MOVE: Matching Game for Partial Offloading in Vehicular Edge Computing

Mahmuda Akter[§], Debjyoti Sengupta[§], Anurag Satpathy, and Sajal Das
Department of Computer Science
Missouri University of Science and Technology, Rolla, USA
E-mail: {makter, dsengupta, anurag.satpathy, sdas}@mst.edu

ICC 2024 - IEEE International Conference on Communications | 978-1-7281-9054-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICC51166.2024.10622282

***Abstract*—Autonomous Vehicles (AVs) require substantial computational resources to perform operations that safely navigate vehicles in urban road networks. Resource-intensive operations are offloaded to roadside units (RSUs), acting as edge servers, to improve the responsiveness and reduce the energy consumed in execution. In this context, a cooperative execution involving the vehicular on-board units (OBUs) and the RSUs can act as a game changer. However, partial offloading is non-trivial and demands addressing the following research challenges. Firstly, the RSU's resources are limited, necessitating regulated resource assignments. Secondly, capturing distinctive vehicle parameters using a unified ranking scheme is imperative. Thirdly, an efficient partition strategy must consider the energy expended and adhere to the real-time operations' deadline needs. This paper proposes a partial offloading scheme, MOVE, catering to the abovementioned challenges. A deferred acceptance algorithm (DAA) with preferences is proposed to address the first two challenges, whereas a novel energy-aware partitioning strategy resolves the final challenge. The performance of the proposed scheme is evaluated against baseline algorithms, and we observed a 54.04% and 52.17% reduction in offloading latency and energy.**

***Index Terms*—Vehicular Edge, Road Side Units, Partial Offloading, Matching Theory, Deadline, Latency, Energy Awareness**

I. Introduction

Imparting autonomy in vehicles implies equipping them with higher cognitive and control functions, requiring higher computational power. To achieve this, an option is to manufacture vehicles with better hardware capability to handle increasingly complex computations in real time. However, this results in higher automobile costs and computing requirements to meet ever-changing demands. Additionally, periodic upgrades in the vehicle's hardware may not be feasible, especially with large-scale data-driven and intelligent models for autonomous driving. To address this problem, and as part of an intelligent transportation system (ITS), computing resources in the form of roadside units (RSUs) are strategically installed [1], [2], catering to vehicular computations during transit [3], [4] and assisting in safe navigation in urban road networks.

Given the scale, offloading computations to the RSU is a non-trivial problem with inherent research challenges. (a) The RSU has limited computing resources, considering urban road networks; (b) A vehicle trajectory and traversal patterns are highly dynamic, comprising multiple attributes. (c) The requirement of a real-time solution, adhering to stringent quality-

[§]Both authors contributed equally to this work.

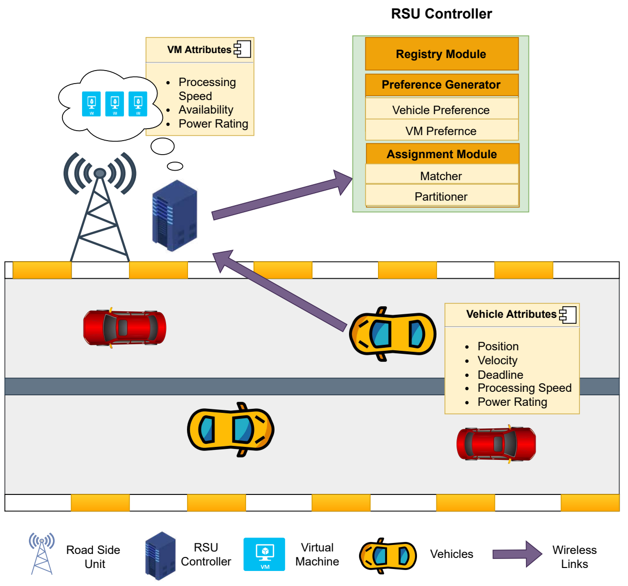

Fig. 1. Vehicular Edge Computing setup with vehicles, RSU, and controller.

of-service (QoS) regarding *service deadlines* is indispensable [5], [6]. This paper proposes a systematic solution to the above challenges. The novel contributions are as follows.

- We propose a framework called **M**atching Game for Partial **O**ffloading in **V**ehicular **E**dge Computing (MOVE) to facilitate collaborative execution between vehicular OBUs and RSU, ensuring deadline-compliant operations.
- To meet the stringent requirements on tolerable latency, we propose a *polynomial-time* and *scalable* matching solution that generates an effective offloading plan and considers the distinctive preferences of competing agents.
- By considering their respective power ratings, we devise a novel partition technique that effectively determines the offloaded and local fraction.
- We evaluate the performance of MOVE; Compare it with three baseline algorithms: Sequential Optimal Matching (SOM), Matching Theory for Task Offloading [7] (MTO), and Random Assignment (RA); and observe a 54.04% and 52.17% reduction in offloading latency and energy.

