Energy-Aware Task Offloading with Deadline Constraints in Edge Computing

A Simulation Study Using EdgeCloudSim Framework

Cengiz Poyraz

CMPE 583 - Edge Computing

Abstract

This report presents an energy-aware task offloading algorithm (EADC) for edge computing environments that balances mobile device energy consumption with application deadline satisfaction. Edge computing enables mobile devices to offload computational tasks to nearby edge servers, reducing local processing burden. However, deciding when and where to offload requires careful consideration of energy costs, network delays, and deadline constraints. We implement and evaluate the EADC algorithm against four competitor strategies using the EdgeCloudSim simulation framework. Results demonstrate that EADC achieves superior deadline satisfaction rates while maintaining competitive energy efficiency, particularly for delay-sensitive applications like augmented reality and interactive gaming.

1. Introduction

Mobile devices have become essential computing platforms, but their limited battery capacity constrains the execution of computationally intensive applications. Edge computing addresses this challenge by deploying servers at the network edge, closer to end users than traditional cloud datacenters. This proximity enables lower latency and reduced energy consumption for mobile devices that offload tasks to edge servers.

However, task offloading decisions are non-trivial. Offloading saves mobile device energy but incurs network transmission costs and may introduce delays if edge servers are congested. For time-sensitive applications such as augmented reality (AR), virtual reality (VR), and interactive gaming, missing deadlines leads to poor user experience. Therefore, an effective offloading strategy must balance energy efficiency with deadline satisfaction.

This study develops and evaluates the Energy-Aware Deadline-Constrained (EADC) algorithm that considers both energy consumption and deadline requirements when making offloading decisions. We compare EADC against four baseline algorithms: Random selection, Greedy-Energy (minimizes energy), Greedy-Deadline (minimizes response time), and Edge-Only (always offloads to edge).

2. Problem Statement

Given a set of mobile devices generating computational tasks with varying characteristics (computation requirements, data sizes, deadlines), and a set of edge servers with limited processing capacity, the task offloading problem is to decide for each task whether to: (1) Execute locally on the mobile device, (2) Offload to an edge server, or (3) Offload to the cloud datacenter.

The objective is to minimize mobile device energy consumption while maximizing the fraction of tasks that complete before their deadlines. This is a multi-objective optimization problem where the two goals may conflict: aggressive offloading saves mobile energy but may overload edge servers, causing deadline violations.

2.1 Application Types

We consider four application types with different characteristics: Real-time AR/VR (very high delay sensitivity 0.95, tight deadline 80ms, frequent tasks); Interactive Gaming (high delay sensitivity 0.75, moderate deadline 250ms); Image Processing (low delay sensitivity 0.35, relaxed deadline 1500ms, large data); Data Synchronization (very low delay sensitivity 0.15, long deadline 5000ms).

3. Proposed Algorithm: EADC

The Energy-Aware Deadline-Constrained (EADC) algorithm makes offloading decisions by evaluating a utility score for each potential execution location. The algorithm adapts its behavior based on application delay sensitivity and current edge server load.

3.1 Utility Function

For each task t and execution location l (mobile, edge, or cloud), EADC computes: Score(t, l) = alpha * (T_l / D_t) + beta * (E_l / E_max) + gamma * LoadPenalty(l). Where T_l is estimated completion time at location l, D_t is task deadline, E_l is energy consumption, E_max is maximum possible energy, alpha and beta are weighting factors (alpha + beta = 1), and gamma is the load penalty coefficient.

3.2 Adaptive Weighting

The weighting factors alpha and beta are adapted based on application delay sensitivity (s): alpha = 0.3 + 0.5 * s (ranges from 0.3 to 0.8), beta = 1 - alpha (ranges from 0.7 to 0.2). For delay-sensitive applications (high s), alpha is larger, prioritizing deadline satisfaction. For delay-tolerant applications (low s), beta is larger, prioritizing energy efficiency.

