Machine Learning-Based Workload Orchestrator for Vehicular Edge Computing

Cagatay Sonmez, *Member, IEEE*, Can Tunca, Atay Ozgovde, *Senior Member, IEEE*, and Cem Ersoy, *Senior Member, IEEE*

***Abstract*— The Internet of Vehicles (IoV) vision encompasses a wide range of novel intelligent highway scenarios that rely on vehicles with an ever-increasing degree of autonomy and the prospect of sophisticated services like e-Horizon and cognitive driving assistance. The self-driving vehicle, on the other hand, entails a new passenger profile where sophisticated infotainment applications are expected to enhance the quality of travel. From the technical stand point, for this vision to become a reality a streamlined edge computing infrastructure, namely Vehicular Edge Computing (VEC), is required where computationally intensive workloads are offloaded to a nearby VEC infrastructure. However, the highly dynamic environment renders it difficult to efficiently operate a VEC system to yield the crisp performance required on an autonomous vehicle. In this setting, where to offload each task stands out as a crucial decision problem, and the conventional methods prove insufficient for its solution. In our work, we proposed a two-stage machine learning-based vehicular edge orchestrator which takes into account not only the task completion success but also the service time. To demonstrate how our approach performs in a realistic setting, we employed EdgeCloudSim to design extensive experiments where the characteristics of the vehicular applications, upload/download sizes, computational footprints of the tasks, the LAN, MAN and WAN network models, and the mobility are considered. Detailed performance evaluation of the proposed system via simulation is carried out where both overall and service type-specific performance scores in comparison with opponent schemes are reported.**

***Index Terms*— Intelligent transportation systems, Internet of Vehicles, vehicular edge computing, task offloading, vehicular edge orchestrator, machine learning.**

I. Introduction

WE ARE witnessing the new era of connected vehicles where transportation is undergoing a radical transformation. Vehicles of varying purpose, functionality and dimension, whether it be an airport shuttle bus, a fleet of trucks or a personal sports car are being transformed into intelligent, autonomous entities that are equipped with many sensors of varying complexities, on-board computational units and versatile communication capabilities that enable them to cooperate with the road-side infrastructure and/or with each other [1]. It is predicted that, by 2030, 95% of all U.S. passenger miles will be served by transport-as-a-service (TaaS) providers who will own and operate fleets of autonomous electric vehicles providing passengers with higher levels of service [2].

Triggered by the autonomy, a new in-vehicle experience is defined for the passengers where the transportation time is planned as if individuals are in a stationary domestic environment or in an office space [3]. In this setting, passengers are expected to not concentrate on the actual travel but instead accomplish daily tasks in an accordingly designed interior space where infotainment and productivity applications will be of primary use. This novel vision comes with a trade-off in which extensive amounts of data are produced on the vehicle itself that have to be processed with stringent real-time requirements. An autonomous vehicle fitted with imaging and scanning systems generates and processes about 4 TB of data for every hour of autonomous driving [4]. Moreover, crucial data related to the vehicular applications typically exist in a distributed fashion over the vehicles and the road-side infrastructure, which necessitates timely communication between all parties. Apart from the sensory processing, the number of in-vehicle applications used by the drivers as well as the vehicle itself has increased accordingly as the vehicles evolved in the recent years by the help of the new generation Internet of Vehicles (IoV) services such as the intelligent navigation and cognitive driver assistance. With this spectrum of increasingly sophisticated services, the on-board computational capabilities of the vehicles may become inadequate [5].

The Vehicular Edge Computing (VEC) is an emerging technology offering a basis for the in-vehicle applications to improve the availability of services using the computation offloading concept [6]. VEC improves the user experience by partially delegating the load via computational offloading to an edge infrastructure streamlined for vehicular applications. Thus, it can handle the delay intolerant complex operations. The advanced driver assistance [7], e-horizon [8], autonomous driving [9], and accident prevention applications [10] have the highest potential for the

Manuscript received January 29, 2020; revised May 13, 2020 and July 13, 2020; accepted September 8, 2020. This work was supported by the Turkish Directorate of Strategy and Budget under the TAM Project number 2007K12-873. *(Corresponding author: Cagatay Sonmez.)*

Cagatay Sonmez is with the Research and Development Center, Arcelik Electronics Plant, 34528 Istanbul, Turkey (e-mail: cagatay.sonmez@boun.edu.tr).

Can Tunca is with Pointr: The Deep Location Company, 34382 Istanbul, Turkey (e-mail: cantunca@gmail.com).

Atay Ozgovde is with the Department of Computer Engineering, Galatasaray University, 34349 Istanbul, Turkey (e-mail: aozgovde@gsu.edu.tr).

Cem Ersoy is with the Department of Computer Engineering, Bogazici University, 34342 Istanbul, Turkey (e-mail: ersoy@boun.edu.tr).

Digital Object Identifier 10.1109/TITS.2020.3024233

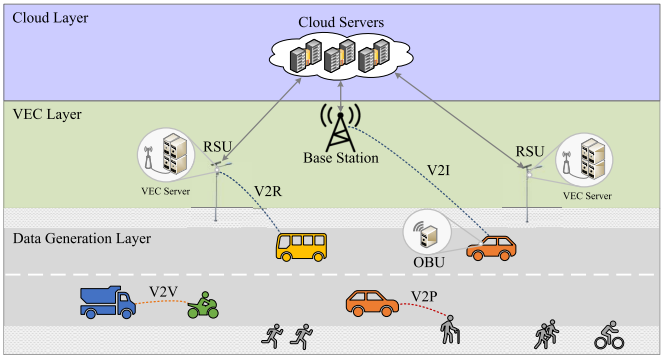


Fig. 1. Multi-tier VEC architecture for vehicular networks.

VEC based systems. These applications generally use machine learning (ML), and artificial intelligence (AI) based algorithms that process real-time data and require a significant computation power [11]. The vehicles cannot execute such algorithms efficiently due to insufficient local computing resources.

The concept of IoV, its pioneering applications, and the services for the future smart highways can benefit from the computation offloading concept over a multi-tier architecture consisting of the connected vehicles, RSUs, and cloud computing elements as shown in Fig. 1. The vehicles are located in the first tier, which can be considered as the data generation layer. They also have computational resources which are provided by their On-Board Units (OBUs). If required, some of the operations can be executed locally by the OBUs at this tier. The second tier consists of the RSUs that can provide fast access to limited resources. The edge servers are located in this tier. Finally, the third tier includes traditional cloud servers.

While the concept of task offloading offers great potential by augmenting the computational capabilities of the vehicles, it is not straightforward to design an offloading based system. Firstly, a very dynamic scene is present where a multiplicity of novel applications with varying loads and requirements are to take advantage of the infrastructure. Secondly, an overall transient operational character is inevitable due to the short residence times of the vehicles under the coverage of fixed edge server components such as Road Side Units (RSU). Moreover, the overall system involves the cooperation of local edge servers with the global cloud servers distributed over a geographical region. While this multi-tier setting increases efficiency, it also increases the state-space, thus, contributes to the overall complexity of the instantaneous decisions that are to be given by the system. As a result, an efficient workload orchestration and task scheduling scheme become a crucial requirement.

The computational and network resources on the VEC systems are heterogeneous and change very quickly. As the number of vehicles increases, the congestion on these resources becomes inevitable, which makes the decision making process more difficult [20]. Therefore, predicting the status of resources by using historical data is a challenging process. The ML-based prediction is one of the efficient methods that can be utilized in such highly dynamic environments. Zhan *et al.* proposed a deep reinforcement learning-based offloading scheduling for VEC [21], and Liu *et al.* proposed a deep learning-based V2X wireless channel prediction model using a long short-term memory (LSTM) network [22]. While ML-based approaches have been applied to various problems in many areas, they have found only limited use for VEC systems. We focus on the workload orchestration problem in the VEC environments and different ML algorithms to fill the gap in this field. Although these algorithms are used in many other areas, our work's novelty is using them for the vehicular edge orchestration as explained in Section III. To the best of our knowledge, we propose the first ML-based workload orchestrator for the multi-tier multi-access VEC architecture used in this work. The main contributions of this article are summarized as follows:

- We define a multi-access multi-tier VEC architecture. The multi-access communication technologies consist of vehicular wireless local area network (WLAN), WAN, and cellular network, which allow vehicles to apply V2I offloading. The multi-tier architecture has three tiers, including vehicles, RSUs, and cloud servers.
- We propose an ML-based workload orchestration approach for VEC systems. In particular, the ML-based orchestrator performs a two-stage process. In the first stage, a classifier model predicts whether the results of the offloading options are successful or not for each target device. In the second stage, a regression model estimates the service time of the related options. Finally, the target device which promises the lowest service time is selected.
- We experimented with multiple classifiers, including naive Bayes (NB), support vector machine (SVM) and multilayer perceptron (MLP) models. The MLP was chosen as the best classifier based on the performance comparison of these models.
- We carried out detailed simulation experiments to evaluate the performance of the proposed workload orchestrator. We consider load/congestion dependent communication and computing delays. The simulation results show that our approach outperforms its competitors regarding the task failure rate and service time.