The rest of the paper is organized as follows. Section II reviews the related literature on offloading. Section III describes the system design and problem formulation, while Section IV presents the proposed solution. Section V discusses the experimental results, and Section VI offers conclusions.

II. Related Work

With the increasing computational demands of modern vehicles, vehicular edge computing (VEC) has emerged as a pivotal technology, extending cloud computing capabilities closer to the edge of vehicular networks [8], [9]. Prior research mostly focused on full offloading to RSUs to reduce the processing burden of the vehicles. While these approaches have effectively utilized the computational resources available in the RSU, they have introduced latency issues for real-time applications due to continuous data transmission with RSU servers. Addressing the latency issues, Deng *et al.* [10] proposed a novel resource allocation policy in multi-user Mobile Edge Cloud (MEC) systems designed to optimize communication and computing resources while maintaining system constraints. Some user applications are highly latency-sensitive, and Wang *et al.* [11] addressed this by using joint task scheduling and resource allocation (JTSRA). Moreover, due to the huge influx of data, full offloading often leads to higher energy consumption on the RSU, and Hu *et al.* [12] proposed an auction-bid scheme to mitigate the issues of energy consumption effectively.

Full offloading incurred increased energy consumption and higher latency, particularly concerning real-time applications such as autonomous vehicles. Therefore, partial offloading with cooperative execution was a viable execution option. MEC has been investigated in works [13], [14], and a balanced and efficient partial offloading scheme with optimal partitions has been proposed. Alternatively, Fang *et al.* [15] discussed offloading in non-orthogonal multiple access (NOMA)-MEC network with multiple users sharing the frequency bands. Based on the user's delay and data transmitting power, a bisection searching (BSS) algorithm is adopted to obtain an optimal task partition ratio to minimize the system delay. Similarly, the work in [16] presented a framework to determine the optimal partition of data, balancing the energy consumed and saved. The balance was established by considering different interdependent applications and system-specific parameters. On the other hand, the work in [17] determined the optimal partition considering the vehicle's CPU speed, available channel bandwidth, and transmission power. Alternatively, Bi *et al.* [17] investigated the offloading decision for moving user equipment (UE) with the agenda of latency and energy reduction. The gap in current literature lies in the predominant focus of most studies on full/partial offloading, which is solely centered on latency and energy reduction, neglecting non-negotiable operational deadlines.

III. System Design and Formulation

This section formally introduces the offloading problem, subsequently followed by the overall objective of MOVE.

A. System Design

We consider a setup with autonomous vehicles $\mathbb{A} = \{a_1, a_2, a_3, \dots a_m\}$ traversing a two-lane network. An autonomous vehicle a_i performs compute-intensive operations by utilizing the on-board units (OBUs) having ω_i capacity (in *cycles/bit*) and execution frequency f_i (*cycles/sec*). For two vehicles a_i and $a_{i'}$, we assume that $\omega_i \neq \omega_{i'}, \forall\, i, i' \in [1, m]$ to impart heterogeneity. The OBUs have integrated transceivers and receivers that assist in communicating with other vehicles and RSUs. The communication channel is assumed to be noiseless with a transmission rate of r (*bps*). The OBUs are resource-limited and may not have the requisite computing resources to perform many real-time operations and may depend on the nearby RSU, which is virtualized into 'n' VMs, captured by $\mathbb{M} = \{m_1, m_2, \dots, m_n\}$. A VM m_j's computing resources are characterized by ω_j (in *cycles/bit*) operating at a frequency of f_j (*cycles/sec*). This limited capacity of the RSU and the real-time needs of the vehicles motivate us to develop a cooperative offloading scheme, where the vehicle and RSUs cooperate in executing the computations strictly adhering to the deadline constraints. The fraction of the computations executed locally and remotely are respectively captured as $\beta^l_{i,j}$ and $\beta^0_{i,j}$, such that $\beta^l_{i,j} + \beta^o_{i,j} = 1$. For instance, vehicular operations such as map updates, path planning, and fleet learning are arbitrarily splittable.

For a vehicle a_i with coordinates (x_i, y_i) and the RSU positioned at location (x_{rsu}, y_{rsu}), with a coverage radius R, the coverage distance to be traversed (d_i) can be computed as:

$$d_i = \sqrt{R^2 - (y_{rsu} - y_i)^2} \pm (x_{rsu} - x_i) \qquad (1)$$

With the assumption of constant velocity, the time-window of a_i before exiting the coverage area, also termed as *sojourn time* (T^{soj}_i), is computed as follows [18]:

$$T^{soj}_i = \frac{d_i}{v_i} \qquad (2)$$

B. Modelling Communication and Processing Delays

The offloading request is expressed as a quadruple $< x_i, y_i, v_i, \Delta_i >$, where Δ_i captures the imposed deadline in performing the requisite operation. We subsequently introduce the communication and processing delays in MOVE.