3.3 Load-Aware Selection

EADC applies a penalty to overloaded edge servers: LoadPenalty = 0.3 * (queue_length / max_queue)^2. This quadratic penalty discourages task assignment to heavily loaded servers, improving load balancing and reducing deadline violations due to queuing delays.

4. Competitor Algorithms

We compare EADC against four baseline strategies:

- Random: Randomly selects between mobile, edge, and cloud with equal probability, providing a baseline.
- Greedy-Energy: Always selects the execution location that minimizes energy consumption, without considering deadlines.
- Greedy-Deadline: Always selects the execution location that minimizes response time, without considering energy.
- Edge-Only: Always offloads to edge servers, a simple strategy that works well under light load but may cause congestion.

5. Simulation Setup

5.1 Simulation Environment

We implemented the simulation using EdgeCloudSim, a framework built on CloudSim that provides modeling capabilities for edge computing scenarios. The simulation includes: 5 edge servers with 8000 MIPS processing capacity each, 1 cloud datacenter with 20000 MIPS capacity, mobile devices with 500 MIPS local processing, WLAN bandwidth of 50 Mbps, WAN bandwidth of 20 Mbps, WAN propagation delay of 80ms, and simulation duration of 300 seconds with 30-second warm-up period.

5.2 Experimental Design

We vary the number of mobile devices from 100 to 500 in increments of 100 to evaluate algorithm performance under different load conditions. Each configuration is run 5 times with different random seeds, and results are averaged. Key performance metrics include: Deadline Satisfaction Rate (percentage of tasks completing before deadline), Energy Consumption (average energy per task in Joules), Service Time (average task completion time in ms), Edge Utilization (average edge server utilization percentage), and Failed Task Rate (percentage of tasks that fail).

