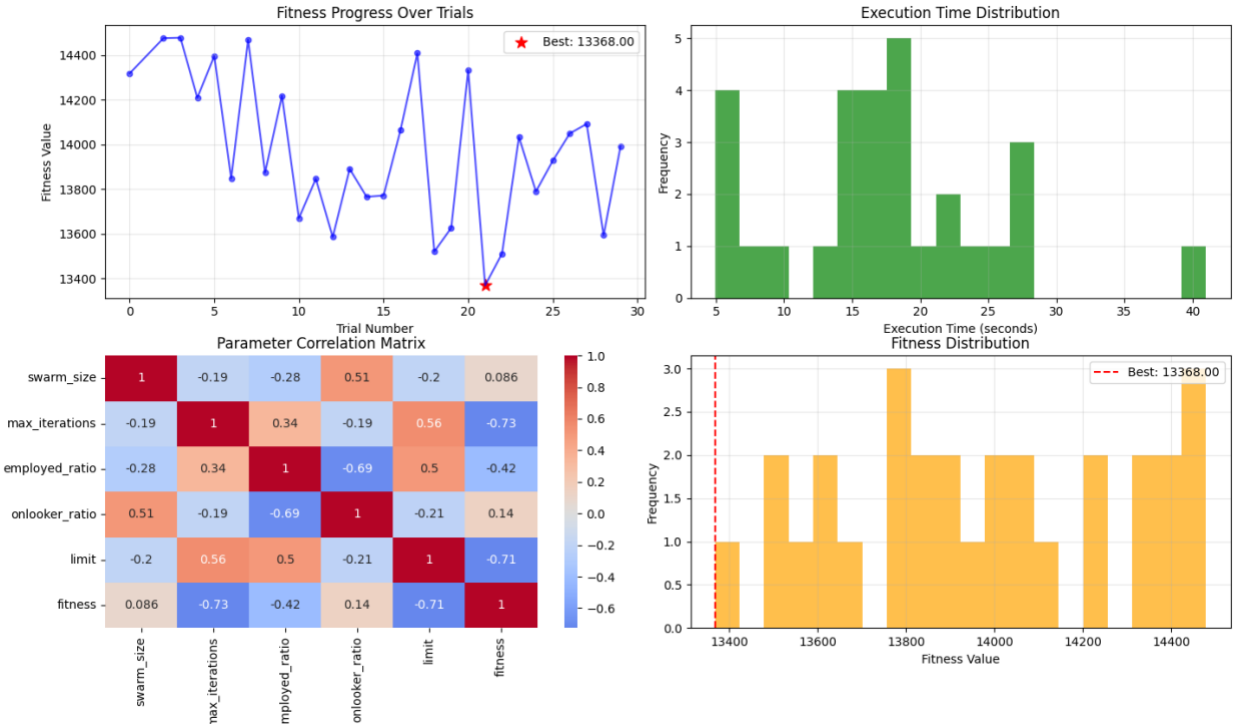
Vehicle S2: S2 → C38 → C28 → C12 → C25 → C14 → C30 → C21 → C29 → C24 → C3 → C31 → C7 → C9 → C26 → C13 → C34 → C41 → C36 → C37 → C39 → C8 → C18 → C16 → C35 → S2 (24 customers)

✅ New best found in trial 19: 13037.00 (improvement: 346.00)

Trial 20/30 - Fitness: 13037.00 - Time: 28.7s - Valid: True

[I 2025-08-19 12:37:47,765] Trial 19 finished with value: 13037.0 and parameters: {'swarm_size': 100, 'max_iterations': 550, 'employed_ratio': 0.4, 'onlooker_ratio': 0.2, 'limit': 35}. Best is trial 19 with value: 13037.0.