1) Communication Delay: Equation (3) captures the delay in communication ($\mathbb{T}^c_i(\beta_{i,j}, \eta)$), considering both transmission and reception. Here, D_i and D'_i denote the size of data pre and post-processing, respectively.

$$\mathbb{T}^c_i(\beta_{i,j}, \eta) = \left(\beta^o_{i,j} \times \frac{D_i}{r}\right) + \frac{D'_i}{r} \qquad (3)$$

2) Local Processing Delay: The local processing delay ($\mathbb{T}^l_i(\beta_{i,j}, \eta)$) captures the time expended in processing $\beta^l_{i,j}$ fraction of the data on the OBU of the vehicle.

$$\mathbb{T}^l_i(\beta_{i,j}, \eta) = \left[(\beta^l_{i,j} \times D_i) \times \frac{\omega_i}{f_i}\right] \qquad (4)$$

3) Offloaded Processing Delay: The processing delay is captured as $\mathbb{T}^p_i(\beta_{i,j}, \eta)$ reflects the time expended by a_i in processing the offloaded computations at the VM m_j of the RSU and is derived as:

$$\mathbb{T}^p_i(\beta_{i,j}, \eta) = \left[(\beta^o_{i,j} \times D_i) \times \frac{\omega_j}{f_j}\right] \qquad (5)$$

4) Total Offloading Delay: It indicates the time elapsed in communication and processing and is captured as:

$$\mathbb{T}_i^o(\beta_{i,j}, \eta) = \mathbb{T}_i^c(\beta_{i,j}, \eta) + \mathbb{T}_i^p(\beta_{i,j}, \eta) \tag{6}$$

C. Energy Consumption

1) Energy Consumption at the VMs: Given the power rating of m_j, denoted as p_j, the energy consumption at m_j denoted by $\mathcal{E}_j$ for processing offloaded data is derived as:

$$\mathcal{E}_j = p_j \times \mathbb{T}_i^p(\beta_{i,j}, \eta) \tag{7}$$

2) Energy Consumption at the Vehicles: A vehicle a_i having a power rating of p_i will have local energy consumption equal to the value obtained as:

$$\mathcal{E}_i = p_i \times \mathbb{T}_i^l(\beta_{i,j}, \eta) \tag{8}$$

D. Problem Formulation

Let us introduce an indicator variable

$$\mathcal{X}(a_i, m_j) = \begin{cases} 1 & \text{If } a_i \text{ is assigned to } m_j \\ 0 & \text{Otherwise} \end{cases} \tag{9}$$

The overall objective of MOVE is portrayed in Equation (10a). The delay in offloading must meet the deadline imposed by the vehicle, which is captured as a Constraint (10b). On the other hand, with Constraint (10c), a vehicle can offload to at most one VM. The energy awareness in MOVE is incorporated via Constraint (10d), which essentially states that partial offloading saves the RSU from expending energy completely. Constraints (10e) and (10f) capture the feasibility of the partitioning logic. Finally, Constraint (10g) captures the values for the decision variables.

$$\mathit{minimize} \left(\sum_{i=1}^{m} \mathbb{T}_i^o(\beta_{i,j}, \eta) \right) \tag{10a}$$

$$\textit{s.t.} \quad \max(\mathbb{T}_i^o(\beta_{i,j}, \eta), \mathbb{T}_i^l(\beta_{i,j}, \eta)) \le \min(\Delta_i, T_i^{soj}) \tag{10b}$$

$$\mathcal{X}(a_i, m_j) \le 1 \tag{10c}$$

$$\sum_{j=1}^{n} \mathcal{E}_j \le \sum_{i=1}^{m} \sum_{j=1}^{n} \mathcal{I}(a_i, m_j) \frac{\omega_j}{f_j} (p_j \times D_i) \tag{10d}$$

$$\boldsymbol{\beta_{i,j}} \in \mathbb{R}^+ \tag{10e}$$

$$\mathbf{1}^{\boldsymbol{T}} \boldsymbol{\beta_{i,j}} \le 1 \tag{10f}$$

$$\forall\, i \in [1, m],\ \forall\, j \in [1, n] \tag{10g}$$

$\mathcal{NP}$**-Hardness**: Indeed the problem expressed as per Equation (10) is a combinatorial optimization problem proven be $\mathcal{NP}$-Hard [19]. This implies that for increasing problem size, obtaining an optimal solution will become extremely complex and challenging. Therefore, in this work, we discuss an effective solution that not only generates a *stable*, *scalable*, *efficient*, *polynomial-time* assignment based on strategy but also considers the individual *preferences* of stakeholders. The next section elaborately discusses the solution strategy adopted.