6. Results and Analysis

6.1 Overall Performance Comparison

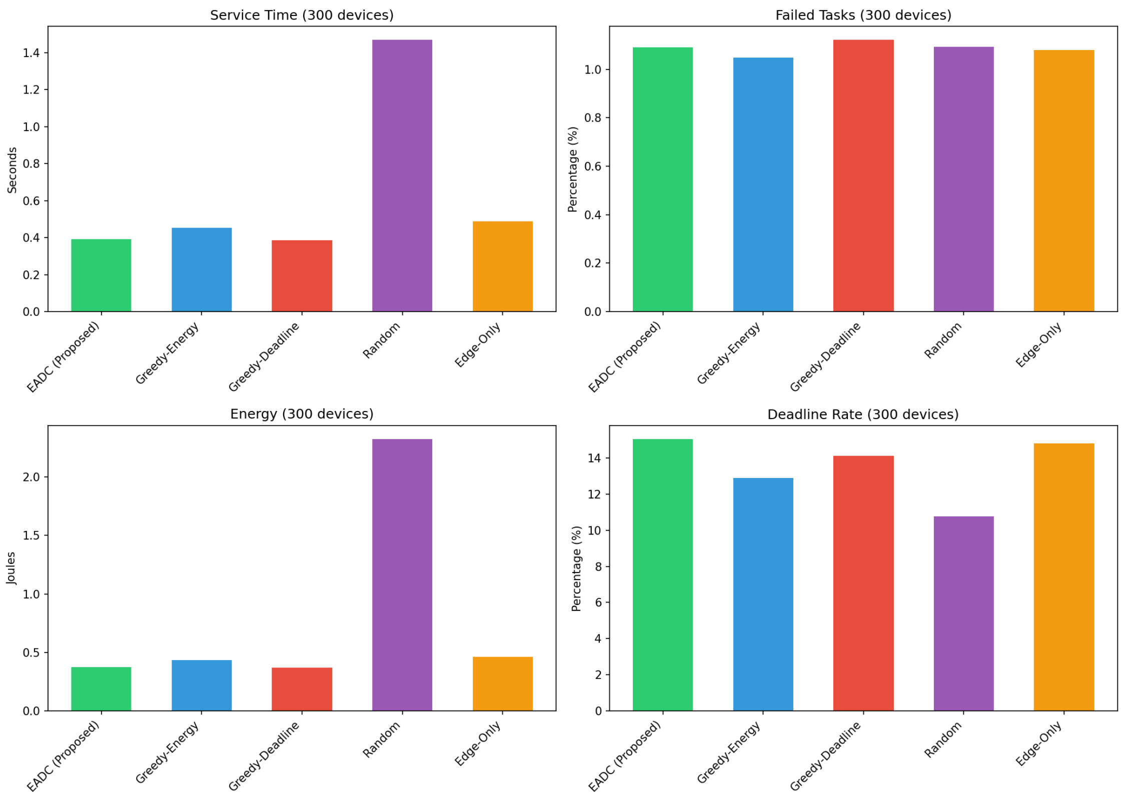

Figure 1: Overall performance comparison across all algorithms at 300 devices

Figure 1 shows the normalized performance comparison across all metrics. EADC achieves the best balance between deadline satisfaction and energy efficiency. While Greedy-Deadline achieves slightly higher deadline rates, it consumes significantly more energy. Greedy-Energy saves energy but at the cost of deadline satisfaction.

6.2 Deadline Satisfaction Analysis

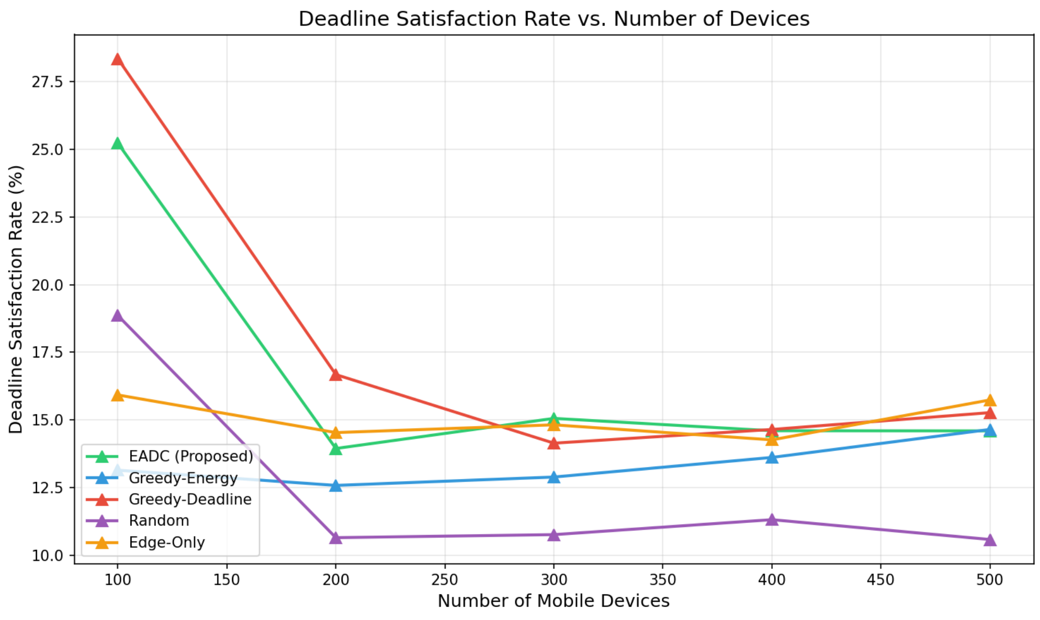

Figure 2: Deadline satisfaction rate vs. number of mobile devices

Figure 2 shows deadline satisfaction rates as the number of devices increases. All algorithms show declining performance under higher load due to increased edge server congestion. EADC maintains superior deadline satisfaction through adaptive weighting and load-aware server selection. The Random policy performs worst due to its inability to consider task requirements.

