(GP4ESP) - **A hybrid Genetic algorithm and Particle swarm optimization algorithm for edge server placement**

HSE University

Department of Computer Science

Date: 15th December 2025

Problem Statement

Edge Computing:

Essential for ultra-low latency services required by smart devices and IoT applications

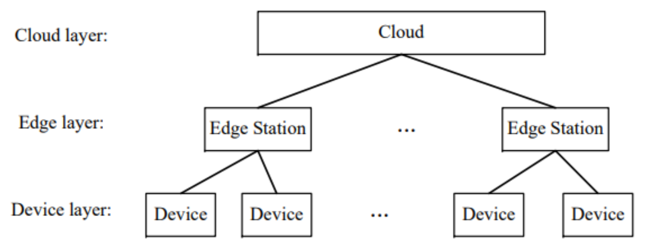
Challenges:

Optimal placement of limited edge servers to efficiently manage request processing

ESP is an NP-hard problem, complex to solve optimally for large systems

Motivation

- Increasing number of connected devices
- Need for effective handling of computational delays at edge servers
- Limitations of current algorithms:
  - Single meta-heuristic methods (limited effectiveness)
  - Inefficient hybridization



Project Goals

Develop	Develop a hybrid algorithm combining Genetic Algorithm (GA) and Particle Swarm Optimization (PSO)
Minimize	Minimize the overall average response time of requests
Enhance	Enhance effectiveness by integrating the strengths of GA (exploration) and PSO (exploitation)

Methodology

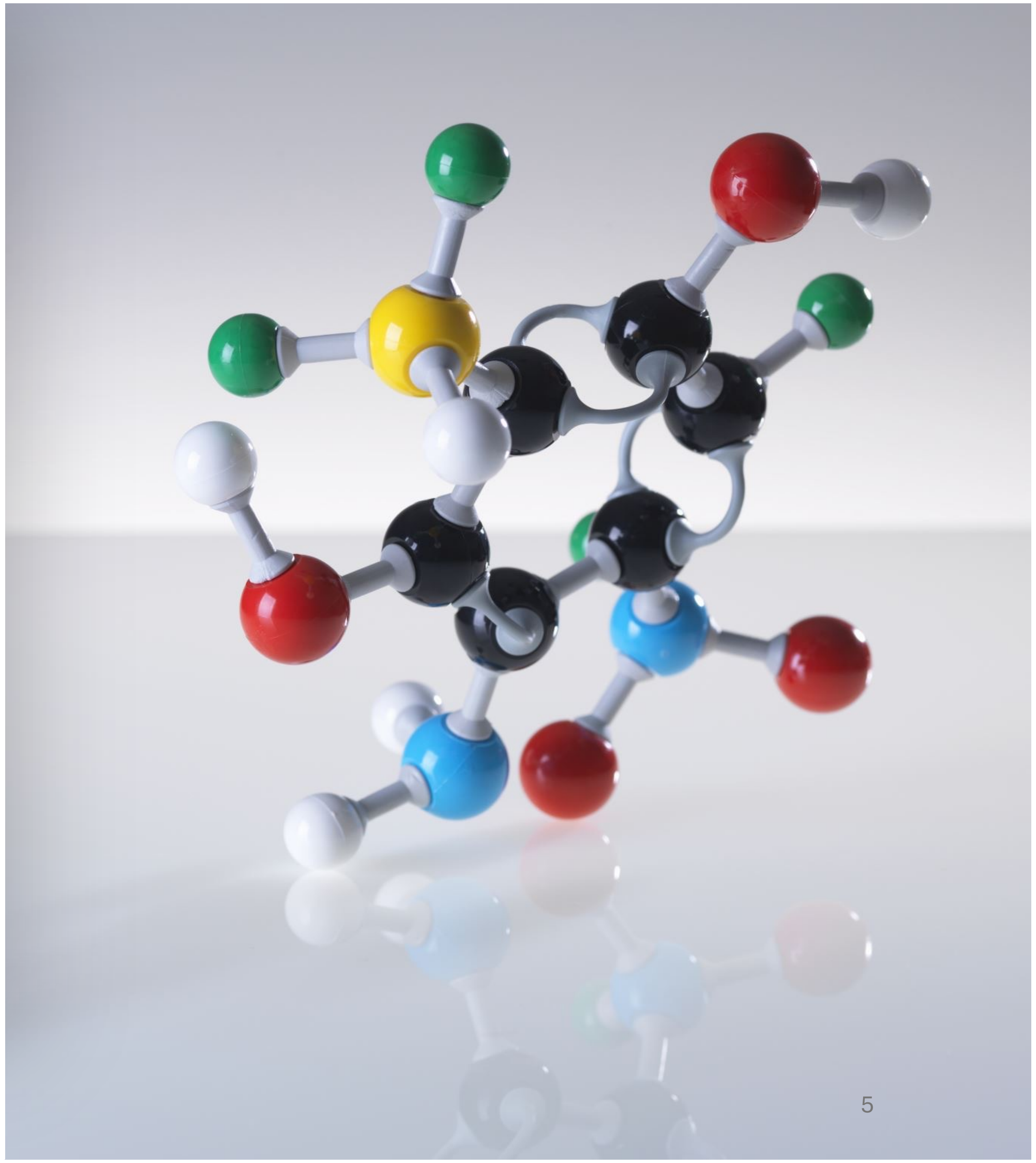
Hybrid Algorithm (GP4ESP)