IV. Proposed Model

The overall working and components of MOVE are captured in the *RSU controller* as illustrated in Figure 1. It has a *registry*, *preference generator*, and *assignment* modules. A vehicle registers itself with the RSU when it enters the coverage area, and the VMs are registered as soon as they are instantiated. The registration involves logging the vehicle and VM attributes with the controller. The preference generation procedure follows the registration, wherein the vehicles and VMs generate their respective preferences that are input to the assignment module. The assignment module encompasses the *matcher* and *partitioner*. The former assigns vehicles to VMs, whereas the latter outputs an energy-balanced partition.

A. Offloading as a Matching Game

The resource competition game between the vehicles for virtualized resources as VMs at the RSUs is modeled as a matching game [20], [21]. Formally, the game is played between two sets of agents $\mathbb{A} = \{a_1, a_2, a_3, \dots a_m\}$ and $\mathbb{M} = \{m_1, m_2, \dots, m_n\}$. The matching game is defined as:

Definition 1: (**Agent Preference**): A matching is an association between two sets of agents $\mathbb{A}$ and $\mathbb{M}$. Every agent $u \in \mathbb{A} \cup \mathbb{M}$ defines a ranked ordering $\succ_u$, capturing its priorities over agents in the opposite set. For instance, $m_j \succ_{a_i} m_{j'}$ implies that a_i prefers machine m_j over $m_{j'}$. On the other hand, $a_i \succ_{m_j} a_{i'}$ states that m_j prefers vehicle a_i over $a_{i'}$.

Definition 2: (**Offloading Game**): Matching between $\mathbb{A}$ and $\mathbb{M}$ is a one-to-one association $\eta : \mathbb{A} \to \mathbb{M} \cup \{\Phi\}$ subject to the following.

Condition 2.1: Each $a_i \in \mathbb{A}$, $|\eta(a_i)| = 1$, and $\eta(a_i) \subset \mathbb{M}$.

Condition 2.2: Each $m_j \in \mathbb{M}$, $|\eta(m_j)| = 1$, and $\eta(m_j) \subset \mathbb{A}$.

Condition 2.3: For a vehicle machine pair (a_i, m_j), $a_i \in \eta(m_j) \iff m_j \in \eta(a_i)$.

Conditions (2.1) and (2.2) enforce a one-to-one assignment between the VMs and the vehicles. The property of symmetry is implied by Condition (2.3), which implies that a VM is assigned to a vehicle *iff* the converse is true.

Definition 3: (**Pairwise Blocked**): A vehicle-VM pair (a_i, m_j) blocks the matching η *iff* the follwing are satisfied.

Condition 3.1: $a_i \notin \eta(m_j)$ and $m_j \notin \eta(a_i)$

Condition 3.2: $m_j \succ_{a_i} \eta(a_i)$

Condition 3.3: $a_i \succ_{v_j} \eta(m_j)$

A pair of agents (a_i, m_j) is said to block a matching if they are incentivized to deviate from their currently assigned partners in η and be matched to each other, which is enforced by Conditions (3.1)-(3.3).

Definition 4: (**Stability**) The matching η is stable *iff* no pair of agents block the matching.

B. Agent Preference

The preference of an entry $a \in \mathbb{V} \cup \mathbb{M}$ is captured as $\mathbb{P}(a)$.

1) Vehicle Preferences: For an AV a_i, the preference over any two VMs is expressed as follows.

$$m_j \succ_{a_i} m_{j'} \iff \mathbb{T}_i^o(\beta_{i,j}, \eta) < \mathbb{T}_i^o(\beta_{i,j'}, \eta); \;\; j \ne j' \tag{11}$$

2) Machine Preferences: To capture the machine priorities, we associate a *prominence score* $\rho_{j,i}$ for each a_i for offloading it to m_j derived as Eq. (12). A higher prominence prioritizes vehicles with limited execution options and closer deadlines, boosting MOVE in achieving its goal of reduced latency.

$$\rho_{j,i} = \frac{1}{|\mathbb{P}(a_i)|} + \frac{T_i^{soj}}{\Delta_i} \tag{12}$$

Therefore, the preference of m_j can be defined as

$$a_i \succ_{m_j} a_{i'} \iff \rho_{j,i} > \rho_{j,i'}; \;\; i \neq i' \tag{13}$$

C. Resource Matching Strategy

The resource competition is modeled as a one-to-one matching game, and the formal procedure for the same is presented as Algorithm 1. The inputs to the algorithm are the sets $\mathbb{A}$ and $\mathbb{M}$, preferences of agents, i.e., $\mathbb{P}(a), a \in \mathbb{A} \cup \mathbb{M}$, power profiles of the respective agents. The output of the algorithm is an energy-efficient partition post-assignment. The algorithm proceeds as follows. The participating agents in the matching are initialized to be free, which implies a null assignment going into RMA. RMA identifies any AV a_i, which is yet to be assigned, i.e., $free[a_i]$, and has some options left, i.e., $\mathbb{P}(a_i) \neq \Phi$. The procedure then extracts the most preferred option m_j from $\mathbb{P}(a_i)$ using **ExtractPreferred** (.). Once identified, the machine m_j is approached for assignment (Step 2-4 of Algorithm 1). The machine can respond to the proposal in three ways (Steps 5-12 of Algorithm 1).