The remainder of this article is structured as follows. Section II reviews the existing works proposing different workload orchestration techniques on VEC. Section III explains the proposed ML-based workload orchestrator for multi-access multi-tier VEC architectures. Section IV presents the simulation experiment design and analyzes the results. Finally, the conclusions and possible directions for future research are outlined in Section V.

II. Related Work

The vehicles can offload some of their tasks via the vehicle to everything (V2X) communication technologies. V2X is a broad term that refers to the different communication models used by the vehicles, such as Vehicle to Vehicle (V2V), Vehicle to Pedestrian (V2P), Vehicle to RSU (V2R) and Vehicle to Infrastructure (V2I) [23]. Different organizations are proposing

TABLE I
COMPARISON OF TASK OFFLOADING APPROACHES IN VEC SYSTEMS

Study	Proposed Algorithm	Mobility Model [a]	# of Vehicles	Hand-over	Cellular Access	Network Congestion	V2V Offloading	V2R (edge) Offloading	V2I (cloud) Offloading	Simulation Tool
Ning et al. [12]	DR Learning-based	DRJ	10	✗	✓	✗ [b]	✗	✓	✗	Python
Sun et al. [13]	Multi-armed bandit based	SUMO	9-18	✗	✗	✗ [c]	✓	✗	✗	Matlab/Veins
Wang et al. [14]	Game theory-based	N/A	6-70	✗	✓	✗ [d]	✗	✓	✗	Matlab
Liu et al. [15]	Matching theory based	SUMO	4-8	✓	✓	✗ [e]	✗	✓	✓	Matlab
Wang et al. [16]	Distributed greedy alg.	CSM	No data	✗	✗	✗ [f]	✓	✓	✗	No data
Xu et al. [17]	Genetic algorithm based	No data	20-120	✗	✗	✗ [g]	✓	✓	✗	CloudSim
Dai et al. [18]	Dist. approximation alg.	CSM	40	✗	✗	✗ [h]	✗	✓	✗	No data
Feng et al. [19]	Hybrid vehicular cloud	SUMO	No data	✗	✓	✗ [i]	✓	✓	✓	Omnet++
Our Study	Machine learning based	VSM	100-1800	✓	✓	✓	✗	✓	✓	EdgeCloudSim

[a] The mobility of vehicles can be integrated with well-known traffic simulation tools like Simulation of Urban Mobility (SUMO), or implemented via a custom movement model; DRJ = Discrete Random Jumps, VSM = Variable Speed Movement, CSM = Constant Speed Movement, N/A = Static Vehicle Position.
[b] [12] assumes that OFDMA technology is utilized for the links between vehicles and RSUs, and there is no interference between vehicles on this link. On the other hand, an interference signal is considered for the links between vehicles and BS.
[c] [13] assumes that the wireless channel state remains static during the uploading process.
[d] [14] assumes that the communication model uses a frequency-flat block fading Rayleigh channel, where the block length is more than the maximum service time of applications.
[e] [15] assumes that each vehicle is allocated an orthogonal channel, i.e., there is no interference among the vehicles. In addition, the WAN delay is assumed as a constant value.
[f] [16] assumes that each vehicle uses an independent and identically distributed channel. Only the distance is considered using a path loss factor.
[g] [17] assumes that each vehicle transmits data with fixed power and constant data transmission rate.
[h] [18] assumes that each vehicle uses CSMA/CA based data transmission, thus only one task can be transmitted in a time slot without collision.
[i] [19] assumes that there can be at most one simultaneous V2X transmission on vehicular or cellular networks.

different solutions to standardize V2X communication models. The 3rd Generation Partnership Project (3GPP) pushes the Long-Term Evolution (LTE) standard to enable V2X services. On the other hand, the Institute of Electrical and Electronics Engineers (IEEE) and the European Telecommunications Standards Institute (ETSI) propose the IEEE 802.11p standard for accessing the V2X services. Both standards have some advantages and disadvantages [24].

The task offloading is one of the most important features used in edge computing. It is comprised of various engineering decisions, such as "when", "how", and "where" to offload. The workload orchestration basically answers the where to offload question, and it has already been studied for the edge computing systems. For instance, we proposed a fuzzy logic-based workload orchestrator for the Multi-Access Edge Computing (MEC) architectures in our previous work [25]. Cao *et al.* discussed different task offloading approaches for the MEC architectures [26]. They showed that using the traditional methods may be useful for a static or slowly varying environment, but decision making is difficult on the fast-changing mobile networks. They recommended using ML-based solutions for such heterogeneous environments where the state of resources changes rapidly and cannot be predicted precisely. As a result, Cao *et al.*'s work inspired us to develop an ML-based workload orchestrator in the VEC domain. The other studies focusing on where to offload problem in the VEC systems are explained in this section and briefly compared in Table I.

There are very few studies in the literature using ML-based task offloading approaches for VEC systems. Ning *et al.* [12] proposed a deep reinforcement learning (DRL) based solution. Reinforcement algorithms have four key elements, which consist of agents, system states, actions, and rewards. When the agent performs an action, the system state changes, and a reward is received depending on the result of the action. The most challenging part of operating DRL-based solutions on the VEC systems is determining the actions and states because the VEC environments are highly dynamic, and the network and computing resources fluctuate rapidly. Ning *et al.* overcome this problem by using a reply buffer that stores previously discovered state, action, reward, and next state values in a large table. As a result, the reward and the next state are immediately observed whenever the target RSU is chosen. However, preparing the reply buffer is a complex operation. We also tried the reinforcement learning algorithm-based workload orchestrator in our study, but it provides poor performance compared to the other supervised-learning methods, which are using tagged historical data.

Sun *et al.* introduced an adaptive learning based task offloading (ALTO) algorithm for the dynamic and uncertain VEC systems [13]. As they stated, the fast varying wireless channel states and computing resources bring extra challenges to the offloading problem. They propose a multi-armed bandit (MAB) theory-based solution. MAB theory is commonly used to learn unknown environments. ALTO works in a distributed manner and tries to minimize the average offloading delay. Although their algorithm only considers the V2V offloading, we adapted Sun *et al.*'s MAB theory-based solution to our multi-access multi-tier VEC architecture as a qualified competitor algorithm. The differences and modifications made for the adaptation are explained in Section IV-A.

Wang *et al.* utilize a game theory-based approach to find the offloading probability of each vehicle in vehicular MEC networks [14]. Their non-cooperative computation offloading game model operates a best-response algorithm that tries to maximize the utility of each vehicle. The vehicles adjust their offloading probability by considering the offloading probability of other vehicles in the previous stage. They showed that this strategy could converge to the unique Nash equilibrium. Their game theory-based solution uses a lightweight and

distributed algorithm. We implemented this algorithm on our simulation tool and employed it as a competitor algorithm. The tasks can be executed locally, or on edge in Wang *et al.*'s architecture, however, our architecture considers both edge and cloud offloading scenarios. As a result, we applied slight changes to Wang *et al.*'s model, as explained in Section IV-A.

Liu *et al.* used a matching-based task offloading approach to minimize the network delay [15]. It is claimed that the matching theory can provide the decentralized and efficient solutions for the complex and dynamic network topologies. In their work, the VEC architecture includes three layers, which include the vehicles, RSUs, and a macro base station (MBS) respectively. The MBS is one of the most critical entities of their system. All the vehicles and RSUs are connected to the MBS, which is responsible for the task offloading and handover operations. The matching algorithm works iteratively by sending a matching request to the RSUs by the vehicles. We think that these requests create an additional overhead on the system and slow down the offloading decision. They performed a simulation study on Matlab by using the data obtained from the Simulation of Urban Mobility [27] (SUMO) tool. Therefore, it can be said that the evaluation was carried out on the realistic road topologies. However, the network model is not realistic since the WAN delay is assumed as a constant value.

Wang *et al.* proposed a federated offloading scheme and a distributed algorithm to minimize the service time on VEC systems [16]. The federated offloading scheme divides the tasks into three parts; one part for the local computation, one part for the edge server through the V2I communication, one part for the idle vehicles through the V2V communication. The allocation ratio among these three parts and the offloading order considering the edge servers and neighbor vehicles affect the total latency. The authors formulate the communicational and computational models used in their scenario and solve this latency-minimization problem. The distributed algorithm is operated to obtain an optimal route among the neighboring vehicles. This algorithm uses a greedy method to find a vehicle that promises the lowest service time. We think the most challenging part of the federated offloading scheme is to divide the tasks at different rates. This rate can be adjusted in the optimization formula easily, but it will be challenging to apply in real life.

