Implementation Report:
NOMA-MEC Based Task Offloading in UAV-assisted IoV Networks

Somesh Waghde (252CS030)
Brijesh Saroj (252CS007)

November 23, 2025

1 Introduction and Problem Statement

The rapid advancement of B5G/6G technologies has created a surge in demand for low-latency and low-energy devices within telematics scenarios. In Internet of Vehicles (IoV) networks, modern vehicles are required to perform computationally intensive tasks (e.g., autonomous driving data processing) [1]. However, the limited computing power of mobile devices is often insufficient to meet these real-time processing requirements.

The core problem addressed in this implementation is the optimization of task offloading in a vehicular environment where:

- Vehicles have limited local processing capacity (C^{loc}).
- Tasks have strict latency constraints (T^{max}).
- Unmanned Aerial Vehicles (UAVs) act as Mobile Edge Computing (MEC) servers.

To address the inefficiency of traditional orthogonal access methods, the system utilizes **Non-Orthogonal Multiple Access (NOMA)**. NOMA allows multiple users to share resource elements, effectively addressing insufficient computing power and improving spectrum utilization. The objective is to minimize the total task processing time under computational and energy constraints.

2 System Model and Proposed Solution

The proposed solution introduces a **UAV-assisted IoV Task Offloading Algorithm (TOA)**. This approach leverages a Genetic Algorithm (GA) to determine the optimal binary offloading strategy for each vehicle.

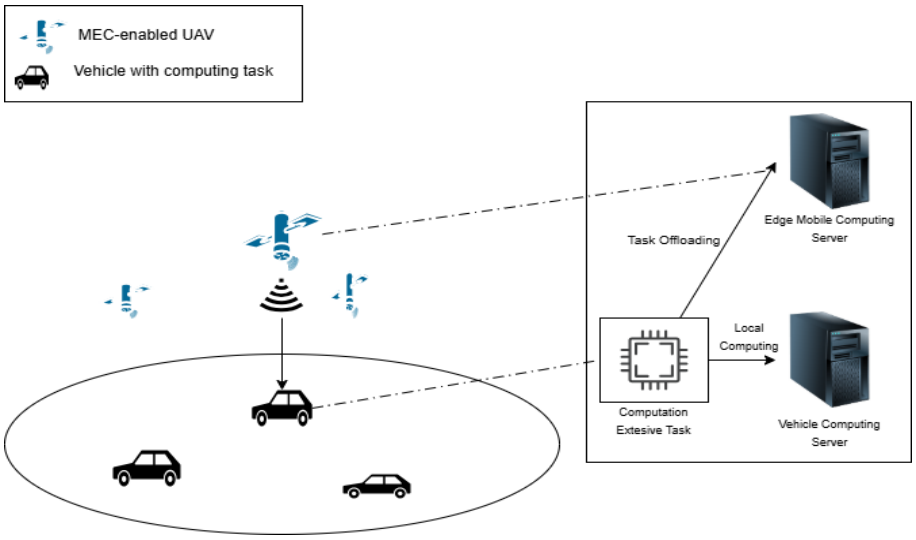

Figure 1: System model showing Vehicles (V) and UAVs acting as Edge Servers using NOMA communication [1].

2.1 Mathematical Framework

The implementation relies on the following mathematical models derived from the source text:

2.1.1 Local Processing Model

If a vehicle j processes a task locally, the processing time T_j^{loc} and energy consumption E_j^{loc} are calculated as:

$$T_j^{loc} = \frac{D_j \omega_j}{C_j^{loc}} \tag{1}$$

$$E_j^{loc} = \alpha D_j \omega_j (C_j^{loc})^2 \tag{2}$$

Where D_j is the input data size, ω_j is the processing density (cycles/bit), and C^{loc} is the local computing capacity (cycles/s).

2.1.2 MEC Offloading Model (NOMA)

When offloading, the vehicle transmits data to a UAV. The transmission rate R_j is determined by the Signal-to-Interference-plus-Noise Ratio (SINR), denoted as $\gamma_{i,j}$. The channel gain $h_{i,j}$ is inversely proportional to the squared distance ($d_{i,j}^2$).

$$R_j = B \log_2(1 + \gamma_{i,j}) \tag{3}$$

The total time for offloading includes transmission time, MEC processing time, and the UAV's flight time to the target position:

$$T_j^{mec} = \frac{D_j}{R_j} + \frac{D_j \omega_j}{f^{mec}} + \frac{d_{i,j}}{v_i} \tag{4}$$

2.2 Optimization Strategy

The problem is modeled as a binary decision process where the decision variable $x_j \in \{0, 1\}$:

- $x_j = 0$: Local execution.
- $x_j = 1$: Offload to UAV.

The TOA utilizes a **Genetic Algorithm (GA)**. The offloading strategy is encoded as a binary gene sequence (e.g., `010101`), where the "fitness" of an individual strategy is defined by the system's total task processing time.

3 Implementation Details

The simulation was implemented using Python, leveraging the `simpy` library for discrete-event simulation and a custom class structure to represent the physical constraints of the IoV network.

3.1 Class Structure

Two primary node classes were defined to mirror the physical entities described in the paper:

1. `VehicleNode`: Encapsulates task parameters.
    - Attributes: Input data size (D_j), processing density (ω_j), max tolerance (T^{max}), and local CPU capability (C^{loc}).
    - *Mapping:* Corresponds to the set B_V and task set B_Q defined in Section II of the source paper.
2. `UAVNode`: Represents the edge server.
    - Attributes: 3D position (L_U), flight speed (ν_i), and transmit power.
    - *Mapping:* Corresponds to set B_U.