(1) The machine m_j is free, i.e., $free[m_j] = True$. The machine m_j and AV a_i are matched in such a case.

(2) The machine m_j is holding onto a proposal from $a_{i'}$ from some previous iteration but $a_i \succ_{m_j} a_{i'}$. Here, $a_{i'}$ is evicted as it is less preferred than a_i, and a_i and m_j were matched.

(3) On the other hand, $a_{i'} \succ_{m_j} a_i$, the request from a_i is rejected and the previous matching is retained.

Once the matching is obtained, RMA obtains an effective partition for each offloaded computation. The fraction of the computation to be performed locally is computed as the ratio of their power ratings (Step 15 of Algorithm 1). Moreover, full offloading is performed if the locally offloaded fraction cannot meet the deadline constraint imposed by the vehicle (Step 15-20 of Algorithm 1). This procedure is repeated for every assigned vehicle.

D. Theoretical Analysis

Theorem 1: (**Stability**): RMA produces a stable matching η wherein no matched agents are incentivized to deviate from their current assignments.

Proof: By Definitions (3) and (4), a matching η is said to be stable *iff* there is no vehicle-VM pair (a_i, m_j) that has incentive to deviate from currently assigned partners.

We prove this by contradiction and assume the presence of a blocking pair (a_i, m_j) in the matching η. This implies that a_i and m_j are not currently matched to each other, i.e., $a_i \notin \eta(m_j)$ and $m_j \notin \eta(a_i)$. As (a_i, m_j) forms a blocking pair, we

Algorithm 1: Resource Matching Algorithm (RMA).

```
Input: $\mathbb{A}$, $\mathbb{M}$, $\mathbb{P}(a_i), p_i \,\forall\, a_i \in \mathbb{A}$;  $\mathbb{P}(m_j), p_j \,\forall\, m_j \in \mathbb{M}$.
Result: $\boldsymbol{\beta}$ {Offloading Vector}
1  Initialize: $\eta(a_i) = \Phi, free[a_i] = True, \forall a_i \in \mathbb{A}$;
       $\eta(m_j) = \Phi, free[m_j] = True, \forall v_j \in \mathcal{M}, \boldsymbol{\alpha} = \Phi$
2  while $\exists\, a_i \,|\, free[a_i] = True \,||\, \mathbb{P}(a_i) \neq \Phi$ do
3      $m_j$ = ExtractPreferred ($\mathbb{P}(a_i)$)
4      Send proposal to $m_j$
5      if $free[m_j]$ then
6          $\eta(a_i) = \{m_j\}$;  $\eta(m_j) = \{a_i\}$
7          $free[a_i] = False$;  $free[v_j] = False$
8      else if $a_i \succ_{m_j} a_{i'}$ then
9          $\eta(a_{i'}) = \Phi$;  free$[a_{i'}] = True$
10         $\eta(a_i) = \{m_j\}$;  $\eta(m_j) = \{a_i\}$
11         $free[a_i] = False$;
12     else
13         Reject the proposal from $a_i$
14 while $\exists\, a_i \,|\, !\,(free[a_i])$ do
15     $\beta^l_{i,j} = \frac{1}{1 + \frac{p_i}{p_j}}$
16     if $\mathbb{T}^l_i(\beta_{i,j}, \eta) \leq \Delta_i$ then
17         $\beta^o_{i,j} := 1 - \beta^l_{i,j}$
18         $\boldsymbol{\beta_{i,j}} := \{\beta^l_{i,j}, \beta^o_{i,j}\}$
19     else
20         $\boldsymbol{\beta_{i,j}} := \{0, 1\}$
21 return $\beta$
```

can safely deduce that $m_j \succ_{a_i} \eta(a_i)$ and $a_i \succ_{m_j} \eta(m_j)$. It can be noted that as $m_j \succ_{a_i} \eta(a_i)$, a_i must have proposed m_j prior to proposing $\eta(a_i)$ and as $\eta(a_i) \neq m_j$, implies that it was previously rejected. This rejection must have been performed to accommodate a more preferred option, i.e., VM $m_{j'}$, which implies that $\eta(a_i) \succ_{a_i} m_j$, i.e., $m_{j'} \succ_{a_i} m_j$, which contradicts our assumption. Therefore, we can safely conclude that RMA produces a stable matching. ■

Theorem 2: (**Time Complexity**): RMA converges to a stable solution in $O(m \times n)$ + $O(m)$ time, which is polynomial.