For tpe



OPTIMIZATION COMPLETED

==========

Total time: 507.67 seconds

Best fitness: 13368.00

Best parameters:

employed_ratio: 0.6000000000000001

onlooker_ratio: 0.25

swarm_size: 80

max_iterations: 325

limit: 23

Best trial: 21. Best value: 13368: 100%|██████████| 30/30 [08:27<00:00, 16.92s/it]

Parameter Importance:

max_iterations: 0.7727

limit: 0.1379

employed_ratio: 0.0503

swarm_size: 0.0392

Optimization completed in 17.78s

Final best fitness: 13368.00

Total improvements: 44

🏆 FINAL BEST SOLUTION ROUTES:

🚚 Primary Vehicle Routes (Depots → Satellites):

Vehicle D1: D1 → S2 → D1 (1 satellites)

Vehicle D2: D2 → S2 → D2 (1 satellites)

Vehicle D3: D3 → S1 → D3 (1 satellites)

Vehicle D4: D4 → S2 → D4 (1 satellites)

Vehicle D5: D5 → S2 → D5 (1 satellites)

Vehicle D6: D6 → S1 → D6 (1 satellites)

Vehicle D7: D7 → S1 → D7 (1 satellites)

Vehicle D8: D8 → S2 → D8 (1 satellites)

Vehicle D9: D9 → S2 → D9 (1 satellites)

Vehicle D10: D10 → S2 → D10 (1 satellites)

🚙 Secondary Vehicle Routes (Satellites → Customers):

Vehicle S1: S1 → C22 → C11 → C30 → C40 → C38 → C3 → C31 → C23 → C16 → C5 → C35 → C37 → C34 → C9 → S1 (14 customers)

Vehicle S2: S2 → C20 → C12 → C24 → C7 → C32 → C8 → C2 → C17 → C6 → C29 → C4 → C28 → C21 → C26 → C15 → C36 → C41 → C14 → C33 → C1 → C19 → C25 → C10 → C27 → C18 → C13 → C39 → S2 (27 customers)

✅ New best found in trial 21: 13368.00 (improvement: 152.00)

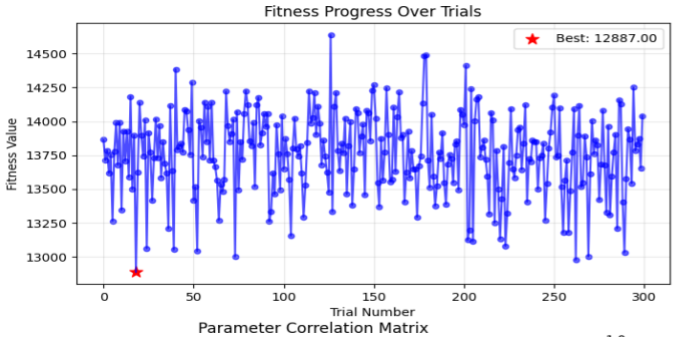
Trial 22/30 - Fitness: 13368.00 - Time: 17.9s - Valid: True

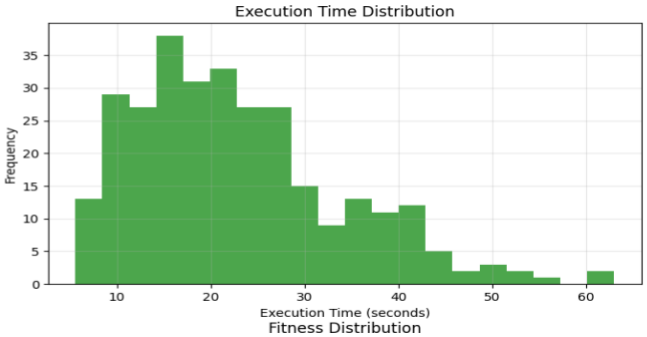
[I 2025-08-19 14:03:48,804] Trial 21 finished with value: 13368.0 and parameters: {'employed_ratio': 0.6000000000000001, 'onlooker_ratio': 0.25, 'swarm_size': 80, 'max_iterations': 325, 'limit': 23}. Best is trial 21 with value: 13368.0.

