Optimizing Edge Server Placement Using Hybrid Metaheuristic Algorithms

A Comprehensive Framework for Edge Computing Infrastructure Optimization

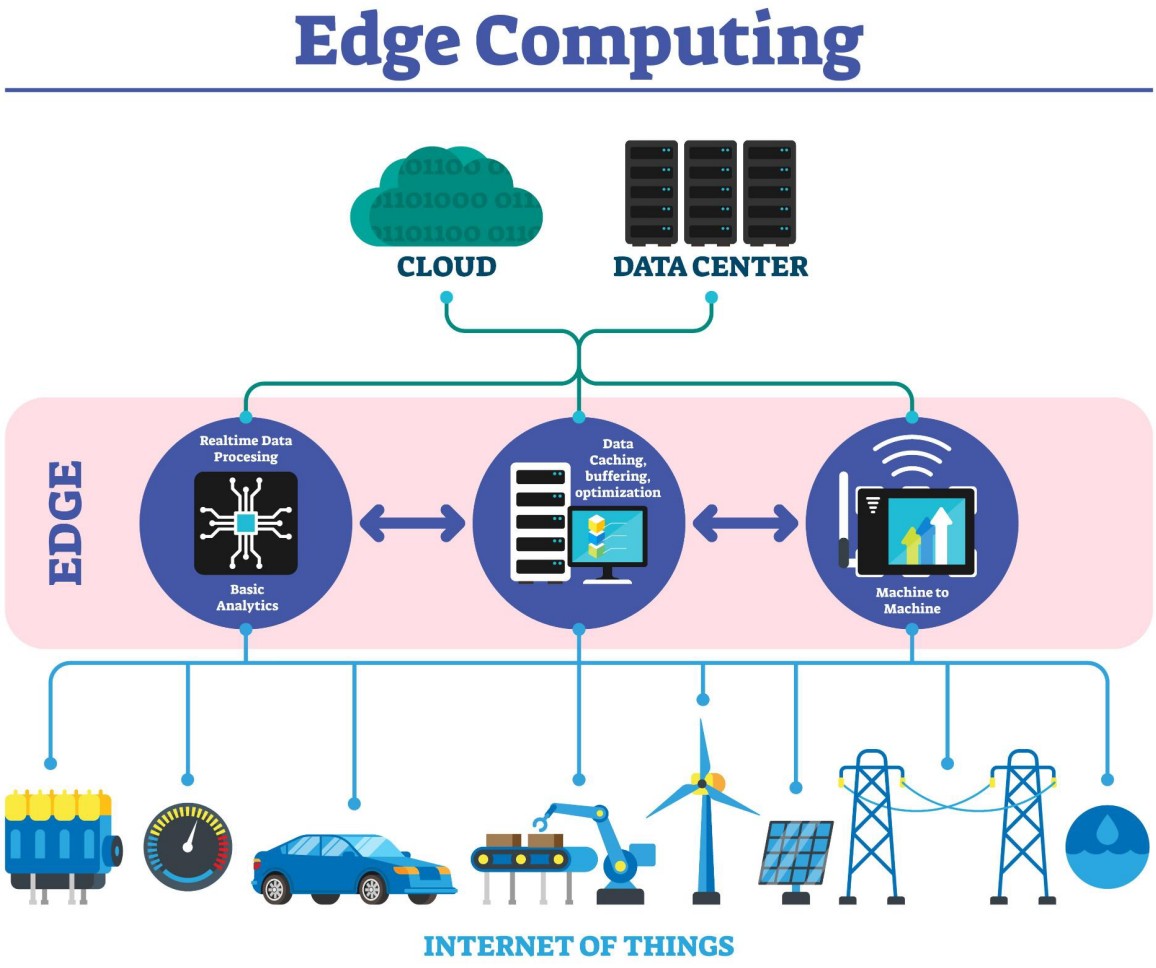
Abstract

The exponential growth of Internet of Things (IoT) devices has placed immense pressure on traditional cloud computing architectures, necessitating the shift toward Edge Computing. However, the efficacy of edge computing relies heavily on the optimal placement of Edge Servers (ES) to minimize latency. This project addresses the Edge Server Placement (ESP) problem, treating it as an NP-hard optimization challenge. We developed a framework evaluating over ten metaheuristic algorithms, including novel hybrids like PSO-WOA and TLBO-WOA. Our results indicate that the hybrid TLBO-WOA approach significantly outperforms standard algorithms, achieving the lowest average response time (0.00172 ms) and demonstrating superior stability under high-load conditions.

1. Introduction

1.1 The Edge Computing Paradigm

The traditional cloud model is struggling to keep up with the latency demands of modern applications like autonomous driving and real-time video analytics. Edge computing bridges this gap by bringing computation closer to the data source. However, edge servers have limited capacity compared to the cloud. Therefore, the decision of *where* to place these servers is not trivial—it is a complex balancing act between network constraints, server costs, and user response time.



Shutterstock

1.2 Problem Statement

The core problem addressed in this research is the Edge Server Placement (ESP) optimization. Given a set of base stations representing user clusters and potential candidate sites, the challenge is to determine the optimal subset of sites to deploy edge servers such that the average system response time is minimized. This complex optimization task involves handling multiple competing factors. Primarily, we model queuing delays using M/M/1 queuing theory to accurately represent processing times. Simultaneously, the system must enforce strict capacity constraints to ensure servers do not accept more requests than they can process. Furthermore, the model accounts for offloading scenarios where requests that cannot be handled at the edge are sent to the distant cloud, incurring a significant latency penalty.