Genetic Algorithm (GA):

- Uses evolutionary operations such as crossover (combining two parent solutions), mutation (randomly altering parts of a solution), and selection (choosing the best solutions)
- Excellent for exploration but slower convergence

Particle Swarm Optimization (PSO):

- Particles update their positions based on both individual knowledge (personal best solution) and social knowledge (global best solution)
- Fast convergence but can easily become trapped in local optima



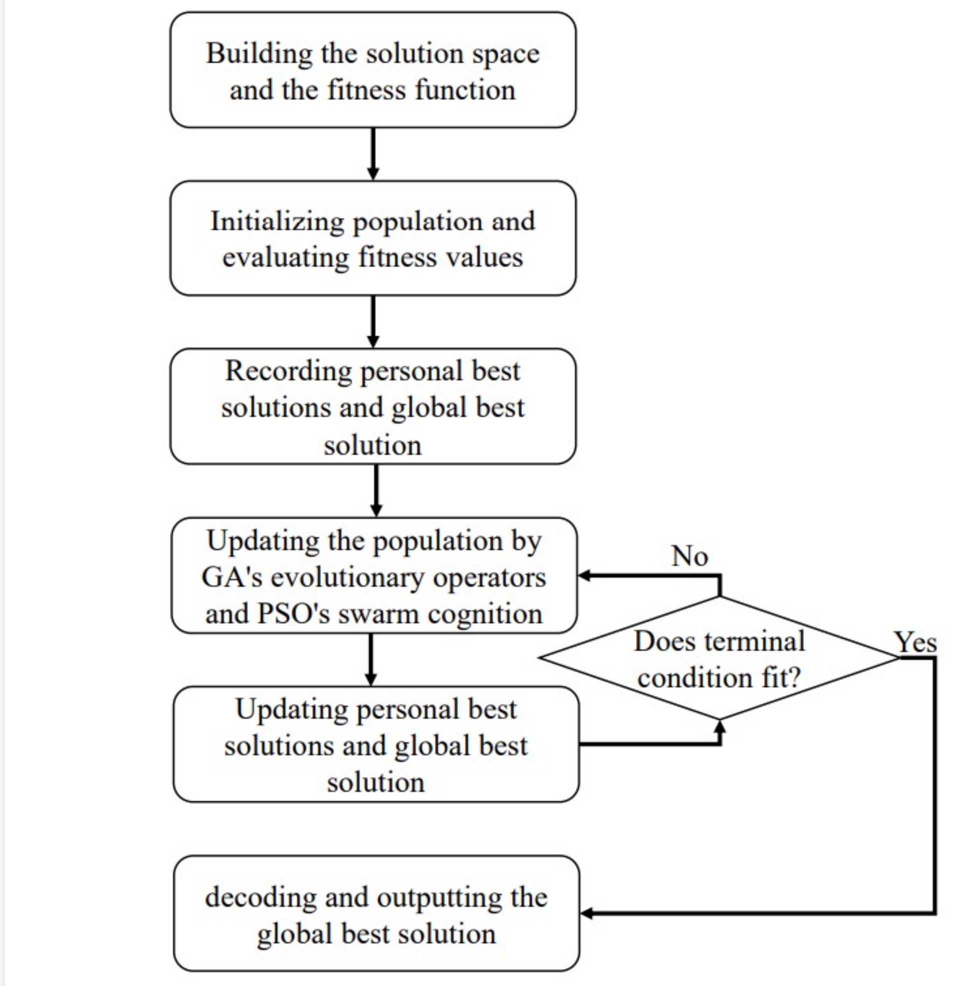
Methodology

Hybrid Algorithm (GP4ESP)

GP4ESP Hybridization Strategy:

- Combines GA's robust exploration with PSO's efficient exploitation
- Each individual solution undergoes:
  - GA crossover with random individuals
  - PSO cognitive crossover using personal best solutions
  - PSO social crossover using global best solutions
- Mutation applied to maintain diversity
- Uses elitism to ensure best solutions are retained

The framework of hybrid meta-heuristic algorithm for edge server placement



Building the solution space and the fitness function
Initializing population and evaluating fitness values
Recording personal best solutions and global best solution
Updating the population by GA's evolutionary operators and PSO's swarm cognition
Updating personal best solutions and global best solution
Does terminal condition fit?
No
Yes
decoding and outputting the global best solution

Experimental Setup

Number of base stations: 1000

Number of edge servers: 600

Load: 0.5

Cloud delay: 50ms

Algorithm parameters:

Population size: 100

Iterations: 100

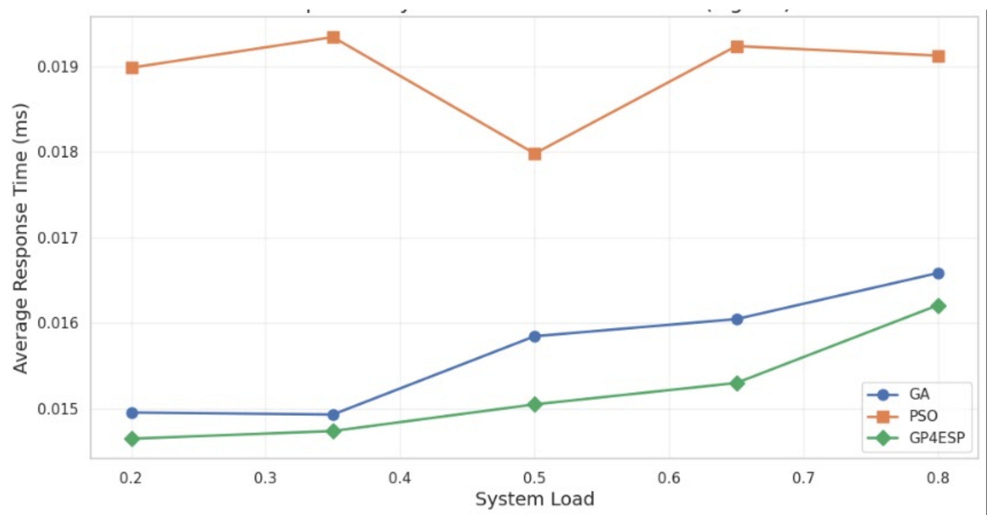
Trials per algorithm: 11

Results

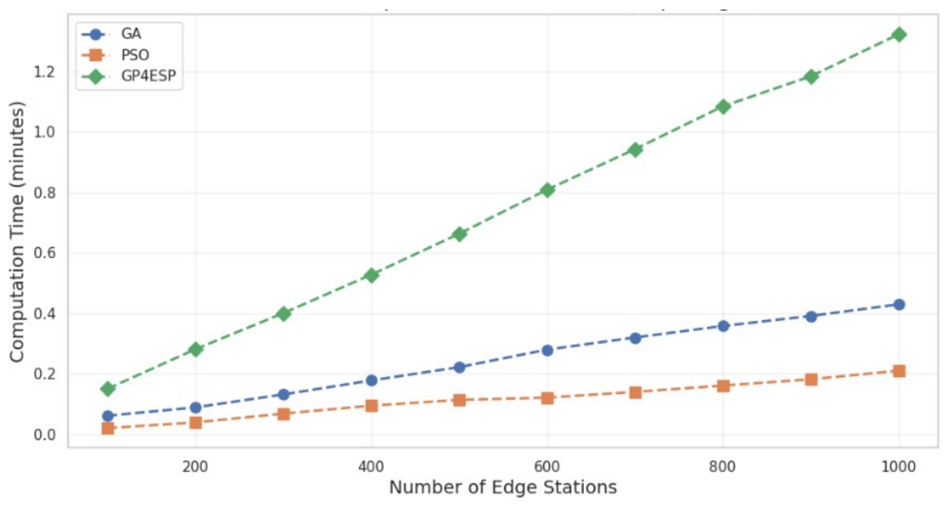
GP4ESP achieved 18.2%–20.7% improvement in response time over GA, PSO, and other algorithms

Stable performance across various problem scales