Xu *et al.* state that smart vehicle applications require computation-intensive tasks [17]. Therefore, they propose a computation offloading method named V2X-COM. This method defines the task offloading process as a multi-objective optimization problem and uses a genetic algorithm to decrease the latency while increasing the resource utilization. V2X-COM offloads the task either to the edge servers or the other vehicles. Since the demand of vehicles is not known in advance, the task offloading is an online problem which requires a low-complexity solution. The complexity of the genetic algorithms is known to be high; hence it may not be sufficiently fast for this problem.

Dai *et al.* address that deciding the target VEC server and determining the offloading ratio is an NP-hard problem [18]. They formulated the VEC server selection and load balancing problem as a mixed-integer nonlinear programming problem, then propose a low-complexity approximation algorithm to solve this problem. The algorithm runs on the vehicles in a distributed manner. In their model, some parts of the tasks can be executed locally on the vehicle, and the rest can be offloaded to the VEC server. Dividing the tasks and combining the results may not be favorable for some applications which require the response in a real-time manner. For example, an image processing based accident prevention application needs to receive a danger signal as quickly as possible.

Feng *et al.* propose a hybrid vehicular cloud (HVC) framework to enhance the computing capacity of vehicles by using computational units of the neighboring vehicles, RSUs and centralized cloud [19]. The objectives of their online algorithm are increasing the number of successfully executed tasks and reducing cellular network usage. The first aim of this algorithm is to find the idle slots on the neighbor vehicles and RSUs by considering the estimated transmission and execution delays. They use the cellular network to access the cloud as a secondary option. If there are no idle slots on the RSUs or neighboring vehicles, the task is processed at the cloud using the cellular network. All the devices in the system work collaboratively by broadcasting a beacon message. The tasks are scheduled consecutively based on their estimated transmission and processing time. Their work assumes that the devices send consistent data to each other. However, misleading information provided by a vehicle will cause subsequent tasks to fail.

III. ML-Based Vehicular Workload Orchestrator

Since the service requests are created one after the other in a dynamic fashion, workload orchestration is an online problem and cannot be solved by the formal optimization tools like CPLEX and Gurobi. Instead, we propose an ML-based orchestrator that tries to maximize the percentage of satisfied services by dynamically changing network conditions and server utilizations.

A VEC system is typically composed of a multitude of edge servers accompanied by cloud resources. Similarly, the communication infrastructure required for VEC incorporates WLAN, MAN, and WAN technologies. Heterogeneous infrastructure combined with intermittent and bursty task requests creates a substantially dynamic scene in terms of task processing. In this setting, the edge orchestrator tries to maintain an efficient operation of the overall system by deciding on the best available computational unit to offload. This decision is considerably challenging as it should take into account both computational and networking states as well as the workload characterization.

A. Multi-Tier Vehicular Edge Computing Architecture

The access technologies used in the VNs and the topology of infrastructure have a considerable effect on the system performance. We think that future smart highways will use the advanced applications on the multi-tier multi-access VEC architectures, which include both edge and cloud servers, as shown in Fig. 3. In our architecture, the vehicles can offload

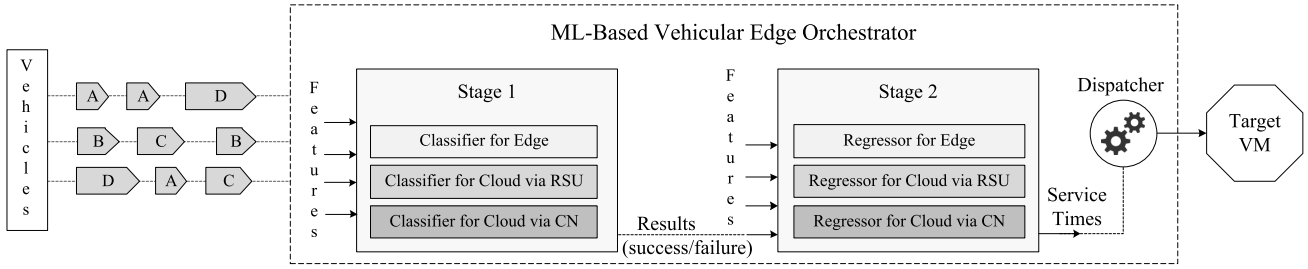


Fig. 2. Two-stage ML-based vehicular edge orchestrator.

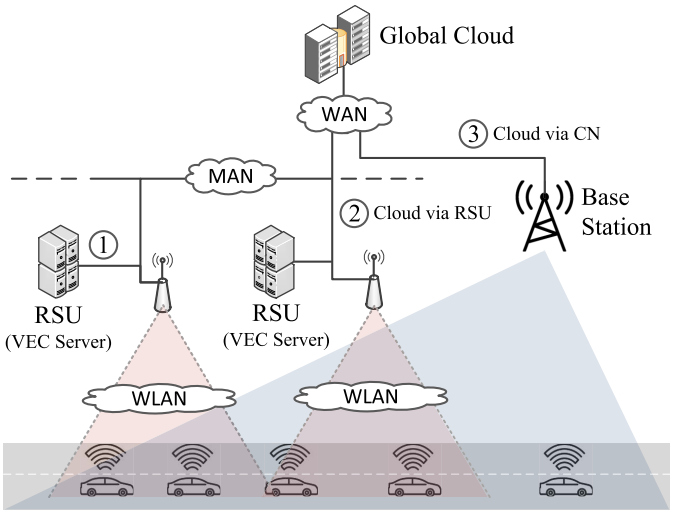


Fig. 3. Multi-tier multi-access VEC architecture and task offloading options.

their tasks to the edge servers by using the WLAN interface. We use WLAN as a generic term representing the short-range wireless access technology. In our simulation study, we prefer to use the IEEE 802.11p protocol for V2R communication. A similar MEC-based vehicular network model is proposed in other studies as well. Peng and Shen [28] and Sami *et al.* [29] use a cellular network for V2I communication and the IEEE 802.11 family protocol for V2R communication.

A task can also be offloaded to the cloud servers via the Internet connection (WAN), which offers the possibility of providing more flexible network infrastructure. In the proposed architecture, there are two alternatives for the broadband connection, one via RSU and the other via the cellular network. The RSUs can access to the cloud resources in a multi-hop manner by using the cellular base station as proposed in [30]. However, this approach creates a significant bottleneck on the base stations. In our proposal, the RSUs use a fiber communication link to provide the Internet connection. Offloading to edge, cloud via RSU, and cloud via cellular network (CN) scenarios are illustrated in Fig. 3 as cases 1, 2, and 3 respectively.

The RSUs are also connected to a metropolitan area network (MAN) in the proposed architecture. The MAN connectivity allows RSUs to share their computational capacity via task migration. Therefore, the VEC layer can be thought of as a shared resource pool. In addition, MAN offers a solution to the handover problem. We use the same handover approach proposed in [15]. If the vehicle leaves the range of the serving RSU before the offloaded task is executed, the result is transmitted to the related vehicle in a multi-hop manner through the other RSUs. The handover operation only fails when the vehicle leaves the range of the serving RSU while uploading or downloading data. Although the network delay on MAN is very small and commonly ignored, we take this delay into consideration.

B. Two-Stage ML-Based Vehicular Edge Orchestrator

As already stated, using the traditional approaches for the workload orchestration cannot provide the desired performance on the dynamic environments where the state of resources changes rapidly. Therefore, different techniques such as ML or the other AI-based solutions are recommended for the workload orchestration [26]. In our work, we propose an ML-based orchestrator that has two-stages to decide where to offload, as shown in Fig. 2. In the first stage, three classification models are used for each option to estimate the result (success or failure) of the offloading process. In the second stage, the service time is estimated by using related regression models for the options predicted to be successful in the first stage. We performed a simulation experiment to collect the training data via EdgeCloudSim [31]. A random orchestration policy is applied under variable load by recording various features that can be used in the classification and regression models. Then, the possible features were investigated, and the ones which have more impact on the result of algorithms were selected. A demonstrative example of a two-stage ML-based vehicular edge orchestrator is illustrated in Fig. 4. As shown in Fig. 4, offloading to the edge and cloud via CN options are predicted to be successful in the first stage; hence they are selected as the initial candidates. The cloud via CN option is determined as the best candidate in the second stage since it promises better service time value than the edge option.