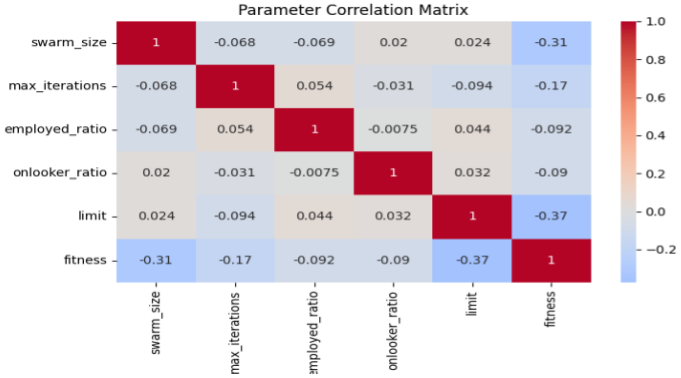
Fitness Progress Pattern:

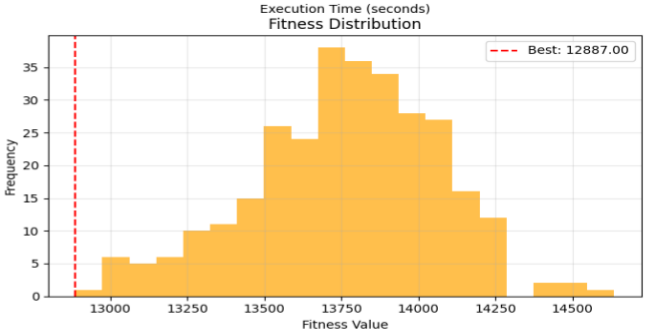
- Erratic, random-looking behavior - fitness jumps between ~13000-14200 with no pattern
- No learning curve - the algorithm doesn't "get smarter" over time
- Systematic but blind sampling - tests combinations in predetermined order
- Best fitness found by chance at trial ~20 (13037)

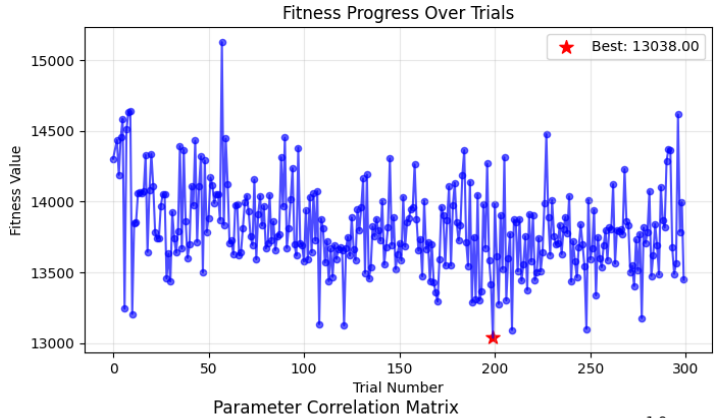
"Figure X clearly illustrates TPE's adaptive learning behavior compared to Grid Search's systematic but uninformed sampling. While Grid Search exhibits erratic fitness values with no improvement pattern over 30 trials, TPE demonstrates classic Bayesian optimization behavior: initial exploration followed by convergence toward optimal parameter regions. This learning capability enables TPE to achieve consistently better results as trial budgets increase, whereas Grid Search performance remains dependent on the predetermined parameter sequence."