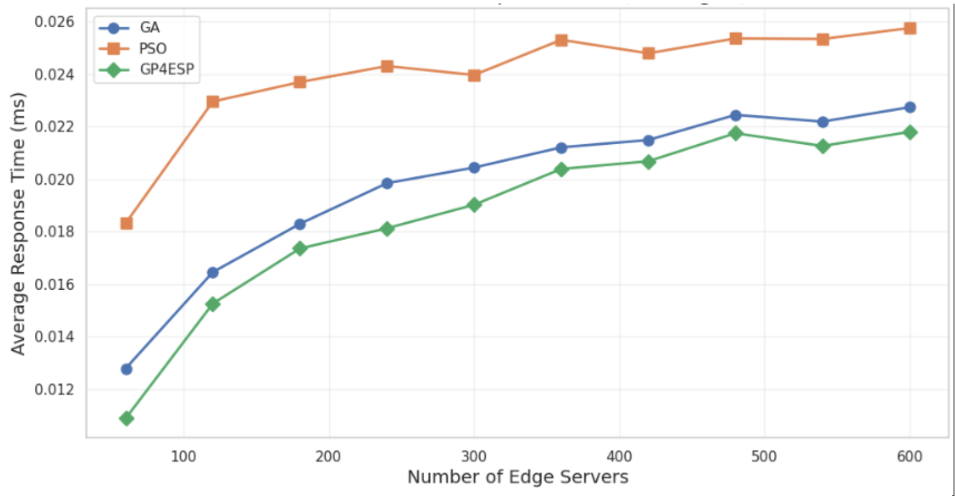
Statistical significance verified (p-value < 0.01)

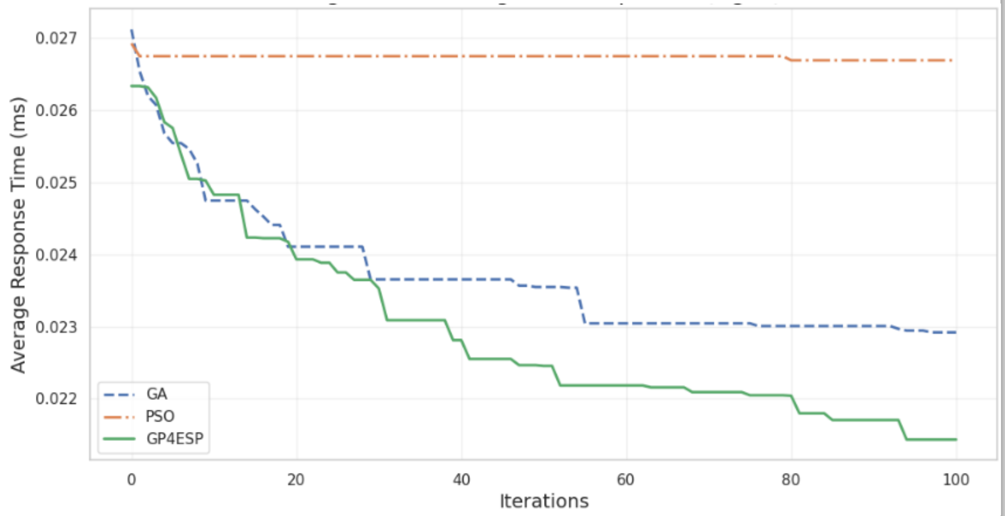


This graph compares average response times of GA, PSO, and GP4ESP algorithms under varying system loads, clearly showing GP4ESP consistently achieves the lowest response times.