Proof: Let the number of vehicles in the setup be 'm', i.e., $|\mathcal{A}| = m$, and the number of VMs is 'n,' i.e., $|\mathcal{M}| = n$. The time complexity of RMA (Algorithm 1) is dependent on (*i.*) resource assignment (Steps 2-13) and (*ii.*) partition logic (Steps 14-20). In its worst case, the former takes $O(m \times n)$, wherein every vehicle a_i having a complete preference $\mathbb{P}(a_i)$ sends out a proposal request to all the m_j's in its preference. However, the latter executes in $O(m)$. Therefore, the overall complexity of RMA is $O(m \times n)$ + $O(m)$, which is polynomial. ■

V. Performance Study

This section describes the baseline algorithms, experimental setup, and the comparative study of results obtained.

A. The Baseline Algorithm

To evaluate the performance of MOVE, we compare it with the following approaches.

- **Sequential Optimal Matching (SOM)**: The vehicles are sequentially matched to the VMs sorted according to their computing capabilities. This assignment should also adhere to the Constraints expressed in Equation (10).

- **A Matching Theory Framework for Task Offloading in Fog Computing for IoT Systems [7] (MTO)**: This protocol uses matching in IoT-Fog systems with the objective of worst-case time reduction. A static model variant with consistent preferences is considered for fair and rational comparison.
- **Random Assignment (RA)**: The vehicles are randomly matched to VMs and checked for the satisfaction of all the Constraints in Equation (10).

B. Experimental Setup

We consider a 2-lane road network with vehicles traversing the area and a RSU with a coverage of 500 m. The compute resources at the RSU are logically partitioned into VMs (equal to number of vehicles) with a processing frequency and execution power ranging between $[0.1-1.0]$ GHz and $[150-500]$ MW, where vehicles are randomly distributed within a two-lane road. Based on urban and semi-urban driving conditions, the vehicles are assumed to be traversing the coverage area with their average speed ranging from 40 to 70 Kmh^{-1}. All the vehicles have onboard units (OBUs) with wireless transceivers and receivers that assist the vehicles in communicating with the RSU, and the power consumed is randomly generated in the range $[250-400]$ *MW*. We assume the communication is carried over a noiseless wireless channel [22] having a fixed bandwidth of 3 MHz. The computations on the vehicles have varying data sizes ranging from [100-1000] KB, where lower data size values represent relatively lightweight computational tasks and the higher values represent more computationally intensive tasks. To represent different levels of urgency, the tasks' deadline values are generated uniformly in the in $[0.5-25]$ sec. Additionally, we evaluate the performance of MOVE against the benchmarks by performing 10 test runs over 4 evaluation scenarios, i.e., S-1, S-2, S-3, S-4 corresponding to 100, 200, 300, and 400 vehicles respectively.

C. Results and Analysis

This subsection presents the comparative study of all the schemes implemented across different evaluation metrics.

1) Offloading Delay: Figure 2 captures the comparative behavior of aggregate latency for all the protocols. It can be observed from the figure that the proposed algorithm, i.e., MOVE, demonstrates superior performance by achieving minimum latency compared to the benchmarking algorithms across test cases. Note that the latency increases for increasing vehicles, which is an expected behavior. The improved performance of MOVE is attributed to the following two reasons. Firstly, the vehicles rank VMs in the order of increasing offloading delay. Secondly, vehicles are proposers, so they approach the VMs in the designated order of $P(v_i)$. These reasons boost the possibility of the vehicles being assigned to more preferred choices. Moreover, unlike MOVE, SOM and MTO perform full offloading in the order of compute capabilities and transmission delays, resulting in higher offloading duration. We observe an interesting behavior for the RANDOM assignment, wherein its offloading delay is lower than that of SOM. However, this is only because of the non-consideration of outage vehicles towards the offloading delays, which can be easily inferred from Figure 3.

2) Outages: MOVE suffers from minimum outages than the baselines. This can easily be established from Figure 3 and is attributed to the ranking scheme adopted (refer to Equation (13)). The VM ranking scheme prioritizes assigning vehicles that have (*i.*) limited execution options, i.e., $|P(v_i))|$ and (*ii.*) lower T_i^{soj}/Δ_i ratio. Such a combined valuation boosts the possibility of vehicles meeting their deadline and sojourn time constraints, thereby exhibiting superior performance compared to SOM and MTO, which consider neither. On the other hand, the RANDOM algorithm exhibits the worst performance due to the non-consideration of deadline and sojourn time while making the assignments.

3) Offloading Energy: The energy expended in the offloading is captured in Figure 4. The energy expended increases with increasing vehicles, which is expected behavior. However, we can infer from the figure that MOVE expends minimum energy compared to the benchmarking algorithms. This is because of the consideration of the energy during the partitioning, resulting in reduced energy consumption. Though the consideration of transmission delay in MTO results in slight energy reduction compared to SOM and RA, much energy is expended in execution, which is not considered, thereby consuming more energy than MOVE However, SOM and RA do not make offloading decisions based on the offloading energy, resulting in much higher energy dissipation.