We opted to utilize a two-stage architecture because the success/failure and the service time of a task are related but different metrics. The service time is meaningful and defined only if a task is successfully executed. It is possible to train a single model that could infer both metrics by assigning large ground truth service time values for failed tasks; however, that would mean alteration of the ground truth, which could introduce artifacts in the data, and would produce an unnecessarily complex model with potentially worse generalizability. Dividing the problem into two sub-problems allows us to use two simpler models that are easier to tune and generalize.

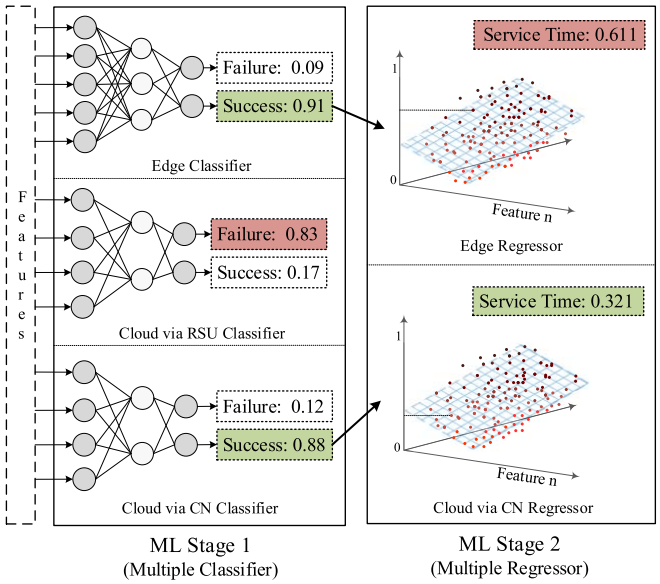


Fig. 4. Demonstrative example of the ML-based orchestrator stages.

Another advantage is to be able to use different input features for the two problems, which may indeed be impacted by different criteria.

In essence, ML-based solutions have already been studied in various domains in recent years. Our two-stage ML-based vehicular edge orchestrator aims at increasing the number of successfully offloaded tasks and reducing the service time as much as possible. The first stage predicts whether offloading to a specific location would result in failure. That location could be eliminated from the decision-making in the second stage, which predicts the service time. As a result, although the classification and regression models are already used in many areas, the novelty of our work is using these models successively for the vehicular edge orchestration problem.

C. Classification Models

The classification model is used to forecast whether the offloaded task to a particular processing unit will be successful or not. The task offloading can be considered as successful if the task execution and transmission (including both the upload and download cases) succeed. Considering the overall system performance of the cases studied, predicting the result of the offloading operation has more impact than predicting the service time. Therefore, the most crucial stage of the proposed ML-based edge orchestrator can be considered as the classification stage. We inspected three different classification algorithms, which are naive Bayes (NB), Support Vector Machine (SVM) and multilayer perceptron (MLP). All the employed models are supervised, i.e., the ground truth class labels of the instances are used to train the models.

NB is a probabilistic classifier that is based on the Bayes theorem [32]. Each class is represented with a multivariate Gaussian probability with respect to the features, with the assumption of independence between the features. Hence the covariance of the fitted Gaussians is diagonal.

SVM is a binary (two-class) classifier that aims to determine a separating hyperplane between the classes that maximizes its margin to the class instances. Hence it uses a solver to determine this discriminating hyperplane. We specifically use John Platt's sequential minimal optimization (SMO) algorithm [33]. The SVM classifier can be used to determine non-linear discriminant hyperplanes via a technique called kernel trick, which implicitly maps the input features to higher dimensional spaces. However, to keep the model simple and improve its generalizability, we utilize a linear kernel. The C parameter (regularization parameter, which determines the model's complexity) is selected to be 1.

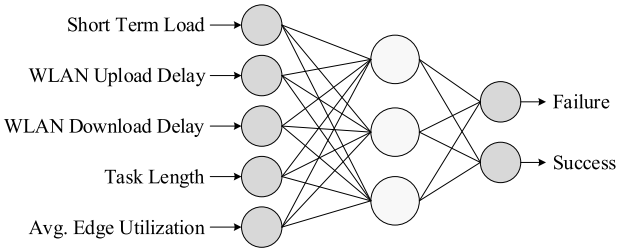


Fig. 5. MLP neural network of the *edge* classifier.

TABLE II
MLP MODEL PARAMETERS

Parameter	Value
Number of hidden units	2 and 3
Learning rate for the back-propagation algorithm	0.1
Momentum rate for the back-propagation algorithm	0.2
Number of epochs to train through	1000

MLP is an artificial neural network which has an input/output layer and an arbitrary number of hidden layers between them [34]. The hidden layers are composed of a multitude of neurons that implement the sigmoid activation function. Hence they can model non-linear relationships between the target classes and input features. The connections between neurons are assigned weights that are learned via the backpropagation algorithm, which is based on the gradient descent algorithm. The learning rate parameter determines the amount of model change allowed on each epoch (iteration), while the momentum parameter helps the algorithm to avoid converging on local minima. The settings of the MLP neural network used for "edge", "cloud via RSU" and "cloud via cellular" cases are similar. We use an MLP model with a fully connected single hidden layer, i.e., the neurons on the hidden layer are connected to all the neurons on the input and the output layers. As an example, the MLP neural network of the "edge" classifier is depicted in Fig. 5. The specific parameters of the utilized MLP models are given in Table II.

We inspected the true positive (T_p), true negative (T_n) rates, and their weighted average values to compare the classification algorithms. The classifiers used for "cloud via RSU" and "cloud via cellular" scenarios have high precision, but the classifier used for "edge" has lower precision since predicting the result of tasks offloaded to the VEC server is the most complex problem. The performance comparison of the classifiers used for "edge" is shown in Fig. 6. SVM and NB provide higher T_p rate, but the T_n rate is not satisfactory. These classifiers predict that the offloading result will be frequently successful. MLP gives more balanced results than its competitors, and it has a

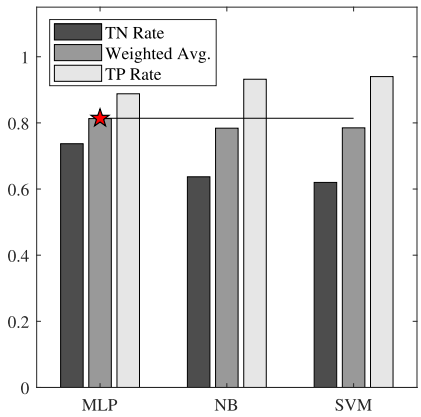


Fig. 6. Performance comparison of the classifiers used for the edge scenario.

TABLE III
COMPARISON OF TRAINING DATA SETS USED BY THE *Edge* CLASSIFIER

Data Set	T_p	T_n	Recall	Precision	F-Score
unbalanced	0.999	0.017	0.905	0.877	0.863
balanced	0.763	0.850	0.806	0.809	0.806
with VD	0.750	0.869	0.810	0.814	0.809
with STL	0.737	0.888	0.813	0.820	0.812
VD + STL	0.745	0.878	0.812	0.817	0.811

higher percentage of correctly classified instances. Therefore, we use the MLP based classifier as the first stage of our ML-based edge orchestrator.

The distribution of the training data is essential for the training phase. The training data set includes more entries for the successfully offloaded tasks. If all the collected data is used, the classifiers tend to predict "success", which yields to a poor T_n rate. Our original training data set includes around 35 million samples, but around 2 million of them belong to the failed tasks. The training data is rearranged randomly in a way to use an equal number of "success" and "failure" cases. This process reduced the number of samples to around 4 million, which includes 250 thousand, 750 thousand, and 1 million samples of failed tasks for the *edge*, *cloud via RSU* and *cloud via CN* scenarios respectively. Balancing the number of samples for the success and failure cases improves the T_n rate significantly, as shown in Table III. After the training data was balanced, the best data set whose elements have the strongest relationship between the inputs and output (correlation coefficient) is found by testing the different sets. We have selected some features based on our previous work and knowledge in this field. For example, the WLAN status, task length, and the average utilization of edge servers have a direct impact on the outcome of offloading to RSUs. On the other hand, the computational parameters do not affect the result of cloud offloading since the cloud servers usually have enough computing resources. Therefore, the network resources such as the WAN and cellular network capacity are taken into account while offloading to the cloud.