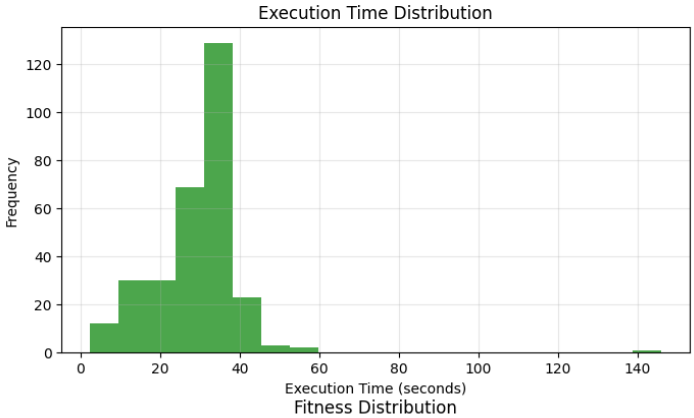

Fitness Progress Over Trials
Best: 12887.00
Fitness Value
Trial Number

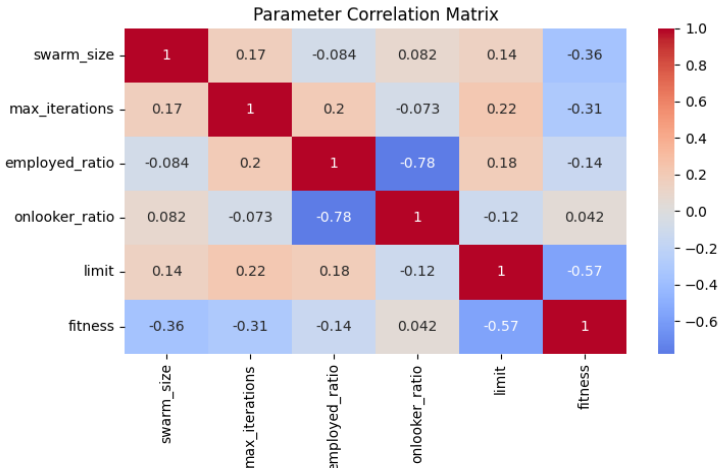

Execution Time Distribution
Frequency
Execution Time (seconds)

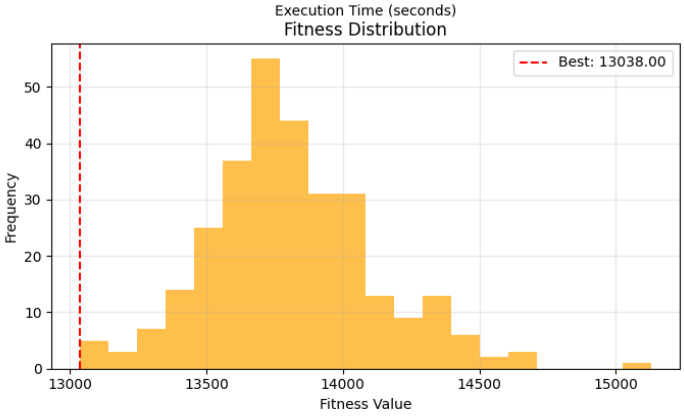

Parameter Correlation Matrix
swarm_size
max_iterations
employed_ratio
onlooker_ratio
limit
fitness