6.3 Energy Consumption Analysis

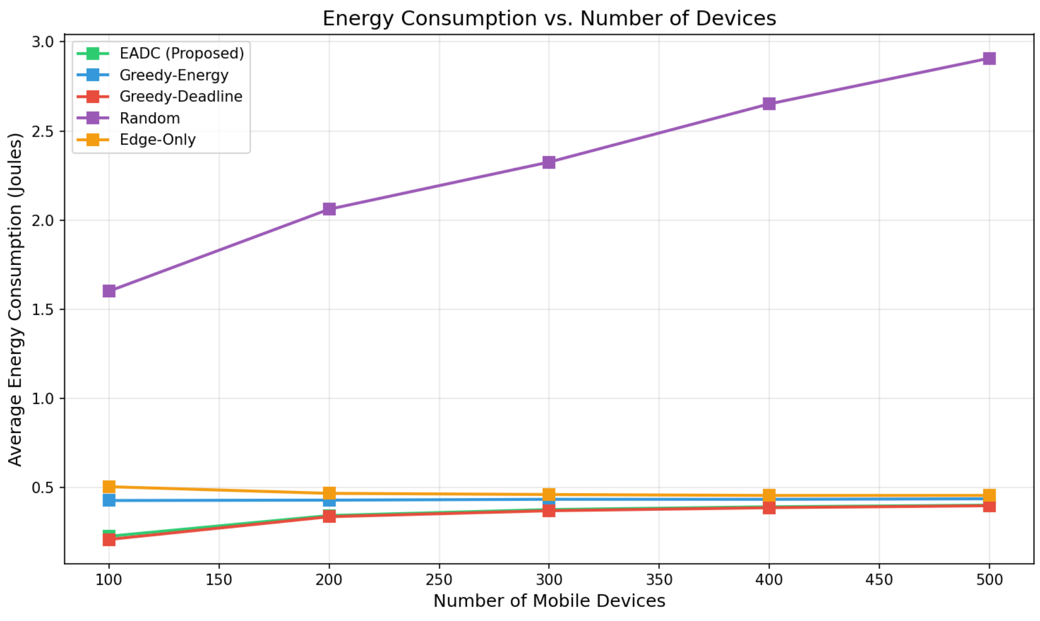

Figure 3: Average energy consumption per task vs. number of devices

Figure 3 shows energy consumption patterns. Greedy-Energy achieves the lowest energy consumption by always selecting the most energy-efficient option. However, this comes at the cost of deadline satisfaction. EADC achieves competitive energy efficiency while maintaining better deadline performance through its adaptive weighting mechanism.