In addition to experience-based features, we tried to improve the prediction performance by using additional features, which may have a positive impact on the results. Since the classifiers used for the edge has lower precision than the others, we worked on this model more intensively. The short-term load (STL) and vehicle density (VD) are tested as new features. STL refers to the number of tasks offloaded to the related server in the recent past, such as the last 5 seconds. The load of the servers does not change immediately after making the decision because the offloaded task reaches to the destination after a network delay. So, the STL value can infer a possible load in the future. VD is the number of vehicles used in the simulation. The system can behave differently under different loads, so this value is used to analyze whether it has an impact on the classifier performance. The important performance metrics for different data sets are listed in Table III. Adding the STL and VD features to the training data set also increases the classifier performance. We use the F-Score value, which is commonly used to compare the classifiers. The recall and precision values are used to compute the F-Score value. Related metrics are calculated by

$$precision = (\frac{T_p}{T_p + F_p} + \frac{T_n}{T_n + F_n})/2 \quad (1)$$

$$recall = (\frac{T_p}{T_p + F_n} + \frac{T_n}{T_n + F_p})/2 \quad (2)$$

$$f - score = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (3)$$

Using STL and VD features separately or together provide similar performances, but the "only STL" case provides slightly better F-Score. In addition, finding the density of vehicles on the road may not be easy in real-life scenarios. Therefore, we chose using the balanced data set with the STL feature to prevent the overhead of calculating VD. The list of features used for each classifier is given in Table IV.

TABLE IV
LIST OF FEATURES USED FOR THE CLASSIFICATION MODELS

Classifier	Feature 1	Feature 2 $*$	Feature 3	Feature 4
edge	STL	WLAN delay	task length	avg. edge util.
cloud via RSU	STL	WAN delay	-	-
cloud via CN	STL	CN delay	-	-

$*$ This feature refers two features for up-link and down-link delays.

TABLE V
LIST OF FEATURES USED FOR THE REGRESSION MODELS

Regressor	Feature 1	Feature 2	Feature 3
edge	task length	Avg. edge util.	-
cloud via RSU	task length	WAN upload delay	WAN download delay
cloud via CN	task length	CN upload delay	CN download delay

D. Regression Models

The regression model is used to predict the service time of the offloaded task. A linear regression model is utilized to predict the service time because the output variable (service time) is a linear combination of our inputs (features). We applied a similar study as we did for the classification models to find the best feature list for the regression models. The list of features used for each regressor is given in Table V.

IV. PERFORMANCE EVALUATION

The performance of the proposed ML-based workload orchestrator is evaluated through an extensive number of experiments. The simulations are performed on

EdgeCloudSim, which is capable of modeling both the computational and networking resources as well as the mobile vehicles [31].

A. Competitor Algorithms

In order to benchmark the proposed solution, we implement three competitor algorithms, which are the *random*, *simple moving average* (SMA) based, the multi-armed bandit (MAB) theory-based, and game theory-based vehicular edge orchestrators. The random orchestrator selects the target server to offload in a probabilistic manner where the probability of selecting all targets is the same. This strategy is used to present the worst-case scenario. SMA is a time series forecasting method that uses short-term historical data to estimate future outcomes. This algorithm uses a fixed-length list to record the time series data, wherein each element of the list contains the data for a specific period. It calculates the average success rate of each option by using a higher weight for the data, which is collected more recently. Then it selects the target machine that has given the highest success rate. The pseudocode of the SMA based workload orchestrator is given in Algorithm 1 where t is the incoming task, C refers to the candidate processing units, S is the list of statistical data, l_S is the length of list S, I_e is the index of the element e of list S, and $SR_{c,w}$ is the average success rate of candidate c for the element (time period) e.

Algorithm 1 SMA Based Vehicular Edge Orchestrator

for all $t \in T$ **do** ▷ For all Tasks
  for all $c \in C$ **do**
    $SCR_c \leftarrow$ GETSCORE(c)
  end for
  Offload the task to candidate c_t, such that:
  $c_t \leftarrow \max_{\forall c \in C} SCR_c$
  Update statistic list S for c_t
end for

procedure GETSCORE(c)
  $totalScore \leftarrow 0$
  for all e in S **do**
    $weight \leftarrow l_S - I_e$
    $score \leftarrow SR_{c,w} * weight$
    $totalScore \leftarrow totalScore + score$
  end for
  Return $totalScore$
end procedure

MAB is a reinforcement learning approach that tries to maximize the expected gain. We implemented a MAB theory-based orchestrator for our multi-access multi-tier VEC architecture based on Sun *et al.*'s work [13]. In their solution, the candidate machines to offload are the neighboring vehicles, while in our solution, the candidates are the edge servers and cloud server. In addition, the cloud server can be accessed via two ways in the proposed architecture, one is the WAN accessed through RSUs, and the other one is the cellular network. Therefore, we have three candidates for the task offloading, which are "edge", "cloud via RSU" and "cloud via CN" as explained in Section III. Although the candidates seem to be different, the problems solved in both studies are very similar.

Algorithm 2 MAB Theory-Based Vehicular Edge Orchestrator

for all $t \in T$ **do**
  if Any candidate $n \in N$ has not selected yet **then**
    Select Candidate n
    $\bar{u}_{t,n} \leftarrow d_{sum}(t,n)/x_t$
    $k_{t,n} \leftarrow 1$
  else
    Observe x_t, calculate $\tilde{x_t}$
    Calculate utility func. of each candidate $n \in N$

$$\tilde{u}_{t,n} \leftarrow \bar{u}_{t-1,n} - \sqrt{\frac{\beta(1-\tilde{x_t})ln(t)}{k_{t-1,n}}} \quad (4)$$

    Offload the task to candidate a_t, such that:
    $a_t \leftarrow \underset{n \in N}{argmin}\ \tilde{u}_{t,n}$
    Observe the sum offloading delay $d_{sum}(t,n)$

$$\bar{u}_{t,a_t} \leftarrow \frac{\bar{u}_{t-1,a_t} k_{t-1,a_t} + d_{sum}(t,n)/x_t}{k_{t-1,a_t}+1} \quad (5)$$

    $k_{t-1,a_t} \leftarrow k_{t-1,a_t} + 1$
  end if
end for

The strategy used to solve the MAB problems varies. The upper confidence bound (UCB) algorithm is one of the most widely used strategies. Sun *et al.* use a UCB strategy with a slight change. They added the input-awareness and occurrence-awareness to the UCB formula and named it as the "adaptive volatile UCB" (AVUCB). We use the same formula without the occurrence-awareness as given in Algorithm 2 where t refers to the time, $d_{sum}(t,n)$ is the service time of candidate n at time t, x_t is the length of task, $\tilde{x_t}$ is the normalized task length, $u_{t,n}$ is the sum delay of offloading one instruction, $k_{t,n}$ is the number of tasks that have been offloaded to candidate n up till time t, β is a constant factor, used 1 in the simulations, and a_t is the candidate with the minimum utility value.

The input-awareness is used to eliminate the imbalance utility function score of the small and large tasks. Equation 5 in Algorithm 2 uses x_t to achieve the input-awareness. According to our observations, the input-awareness significantly improves the orchestrator performance. The occurrence-awareness is used to provide equal chances to the serving vehicles. Since the neighbors of the vehicles can frequently change due to the mobility, AVUCB uses a different time value for each serving vehicle. However, in our architecture, vehicles always have three candidates, so this concept is not necessary for us. We use $ln(t)$ in Equation 4 of Algorithm 2, while AVUCB uses $ln(t - t_n)$ where t_n is the occurrence time of each serving vehicle.

The game theory-based algorithm uses a multi-user non-cooperative computation offloading game to adjust the offloading probability of vehicles to achieve the maximum utility. We implemented the computation offloading game based on

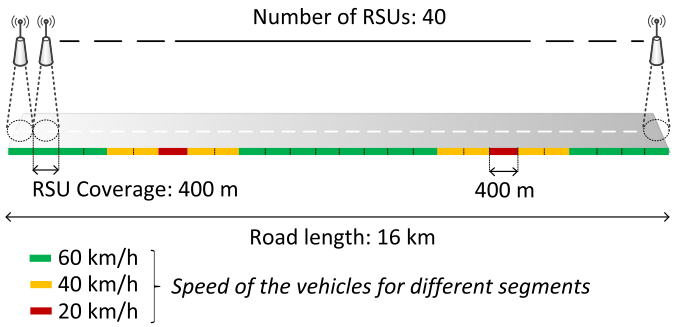


Fig. 7. Vehicular mobility model in the simulation.