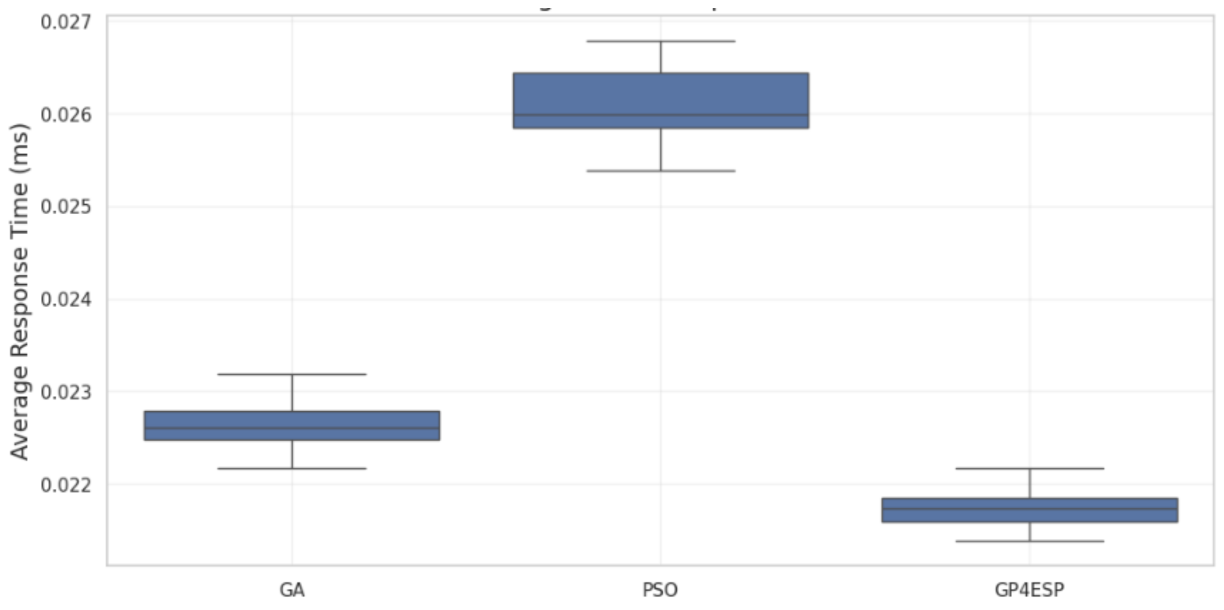
This graph illustrates the computation time required by GA, PSO, and GP4ESP algorithms, showing GP4ESP demands the highest computation time as the number of edge stations increases.





This graph compares average response times of GA, PSO, and GP4ESP algorithms, demonstrating GP4ESP consistently achieves the lowest response times as the number of edge servers increases.

This graph shows the convergence behavior of GA, PSO, and GP4ESP algorithms across iterations, highlighting GP4ESP's superior convergence to lower response times.



This graph presents a boxplot comparison of GA, PSO, and GP4ESP algorithms, clearly indicating GP4ESP achieves the lowest average response times with minimal variability.

Future Work

Dynamic Load Balancing:

Develop a strategy to dynamically redistribute requests among edge servers based on real-time loads and performance metrics

Evaluate impact on overall system latency and server utilization

Inter-station Cooperation:

Enable edge stations to cooperate in processing tasks during peak times

Investigate optimal communication protocols and efficiency trade-offs

Real-world Data Integration:

Utilize actual datasets from real IoT deployments and edge computing scenarios

Validate and enhance the practicality of GP4ESP

Parallel Processing Enhancements:

Leverage parallel computing techniques (e.g., multi-threading, GPU computing) to speed up algorithm execution

Robustness Analysis:

Perform robustness tests against variations in network conditions and server performance

Evaluate algorithm adaptability under different operational environments

Thank You