3.2 Algorithm Implementation

The GA logic was encapsulated in an optimizer class. The key components of the code map directly to the equations provided in the system model.

3.2.1 NOMA Interference Modeling

A critical part of the implementation is the calculation of Interference ($G_{i,j}$) from other UAVs, which reduces the SINR. The implementation iterates through all UAVs ($x \neq i$) to sum their interference power:

```
# Calculate interference from other UAVs (x) on vehicle j
G_i_j = np.sum(
    [self.p_u_i[x] * (self.h0 / np.linalg.norm(self.U_pos[x] - self.V_pos[j])
    **2)
     for x in range(self.I) if x != i_uav]
)
```

Listing 1: NOMA Interference Calculation Snippet

3.2.2 Fitness Function and Energy Extension

The `fitness_function` method calculates the total delay.

- **Penalty Mechanism:** If a task exceeds T^{max}, a `rejection_penalty` is added to the fitness score to discourage invalid solutions.
- **Energy Extension:** While the primary goal is time minimization, the implementation extends the model by explicitly calculating UAV Operation Energy:

$$E_{UAV} = P_{UAV\_op} \times T_{flight} \tag{5}$$

4 Baseline Comparison

To validate the effectiveness of the proposed GA approach, a baseline script (`UavIoVTask Offloading All Offload`) was implemented.

4.1 The "All-Offload" Strategy

This baseline assumes a naive approach where every vehicle attempts to offload its task to the MEC server ($x_j = 1$ for all j). The paper explicitly compares the TOA against this strategy, noting that as task numbers increase, the all-offload scheme suffers due to UAV flight energy demands and queuing [1].

4.2 Performance Observations

Based on the simulation results as shown in Figure[2] and Figure[3]:

1. **Latency:** The GA approach dynamically retains tasks locally when the transmission channel is poor or when the UAV is too distant. The All-Offload method fails to account for this, resulting in higher average delays.
2. **Energy:** The All-Offload method incurs massive energy penalties because it forces UAV flight and wireless transmission even for small tasks that could be efficiently processed locally.

3. **Scalability:** The GA implementation demonstrates that as the number of tasks increases (e.g., from 10 to 50), the intelligent allocation allows the system to stay within T^{max} constraints significantly more often than the baseline.

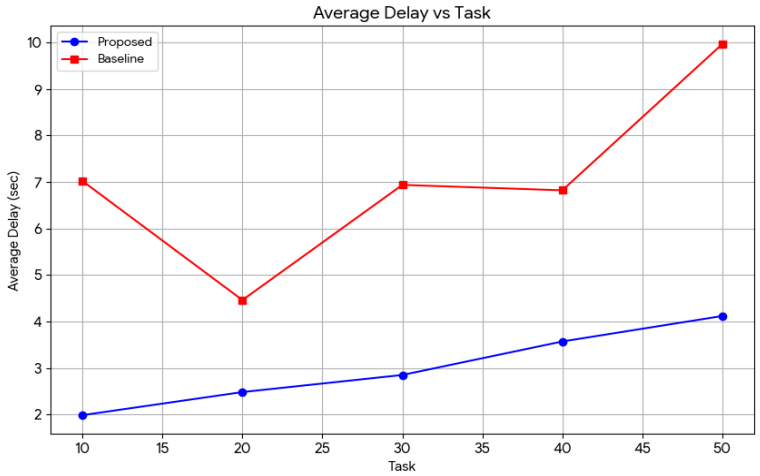

Figure 2: Average delay comparison with all offloading.

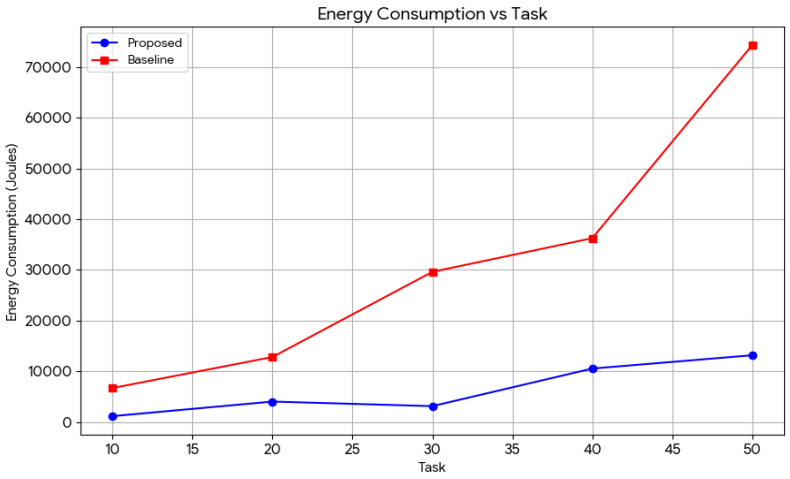

Figure 3: Energy consumption comparison with all offloading.

5 Conclusion

The implementation successfully translates the theoretical NOMA-MEC UAV model into a functional simulation. By utilizing a Genetic Algorithm, the system efficiently navigates the tradeoff between local processing capabilities and edge computing resources. The simulation confirms the conclusion that the TOA algorithm can effectively satisfy computational resources and energy consumption constraints while reducing total task processing time compared to static allocation methods.

References

[1] Xiao, T., Du, P., Gou, H., & Zhang, G. (2024). *NOMA-MEC Based Task Offloading Algorithm in UAV-assisted IoV Networks*. 2024 3rd International Conference on Computing, Communication, Perception and Quantum Technology (CCPQT). IEEE.