Wang *et al.*'s work [14]. They use a best-response strategy, which is given in Equation 8. This strategy uses the latest offloading probabilities of other vehicles to converge to a unique Nash equilibrium. In Wang *et al.* model, the tasks can be executed on the vehicles locally or on the edge servers remotely. They try to find the probability of offloading to edge servers. However, our model has three remote offloading options, as explained in Section III. In order to use their game model, we decrease our options as edge or cloud offloading. If the model decides to execute a task on the cloud server, we prefer either the cellular network or high-speed network provided by the RSU according to the bandwidth they promised. The modified game theory-based vehicular edge orchestrator given in Algorithm 3 where x_t is the length of the task, C_e and C_c are the capacity of edge and cloud servers, U_e and U_c are the average utilization of edge and cloud servers between 0 and 1, $d_{i,max}$ is the maximum delay for the task generated by vehicle i, $d_{i,e}$ is the estimated delay on edge for the task generated by vehicle i, $d_{i,c}$ is the estimated delay on cloud for the task generated by vehicle i, ρ is the pricing factor, λ_i is the task arrival rate of vehicle i, and p_i is the probability of offloading to cloud for the task generated by vehicle i. The pricing factor ρ is used to adjust the degree of willingness to cloud offloading. A smaller pricing factor increases the possibility of uploading to the cloud server. In our simulations, the pricing factor is selected as 0.6. The task arrival rate is utilized as $\lambda \in [0.5, 0.9]$ in the original work. However, this value varies between 3 and 15 in our experiments. Therefore, we normalize λ between 0.15 and 0.75 to avoid p_i of going to positive or negative infinity.

B. Simulation Setup

The road and mobility model used in the simulations is implemented on EdgeCloudSim, as shown in Fig. 7. To simulate the vehicular mobility more realistically, the road is divided into segments, and a dynamic velocity value for the vehicle position is used. Therefore, the speed of the vehicles varies at each segment to differentiate the traffic density on the road. The hotspot locations, which are shown with red color, occur due to the traffic jam. 100 to 1800 vehicles are distributed to random locations when the simulation is started. Then they move in a single direction with a predefined speed with respect to the crossed segment. The road is modeled as a circular route to keep the number of vehicles the same during the simulation.

Algorithm 3 Game Theory-Based Vehicular Edge Orchestrator

1: Initialization: offloading probability vector p, pricing vector ρ, and task arrival rate vector λ.
2: **for all** $t \in T$ **do**
3: $i \leftarrow VehicleIndex$ ▷ t generated by i^{th} vehicle
4: Estimate transmission delay $d^t_{i,e}$ and $d^t_{i,c}$ of task for WLAN and WAN
5: Estimate processing delay $d^c_{i,e}$ and $d^c_{i,c}$ of task on edge and cloud:

$$d^p_{i,e} \leftarrow \frac{x_t}{C_e}(1 - U_e) \tag{6}$$

$$d^p_{i,c} \leftarrow \frac{x_t}{C_c}(1 - U_c) \tag{7}$$

6: Calculate total delay $d_{i,e}$ and $d_{i,c}$ of task on edge and cloud:
7: $d_{i,e} \leftarrow d^p_{i,e} + d^t_{i,e}$
8: $d_{i,c} \leftarrow d^p_{i,c} + d^t_{i,c}$
9: Update the best-response strategy:

$$p_i = \left[\frac{d_{i,e} - d_{i,c}}{2\rho d_{i,max}(1 - \prod_{i \neq j}(1 - \lambda_j p_j))} \right]_0^1 \tag{8}$$

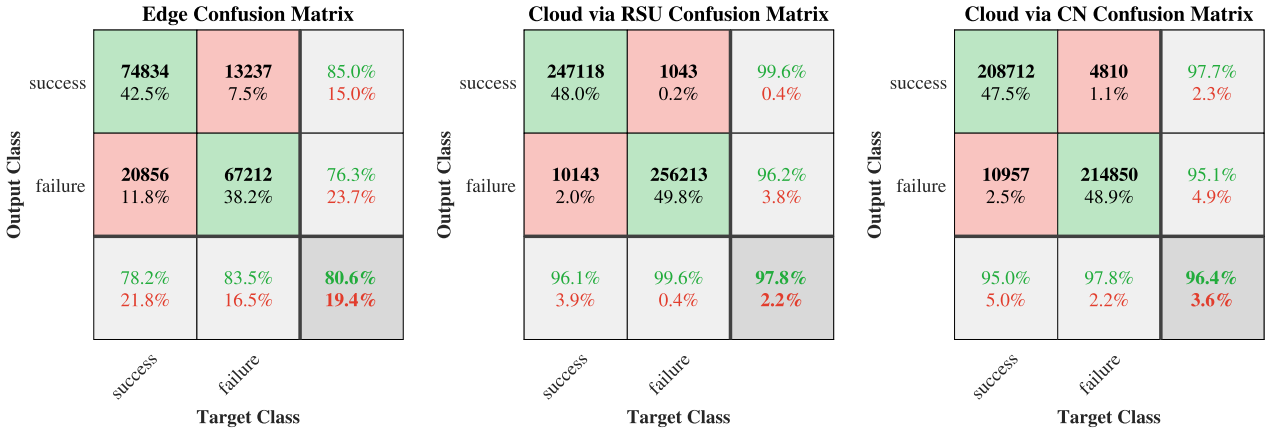
10: **end for**

The vehicles offload some of their tasks to the other computing units. It is assumed that the vehicles handle some operations locally, but some tasks are offloaded to a remote server if the OBU is not capable of executing the related tasks. The workload orchestrator takes care of the tasks which are decided to be offloaded to a remote server, which can be an edge or a cloud server. Therefore, the OBU execution is not addressed in this study. The edge servers are provided by the RSUs, which also operates an access point. The transmission between the vehicles and RSUs is performed over an 802.11p WLAN. According to Lin *et al.*'s measurements, the 802.11p protocol can achieve around 10 Mbps data rate using 16QAM for the V2I communication [35]. Therefore, we use a 10 Mbps data rate for the 802.11p interface. All RSUs are connected to a high-speed network with Internet access that can be provided through conventional infrastructures such as a fiber network. The vehicles can offload their tasks to the cloud server through the RSU or the cellular network provided by the LTE base station. Xu *et al.* carried out the experiments to evaluate the performance of 4G-LTE used by the connected vehicles [36]. Based on their observations, we use a 20 Mbps data rate for the LTE cellular network.

We use three different applications that run on the vehicles to generate tasks at different rates. The applications have different characteristics, such as the inter-arrival times of tasks, the length of tasks, and the data size to upload or download. The characteristics of the applications are given in Table VI, and the other simulation parameters are listed in Table VII.

C. Simulation Results

Before presenting the comparative evaluation of the algorithms, the performance of the classifiers used in the first



(a) Confusion Matrix for edge (b) Confusion Matrix for cloud via RSU (c) Confusion Matrix for cloud via CN

Fig. 8. Confusion matrix of the classifiers.

TABLE VI
APPLICATION CHARACTERISTICS

	Navigation App	Danger Assessment	Infotainment App
Usage Ratio* (%)	30	35	35
Task Interarrival time (sec)	3	5	15
Max Delay Requirement (sec)	0.5	1	1.5
Delay Sensitivity (sec)	0.5	0.8	0.25
Upload/Download Data (KB)	20/20	40/20	20/80
Task Length (GI)	3	10	20
RSU/Cloud VM Utilization(%)	6/1.6	20/4	40/8

* Percentage of the vehicles requesting related service.

TABLE VII
SIMULATION PARAMETERS

Parameter	Value
Simulation Time/Warm-up Period	60/3 minutes
Number of Repetitions	100
Network Delay Model	MMPP/M/1 queue model
# of VMs per Cloud/RSU	20/40 (2 per RSU)
Capacity of Cloud/RSU VM	150/20 GIPS
Range of RSU (WLAN)	200 meters
WLAN/MAN Bandwidth	10/1000 Mbps
WAN/WAN over LTE Bandwidth	50/20 Mbps
WAN/WAN over LTE Propagation Delay	150/160 ms

stage of our ML-based workload orchestrator is addressed. The confusion matrix of each classifier is shown in Fig. 8. The green boxes in the confusion matrix correspond to correctly classified observations, while the red boxes show the misclassified instances. The columns on the far right of the plot show the precision for each class. The rows at the bottom of the plot show the recall for each class. The cell in the bottom right of the plot shows the overall accuracy. As already stated, it is more difficult to predict the result of a task offloaded to the edge server. Therefore, the MLP based classifier for edge provides the lowest accuracy while it is adequate for the other cases.