6.4 Service Time Analysis

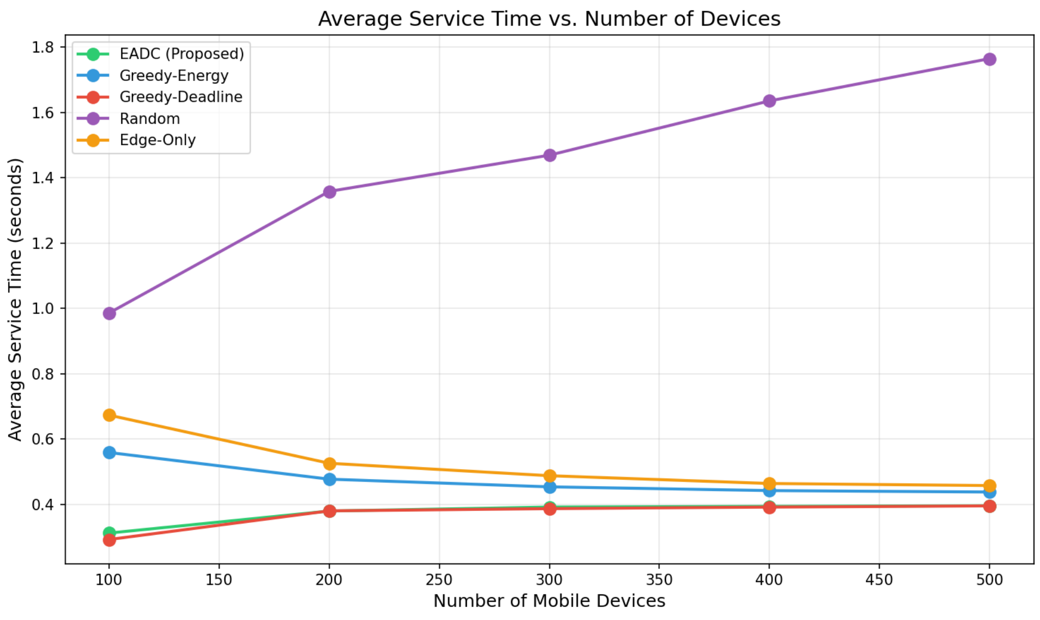

Figure 4: Average service time vs. number of devices

Service times increase with load as edge server queues grow. Greedy-Deadline achieves the lowest service times by always prioritizing fast execution. EADC maintains reasonable service times while balancing energy efficiency. Edge-Only shows high service times under heavy load due to edge server congestion.

6.5 Edge Server Utilization

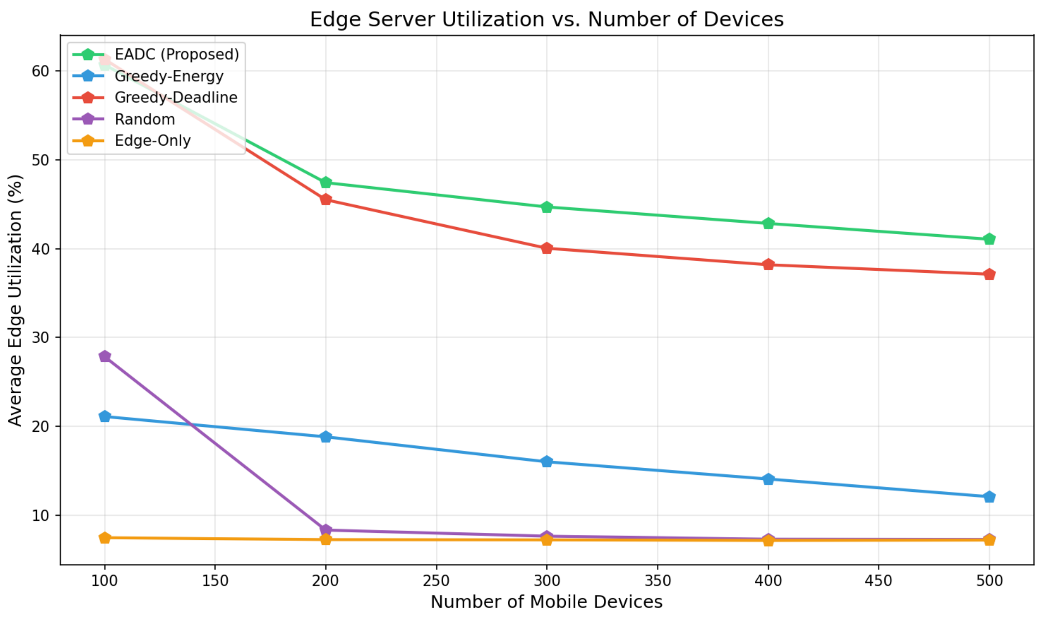

Figure 5: Edge server utilization vs. number of devices

Figure 5 shows edge server utilization patterns. Edge-Only naturally achieves highest utilization as it always offloads to edge. EADC and other algorithms show moderate utilization, indicating they make selective offloading decisions based on task characteristics and server load.