4) Agent Satisfaction: The agents' satisfaction reflects their happiness quotient in the matching given a list of preferences. Note that it can only be computed for matching-based schemes, which require agents to have such a ranked ordered list, which is not true with non-matching-based solutions. The satisfaction $S(a)$ of an agent $a \in \mathbb{A} \cup \mathbb{M}$ can be computed as per Equation (14), where, $rank(a)$, captures the position of a in $\mathbb{P}(a)$ such that $rank(a) \in [0, |\mathbb{P}(a)|-1]$.

$$S_a(\%) = \frac{|\mathbb{P}(a)| - rank(a)}{|\mathbb{P}(a)|} * 100 \tag{14}$$

Figures 5 and 6 capture the average satisfaction of VMs and vehicles for MOVE and MTO. The former outperforms the latter, but we observe a slight dip with increasing vehicles in both protocols, as the preferred VMs can be assigned to a vehicle. On the other hand, the satisfaction of VMs also shows similar behavior for both the protocols considered. The improved satisfaction levels in MOVE imply that it can achieve a relatively fair assignment compared to MTO.

VI. Conclusions & Future Directions

This paper introduces a cooperative offloading scheme using matching-theory, enabling vehicles and RSUs to execute computations with deadline constraints. Due to resource scarcity at RSUs compared to the high number of vehicles, resource competition is modeled as a matching game with preferences. These preferences are tailored to enhance the scheme's objectives. Cooperative execution is facilitated by a partition

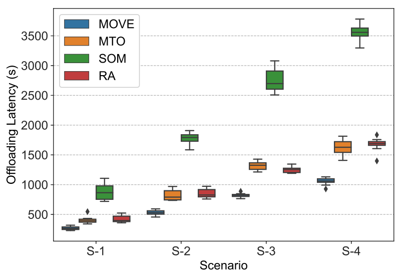

Fig. 2. Offloading Latency (s) vs. Scenario.

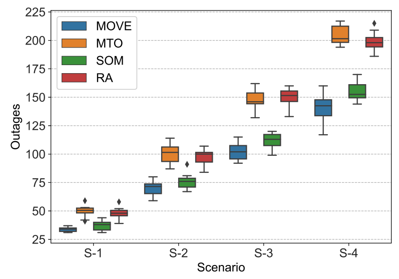

Fig. 3. Outages vs. Scenario.

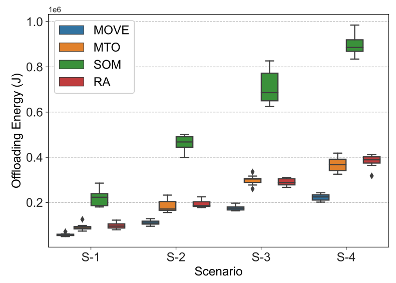

Fig. 4. Offloading Energy (J) vs. Scenario.

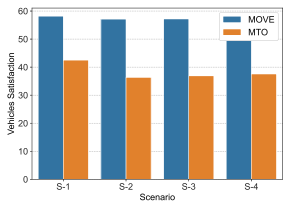

Fig. 5. Vehicles Satisfaction (%) vs. Scenario.

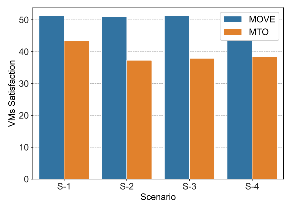

Fig. 6. VM Satisfaction (%) vs. Scenario.

logic considering VM and vehicle power ratings. Performance evaluation against three baselines demonstrates a reduction of 54.04% and 52.17% in offloading latency and energy consumption, respectively, showcasing the effectiveness of MOVE.

In future, we will plan to incorporate mobility by accounting for variable speeds, which significantly influence decision-making, particularly in terms of sojourn time. Furthermore, we intend to investigate the potential queueing delays at RSUs. Additionally, we intend to explore aspects such as channel interference, hybrid communication, dynamic preferences, and road layouts through real-world prototype implementations.

Acknowledgements: This work is supported by Kummer Doctoral Fellowship at Missouri S&T and NSF grants EPCN-2319995, OAC-2104078, SaTC-2030624, and CNS-1818942.

REFERENCES

[1] P. Chennakesavula, J.-M. Wu, and A. Ambikapathi, “Incentive-Driven Fog-Edge Computation Offloading and Resource Allocation for 5G-NR V2X-Based Vehicular Networks,” in *IEEE 97th Vehicular Technology Conference (VTC2023-Spring)*, 2023, pp. 1–5.