The failure rate is one of the most critical performance criteria. A task cannot be executed if the utilization of a VM is too high to accept the offloaded tasks, or if the network bandwidth is not sufficient to transfer either the input or the output of the task in a reasonable amount of time. Therefore, a task can fail due to the lack of computational or networking resources in our simulation. The mobility also leads to the task

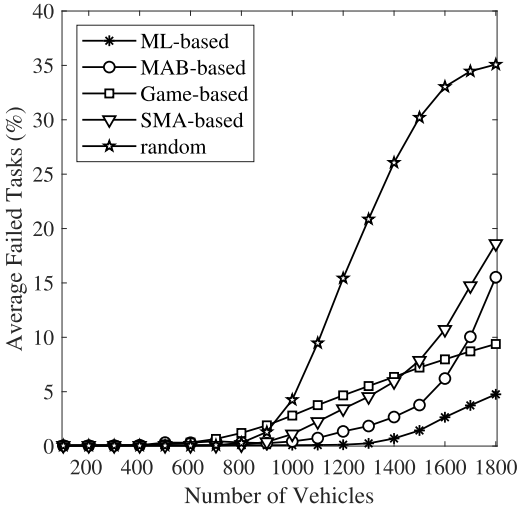
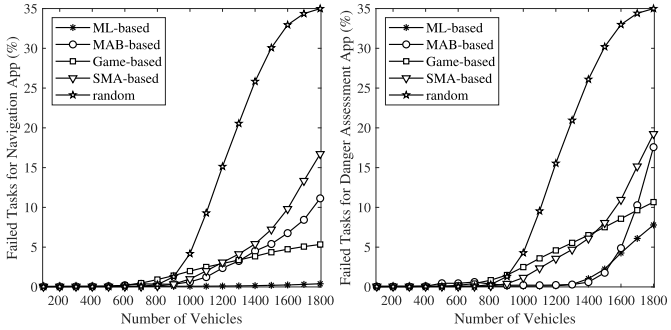


Fig. 9. Percentage of failed tasks (%).

failure, but this case is not addressed since it accounts for a small proportion of the total unsuccessful operations. The average number of failed tasks with respect to the number of vehicles is shown in Fig. 9. The ML-based approach outperforms its competitors because the other approaches have an exploration phase in which some wrong decisions are made to explore the current situation of the system and make more accurate decisions accordingly. Especially after 1500 vehicles, the system is overloaded, and even a small number of wrong decisions adversely affect the performance. The ML-based solution provides superior results for the small tasks which are generated by the navigation application. The inter-arrival time of the small tasks is shorter than the other tasks, so the navigation application has more impact on the average failed task performance.

The results for small and relatively large task sizes are shown in Fig. 10. Fig. 10(a) presents the percentage of failed tasks for the navigation application, while Fig. 10(b) presents the same results for the danger assessment application. According to our observations, the ML-based algorithm tends to send the small tasks to a nearby edge server and the larger to a remote cloud server. The MAB-based algorithm is also a load-aware solution that takes care of the task size.



(a) Navigation app. score (b) Danger assessment app. score

Fig. 10. Average failed tasks for different application types.

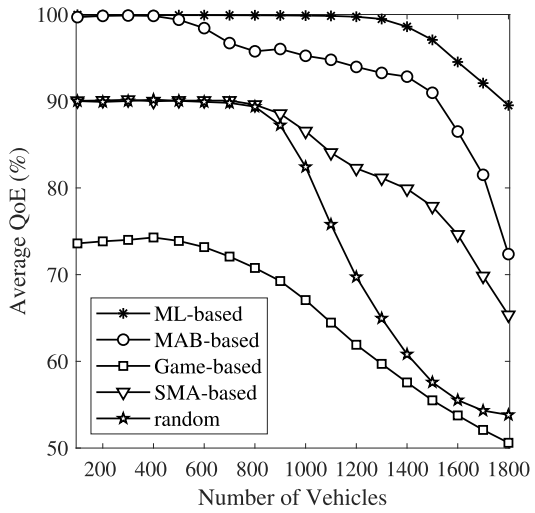


Fig. 11. Average Quality of Experience (QoE).

However, it does not work well when the system becomes congested, and consequently, the loss of tasks begins. The game theory-based algorithm offloads more of the task to edge servers regardless of the task type. The best-response strategy used in the game model uses the predicted service time values as the input variables and does not have any feedback mechanism to evaluate its decision. Almost all of the task failures occur due to the lack of the edge server capacity. Therefore, the task loss situation is observed after 600 vehicles and increases linearly. The SMA-based algorithm does not consider the task size. In particular, it increases the probability of selecting a computing unit that has recently yielded better results. Hence, this strategy results in more task loss when the system is overloaded.

The average service time score of the successfully offloaded tasks can mislead an evaluator in the delay-loss systems. For example, we cannot say that the performance is good in a scenario where the orchestrator provides an adequate service time while losing most of the tasks. Therefore, we use a Quality of Experience (QoE) formula, which considers both the service time and task loss as defined in Equation 9.

$$QoE_i = \begin{cases} 0, & \text{if } T_i \geq 2R_i \\ (1 - \dfrac{T_i - R_i}{R_i}) \cdot (1 - S_i), & \text{if } R_i < T_i < 2R_i \\ 1, & \text{if } T_i \leq R_i \end{cases} \quad (9)$$

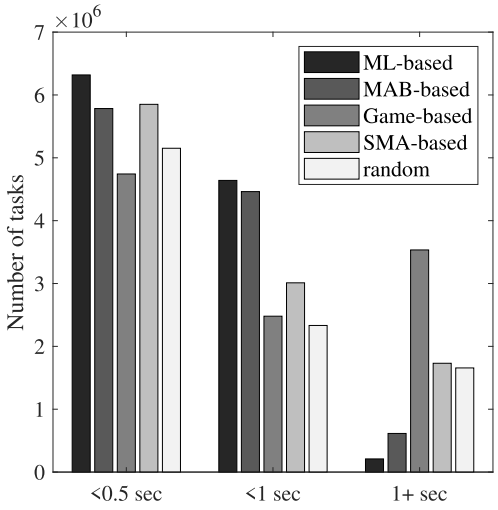
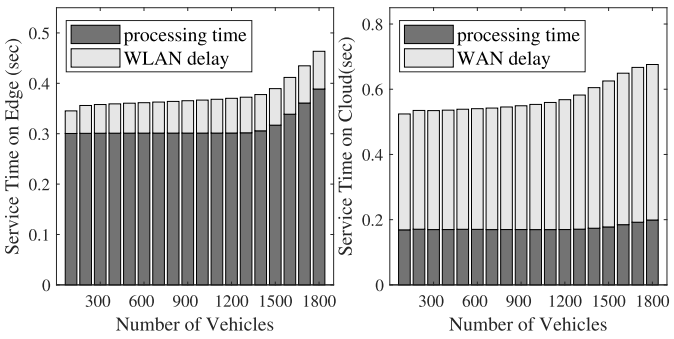


Fig. 12. Histogram of service times.

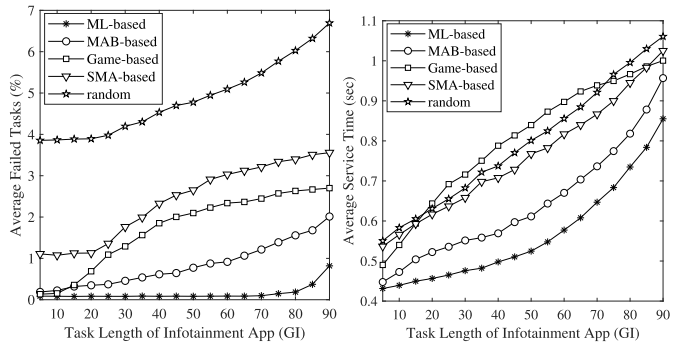
where, T_i is the actual service time, R_i is the delay requirement, and S_i is the delay sensitivity of task i. The delay requirement is used to declare the maximum service time allowed for the related task. The average QoE value decreases when task i is completed later than R_i. If the service time exceeds twice the expected value, the QoE value is set to 0. The delay sensitivity refers to the delay tolerance. This value is between 0 and 1, and higher for the delay intolerant applications. The average QoE with respect to the number of vehicles is shown in Fig. 11. We determined the delay requirements (in proportion to the task sizes) and delay sensitivities of tasks as in Table VI. The ML-based and MAB-based approaches provide the maximum QoE (100%) when the number of vehicles is low, e.g. less than 500, because their goal is to minimize the service time. However, the random and SMA-based approaches do not aim to minimize the service time, so their maximum QoE is around 90% even if there is no task failure. In general, the ML-based method provides the best performance in terms of QoE. The MAB-based solution does not meet the delay requirement criteria after the number of vehicles exceeds 500. The main reason for this problem is that the MAB-based approach sends some large tasks to the edge servers and exceeds the maximum service time allowed for these tasks. On the other hand, the ML-based solution mostly sends large tasks to the cloud server and does not cause congestion in the edge layer. The Game-based solution provides the worst QoE performance because the main objective of the game model is not minimizing the service time; in fact, it tries to converge to a stable equilibrium.

The proportion of the service times of successfully completed tasks is shown in Fig. 12. Since the smaller service time infers the better performance, it is better to have a higher value on the left side of the graph. The ML-based solution provides the best performance in terms of the service time because it has more tasks that are completed under 0.5 seconds and fewer tasks that take more than 1.5 seconds to execute. With a similar perspective, the performance of the other algorithms is ranked from the best to the worst as the MAB-based, SMA-based, random, and Game-based.