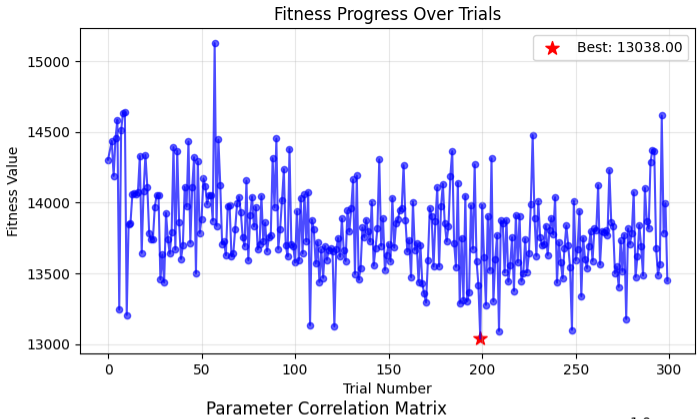

Fitness Distribution
Best: 12887.00
Frequency
Fitness Value

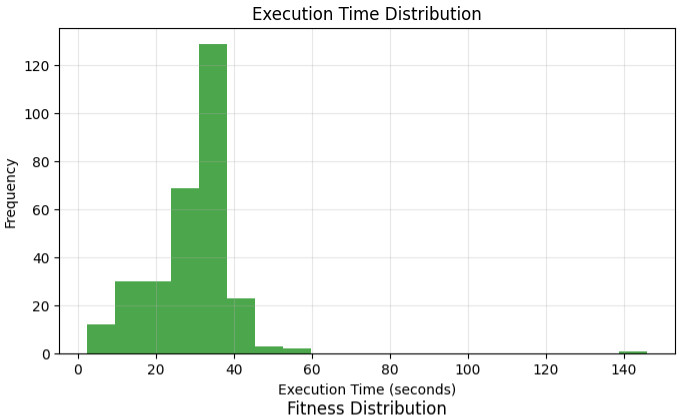

Fitness Progress Over Trials
Best: 13038.00
Fitness Value
Trial Number

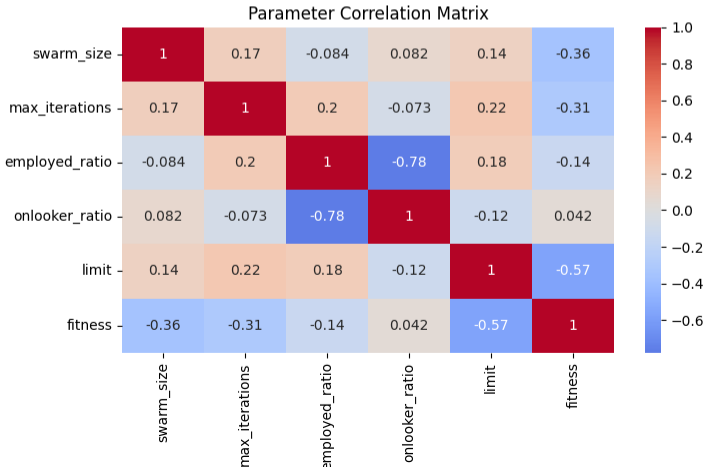

Execution Time Distribution
Frequency
Execution Time (seconds)

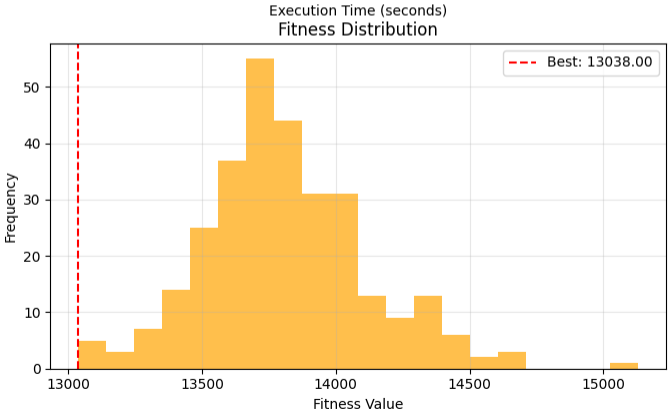

Parameter Correlation Matrix
swarm_size
max_iterations
employed_ratio
onlooker_ratio
limit
fitness


Fitness Distribution
Best: 13038.00
Frequency
Fitness Value









For grid search

Total time: 6888.29 seconds

Best fitness: 12887.00

Best parameters:

swarm_size: 60

max_iterations: 450

employed_ratio: 0.35

onlooker_ratio: 0.25

limit: 25

Parameter Importance:

limit: 0.4645

swarm_size: 0.2234

max_iterations: 0.1250

onlooker_ratio: 0.1090

employed_ratio: 0.0781

for tpe

==========

OPTIMIZATION COMPLETED

==========

Total time: 8775.34 seconds

Best fitness: 13038.00

Best parameters:

employed_ratio: 0.55

onlooker_ratio: 0.25

swarm_size: 130

max_iterations: 400

limit: 25

Parameter Importance:

limit: 0.7615

swarm_size: 0.1370

max_iterations: 0.0876

employed_ratio: 0.0139