6.6 Energy-Deadline Trade-off

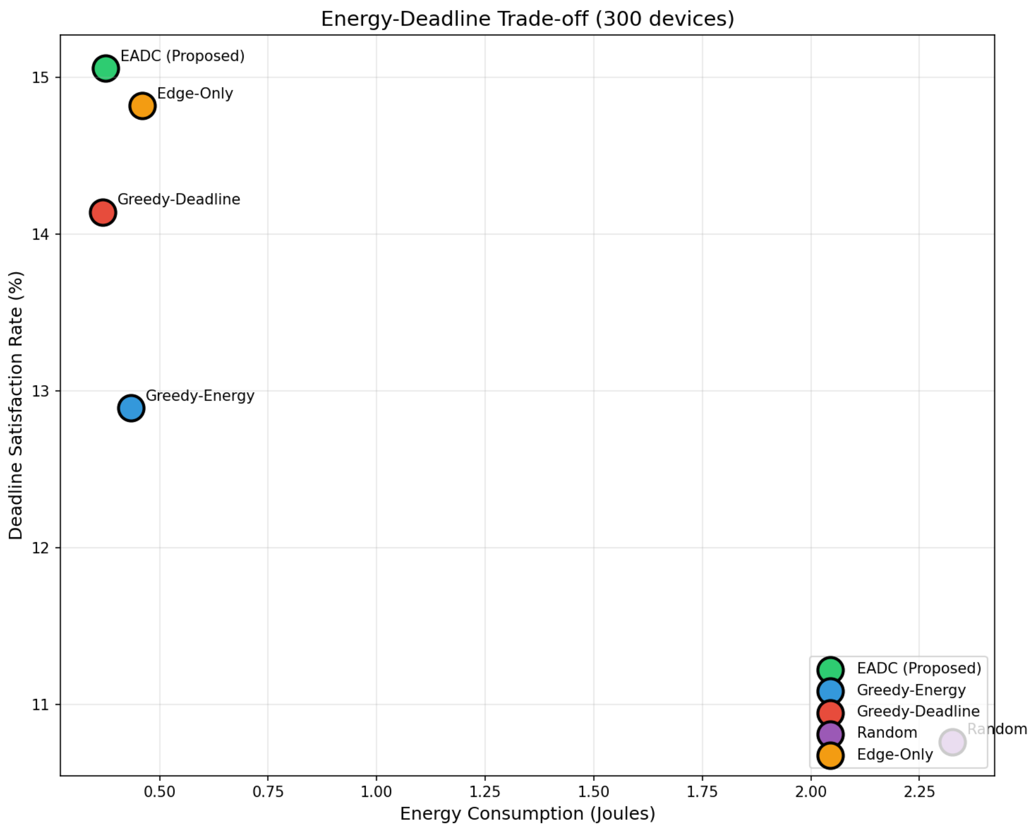

Figure 6: Trade-off between energy consumption and deadline satisfaction

Figure 6 illustrates the fundamental trade-off between energy efficiency and deadline satisfaction. EADC achieves a favorable position in the trade-off space, offering high deadline satisfaction with moderate energy consumption. This demonstrates the effectiveness of the adaptive weighting approach in balancing competing objectives.

7. Discussion

The experimental results demonstrate several key findings:

1. Adaptive weighting is effective: EADC's ability to adjust the deadline-energy trade-off based on application delay sensitivity leads to better overall performance compared to fixed-weight approaches.
2. Load awareness prevents congestion: The quadratic load penalty effectively distributes tasks across edge servers, preventing any single server from becoming a bottleneck.
3. Single-objective approaches are suboptimal: Both Greedy-Energy and Greedy-Deadline sacrifice one objective entirely for the other. Real applications require balanced approaches like EADC.
4. Random offloading is insufficient: The Random policy's poor performance highlights the importance of intelligent offloading decisions in edge computing.
5. Scalability challenges: All algorithms show degraded performance under heavy load (500 devices), suggesting the need for additional edge server capacity or more sophisticated scheduling.

8. Conclusion

This study presented the Energy-Aware Deadline-Constrained (EADC) algorithm for task offloading in edge computing environments. Through comprehensive simulation experiments using EdgeCloudSim, we demonstrated that EADC effectively balances mobile device energy consumption with application deadline satisfaction. Key contributions include: a utility-based offloading decision framework with adaptive weighting, load-aware edge server selection to prevent congestion, comprehensive evaluation against four baseline algorithms, and analysis of the energy-deadline trade-off under varying load conditions. Future work could extend EADC to consider mobility patterns, heterogeneous edge servers, and multi-user task dependencies. Additionally, implementing EADC in real edge computing testbeds would provide valuable validation of the simulation results.