[2] C. Swain, M. N. Sahoo, A. Satpathy, K. Muhammad, S. Bakshi, and J. J. P. C. Rodrigues, “A-DAFTO: Artificial Cap Deferred Acceptance-Based Fair Task Offloading in Complex IoT-Fog Networks,” *IEEE Transactions on Consumer Electronics*, vol. 69, no. 4, pp. 914–926, 2023.

[3] J. Wang, J. Steiber, and B. Surampudi, “Autonomous ground vehicle control system for high-speed and safe operation,” in *American Control Conference*, 2008, pp. 218–223.

[4] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, and T. Azumi, “Autoware on board: Enabling autonomous vehicles with embedded systems,” in *ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, 2018, pp. 287–296.

[5] H. Lu, Q. Liu, D. Tian, Y. Li, H. Kim, and S. Serikawa, “The Cognitive Internet of Vehicles for Autonomous Driving,” *IEEE Network*, vol. 33, no. 3, pp. 65–73, 2019.

[6] N. Lu, N. Cheng, N. Zhang, X. Shen, and J. W. Mark, “Connected Vehicles: Solutions and Challenges,” *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 289–299, 2014.

[7] F. Chiti, R. Fantacci, and B. Picano, “A matching theory framework for tasks offloading in fog computing for IoT systems,” *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 5089–5096, 2018.

[8] A. B. De Souza, P. A. L. Rego, T. Carneiro, J. D. C. Rodrigues, P. P. R. Filho, J. N. De Souza, V. Chamola, V. H. C. De Albuquerque, and B. Sikdar, “Computation Offloading for Vehicular Environments: A Survey,” *IEEE Access*, vol. 8, pp. 198 214–198 243, 2020.

[9] H. Wang, T. Liu, B. Kim, C.-W. Lin, S. Shiraishi, J. Xie, and Z. Han, “Architectural Design Alternatives Based on Cloud/Edge/Fog Computing for Connected Vehicles,” *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2349–2377, 2020.

[10] Y. Deng, Z. Chen, X. Chen, and Y. Fang, “Throughput Maximization for Multiedge Multiuser Edge Computing Systems,” *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 68–79, 2022.

[11] G. Wang, F. Xu, and C. Zhao, “Multi-access edge computing based vehicular network: Joint task scheduling and resource allocation strategy,” in *IEEE International Conference on Communications Workshops (ICC Workshops)*, 2020, pp. 1–6.

[12] B. Hu, Y. Shi, and Z. Cao, “Adaptive energy-minimized scheduling of real-time applications in vehicular edge computing,” *IEEE Transactions on Industrial Informatics*, vol. 19, no. 5, pp. 6895–6906, 2023.

[13] J. Ren, G. Yu, Y. Cai, and Y. He, “Latency optimization for resource allocation in mobile-edge computation offloading,” *IEEE Transactions on Wireless Communications*, vol. 17, no. 8, pp. 5506–5519, 2018.

[14] J. Wang, T. Lv, P. Huang, and P. T. Mathiopoulos, “Mobility-aware partial computation offloading in vehicular networks: A deep reinforcement learning based scheme,” *China Communications*, vol. 17, no. 10, pp. 31–49, 2020.

[15] F. Fang, Y. Xu, Z. Ding, C. Shen, M. Peng, and G. K. Karagiannidis, “Optimal Resource Allocation for Delay Minimization in NOMA-MEC Networks,” *IEEE Transactions on Communications*, vol. 68, no. 12, pp. 7867–7881, 2020.

[16] S. Chouhan, “Energy optimal partial computation offloading framework for mobile devices in multi-access edge computing,” in *International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2019, pp. 1–6.

[17] J. Bi, H. Yuan, S. Duanmu, M. Zhou, and A. Abusorrah, “Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization,” *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3774–3785, 2021.

[18] S. S. Shinde, A. Bozorgchenani, D. Tarchi, and Q. Ni, “On the Design of Federated Learning in Latency and Energy Constrained Computation Offloading Operations in Vehicular Edge Computing Systems,” *IEEE Trans. on Vehicular Technology*, vol. 71, no. 2, pp. 2041–2057, 2022.

[19] Z. Liu, X. Yang, Y. Yang, K. Wang, and G. Mao, “DATS: Dispersive stable task scheduling in heterogeneous fog networks,” *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3423–3436, 2018.

[20] B. Gu and Z. Zhou, “Task Offloading in Vehicular Mobile Edge Computing: A Matching-Theoretic Framework,” *IEEE Vehicular Technology Magazine*, vol. 14, no. 3, pp. 100–106, 2019.

[21] C. Swain, M. N. Sahoo, and A. Satpathy, “SPATO: A student project allocation based task offloading in IoT-Fog systems,” in *IEEE International Conference on Communications*, 2021, pp. 1–6.

[22] D. Jiang and L. Delgrossi, “IEEE 802.11p: Towards an International Standard for Wireless Access in Vehicular Environments,” in *IEEE Vehicular Technology Conference*, 2008, pp. 2036–2040.