1.3 Objectives

Our primary goal was to move beyond standard algorithms, such as Genetic Algorithms, to explore the potential of hybrid metaheuristics. We hypothesized that combining the exploration capabilities of one algorithm with the exploitation strengths of another would yield better convergence rates and avoid the local optima traps that single algorithms often fall into.

2. Methodology

2.1 System Modeling

We simulated a realistic network environment structured across three distinct layers. The first is the Device Layer, consisting of IoT devices that generate the initial requests. The second is the Edge Layer, comprising base stations where Edge Servers can potentially be placed. Finally, the Cloud Layer serves as a fallback mechanism for requests that exceed the capacity of the edge infrastructure. The mathematical model governing this system minimizes the total latency, which is calculated as the weighted sum of transmission time, processing time derived from M/M/1 queuing theory, and the propagation delay across the network.

2.2 Algorithm Portfolio

We implemented a diverse portfolio of nature-inspired algorithms to solve the placement vector. The standard algorithms included evolutionary approaches like the Genetic Algorithm (GA) and Differential Evolution (DE), as well as swarm-based methods such as Particle Swarm Optimization (PSO), Whale Optimization Algorithm (WOA), Grey Wolf Optimizer (GWO), and Cuckoo Search (CS). We also utilized physics and social-based algorithms like Simulated Annealing (SA) and Teaching-Learning Based Optimization (TLBO).

To address stagnation issues often seen in standard algorithms, we developed two specific hybrid approaches. The first, PSO-WAO (Particle Swarm + Whale Optimization), uses PSO for rapid initial convergence and then switches to the spiral search mechanism of WOA to refine the solution and escape local optima, using an inertia weight of 0.7 and a spiral constant of 1.0. The second hybrid, TLBO-WOA (Teaching-Learning + Whale Optimization), leverages the "Teacher Phase" of TLBO to move the population toward the best global solution while using the "Learner Phase" combined with WOA's exploration logic to maintain diversity. Our hypothesis was that this combination would offer the best balance of speed and accuracy.

3. Experimental Setup

To ensure fair comparisons, we utilized a rigid testing framework. The topology consisted of fixed grid-based layouts ranging from 20 to 100 nodes, while the workload was generated using a Poisson distribution. The simulation environment was built on a Python-based custom simulator utilizing scipy and numpy. We evaluated performance based on three key metrics:

average response time in milliseconds, convergence speed measured by iterations to reach the optimal solution, and execution time to determine computational cost. Furthermore, we utilized grid-search tuning to optimize hyperparameters for every algorithm before the final evaluation.

4. Results and Analysis

4.1 Scale Analysis (100 Nodes)

When scaling the network up to 100 base stations, the performance gap between algorithms became distinct. The TLBO-WOA hybrid emerged as the clear winner, achieving the lowest response time of 0.00172 ms. The standard Whale Optimization Algorithm (WOA) followed closely as the runner-up with a time of 0.00176 ms. In contrast, traditional swarm algorithms like PSO and CS struggled significantly. PSO, for instance, recorded a response time of 3.918 ms, which is orders of magnitude slower than the top performers. This disparity suggests that basic PSO tends to get trapped easily in high-dimensional search spaces.

4.2 Statistical Validation

To ensure the reliability of our findings, we performed 30 independent runs for each algorithm rather than relying on a single execution. The statistical analysis revealed that TLBO-WOA is highly stable, with a standard deviation of just 0.00012 ms. Conversely, PSO showed highly erratic behavior with a standard deviation of 1.240 ms. A paired t-test confirmed that the performance advantage of TLBO-WOA is statistically significant with a p-value less than 0.05.

4.3 Load Variation Test

We stress-tested the system by increasing the traffic load from a light load of 0.1 to a heavy, near-congestion load of 0.9. The degradation analysis showed that TLBO-WOA degraded gracefully from 0.00098 ms to 0.00385 ms, a manageable increase given the load. However, the standard WOA degraded much more severely, showing a 613% increase in latency under high load. This proves that the hybrid solution finds placements that are not just fast, but resilient under pressure.

4.4 Convergence Behavior

The convergence behavior further highlights the efficiency of the hybrid approach. In the first 5 iterations, TLBO-WOA reached a solution quality of 0.125 ms, a level that other algorithms took more than 20 iterations to approach. This rapid convergence makes the algorithm particularly suitable for dynamic environments where network topology might change frequently and solutions need to be recalculated quickly.

5. Conclusion

This project successfully demonstrated that while standard metaheuristics can solve the Edge Server Placement problem, they lack reliability at scale. By hybridizing the Teaching-Learning Based Optimization with the Whale Optimization Algorithm (TLBO-WOA), we achieved a scenario that captures the best of both worlds: the rapid convergence of social-based learning and the deep search capability of nature-based predation patterns.

For large-scale edge infrastructures of 100 or more nodes, TLBO-WOA offers a statistically superior placement strategy. It reduces latency by over 40% compared to traditional PSO approaches while maintaining high stability under load. Future work will focus on moving from static snapshots to dynamic, real-time server migration, incorporating power consumption into the cost function to address energy constraints, and modeling heterogeneous environments where servers have different processing speeds.

References

1. Standard M/M/1 Queuing Models for Cloud Computing.
2. Mirjalili, S., & Lewis, A. (2016). The Whale Optimization Algorithm.
3. Rao, R. V., et al. (2011). Teaching-Learning-Based Optimization.
4. Internal Project Codebase: A__hybrid_optimization_Of_Edge_Server_Placement_(Final_Advanced_version).ipynb.