(a) Service times on edge servers (b) Service times on cloud servers

Fig. 13. Service time ratio of the edge and cloud servers.



(a) Average failed tasks (%) (b) Average service time (sec)

Fig. 14. Comparative evaluation for different task length.

Since the Game-based model offloads more of the task to the edge server, it causes significant congestion on edge servers. The service time includes both the computational and networking delay. The ratio of these delays is shown in Fig. 13. Fig. 13(a) gives the service time ratio of edge servers, whereas Fig. 13(b) presents the same result for the cloud servers. As expected, the significant proportion of the service times on the edge servers occur due to the processing delay. On the other hand, cloud servers execute tasks in a fast manner, but it takes more time to transmit the data to the cloud servers.

We increase the load on the system by altering the number of vehicles used in the simulation. The load can also be generated by other methods, such as increasing the task arrival rate and task length. As an example, we performed an experiment by changing the task length of the infotainment application. The task length of this application was 20 GIPS in our detailed experiments. We changed this value between 5 and 90 GIPS in a scenario with 1000 vehicles. The average task failure and the service time performances are shown in Fig. 14. It can be seen that the performance decreases as the task length increases as expected. The proposed ML-based approach again provides the best score in this experiment.

V. Conclusion

Orchestrating the dynamic and heterogeneous resources in the VEC systems is a challenging task. The vehicular workload orchestrator promises to offload the incoming tasks to the optimal computing unit to improve the system performance. Since the workload orchestration is an online problem, the formal optimization tools cannot solve this problem efficiently, since the inputs of the problem is very rapidly changing. In this study, an ML-based workload orchestrator for multi-access multi-tier VEC architectures is proposed. It can handle the uncertain nonlinear systems efficiently by considering multiple criteria. In essence, ML algorithms are used to solve different problems in different areas. The novelty of our work is using the existing algorithms for vehicular edge orchestration problem in two stages as described. We also implemented SMA-based, MAB theory-based and Game theory-based approaches to compare the proposed solution. Based on the extensive simulation results, the ML-based workload orchestrator outperforms its competitors in terms of the average task failure rates and quality of experience wihch includes the service times. As a future work, we plan to optimize the features used in the classification and regression stages of the ML-based workload orchestrator by using an intelligent optimization algorithm. In addition, we want to implement a deep learning-based approach to solve the workload orchestration problem. Finally, we want to enhance our problem by enabling a V2V offloading in which the vehicles have an on-board computation unit.

References

[1] R. I. Meneguette, A. Boukerche, and A. H. M. Pimenta, "AVARAC: An availability-based resource allocation scheme for vehicular cloud," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 10, pp. 3688–3699, Oct. 2019.

[2] J. Arbib and T. Seba, *Rethinking Transportation 2020–2030* (RethinkX Sector Disruption). New York, NY, USA: Springer, 2017. [Online]. Available: https://books.google.com.tr/books?id=lLMtswEACAAJ

[3] T. Litman, *Autonomous Vehicle Implementation Predictions: Implications for Transport Planning* (DesLibris: Documents Collection). Victoria, BC, Canada: Victoria Transport Policy Institute, 2013. [Online]. Available: https://books.google.com.tr/books?id=G3FKnwEACAAJ

[4] E. Curry and A. Sheth, "Next-generation smart environments: From system of systems to data ecosystems," *IEEE Intell. Syst.*, vol. 33, no. 3, pp. 69–76, May 2018.

[5] R. F. Atallah, C. M. Assi, and M. J. Khabbaz, "Scheduling the operation of a connected vehicular network using deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 5, pp. 1669–1682, May 2019.

[6] K. Zhang, Y. Mao, S. Leng, S. Maharjan, and Y. Zhang, "Optimal delay constrained offloading for vehicular edge computing networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[7] Y. F. Payalan and M. A. Guvensan, "Towards next-generation vehicles featuring the vehicle intelligence," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 1, pp. 30–47, Jan. 2020.

[8] D. Grewe, M. Wagner, M. Arumaithurai, I. Psaras, and D. Kutscher, "Information-centric mobile edge computing for connected vehicle environments: Challenges and research directions," in *Proc. Workshop Mobile Edge Commun.*, New York, NY, USA, 2017, pp. 7–12.

[9] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proc. IEEE*, vol. 107, no. 8, pp. 1697–1716, Aug. 2019.

[10] J. C. Nobre *et al.*, "Vehicular software-defined networking and fog computing: Integration and design principles," *Ad Hoc Netw.*, vol. 82, pp. 172–181, Jan. 2019.

[11] W. Tong, A. Hussain, W. X. Bo, and S. Maharjan, "Artificial intelligence for Vehicle-to-everything: A survey," *IEEE Access*, vol. 7, pp. 10823–10843, 2019.

[12] Z. Ning, P. Dong, X. Wang, J. J. P. C. Rodrigues, and F. Xia, "Deep reinforcement learning for vehicular edge computing: An intelligent offloading system," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 6, p. 60, Oct. 2019.

[13] Y. Sun *et al.*, "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3061–3074, Apr. 2019.

[14] Y. Wang *et al.*, "A game-based computation offloading method in vehicular multi-access edge computing networks," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4987–4996, Jun. 2020.

[15] P. Liu, J. Li, and Z. Sun, "Matching-based task offloading for vehicular edge computing," *IEEE Access*, vol. 7, pp. 27628–27640, 2019.
[16] H. Wang, X. Li, H. Ji, and H. Zhang, "Federated offloading scheme to minimize latency in MEC-enabled vehicular networks," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2018, pp. 1–6.
[17] X. Xu, Y. Xue, X. Li, L. Qi, and S. Wan, "A computation offloading method for edge computing with Vehicle-to-Everything," *IEEE Access*, vol. 7, pp. 131068–131077, 2019.
[18] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4377–4387, Jun. 2019.
[19] J. Feng, Z. Liu, C. Wu, and Y. Ji, "Mobile edge computing for the Internet of vehicles: Offloading framework and job scheduling," *IEEE Veh. Technol. Mag.*, vol. 14, no. 1, pp. 28–36, Mar. 2019.
[20] Y. Wang, S. Wang, S. Zhang, and H. Cen, "An edge-assisted data distribution method for vehicular network services," *IEEE Access*, vol. 7, pp. 147713–147720, 2019.
[21] W. Zhan *et al.*, "Deep-Reinforcement-Learning-Based offloading scheduling for vehicular edge computing," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5449–5465, Jun. 2020.
[22] G. Liu, Y. Xu, Z. He, Y. Rao, J. Xia, and L. Fan, "Deep learning-based channel prediction for edge computing networks toward intelligent connected vehicles," *IEEE Access*, vol. 7, pp. 114487–114495, 2019.
[23] Z. MacHardy, A. Khan, K. Obana, and S. Iwashina, "V2X access technologies: Regulation, research, and remaining challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 1858–1877, 3rd Quart., 2018.
[24] J. E. Siegel, D. C. Erb, and S. E. Sarma, "A survey of the connected vehicle landscape—Architectures, enabling technologies, applications, and development areas," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 8, pp. 2391–2406, Aug. 2018.
[25] C. Sonmez, A. Ozgovde, and C. Ersoy, "Fuzzy workload orchestration for edge computing," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 2, pp. 769–782, Jun. 2019.
[26] B. Cao, L. Zhang, Y. Li, D. Feng, and W. Cao, "Intelligent offloading in multi-access edge computing: A State-of-the-Art review and framework," *IEEE Commun. Mag.*, vol. 57, no. 3, pp. 56–62, Mar. 2019.
[27] D. Krajzewicz, "Traffic simulation with SUMO–simulation of urban mobility," in *Fundamentals of Traffic Simulation*. USA: Springer, 2010, pp. 269–293.
[28] H. Peng and X. S. Shen, "Deep reinforcement learning based resource management for multi-access edge computing in vehicular networks," *IEEE Trans. Netw. Sci. Eng.*, early access, Mar. 6, 2020, doi: 10.1109/TNSE.2020.2978856.
[29] H. Sami, A. Mourad, and W. El-Hajj, "Vehicular-OBUs-As-On-Demand-fogs: Resource and context aware deployment of containerized micro-services," *IEEE/ACM Trans. Netw.*, vol. 28, no. 2, pp. 778–790, Apr. 2020.
[30] H. Peng, Q. Ye, and X. Shen, "Spectrum management for multi-access edge computing in autonomous vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 7, pp. 3001–3012, Jul. 2020.
[31] C. Sonmez, A. Ozgovde, and C. Ersoy, "Edgecloudsim: An environment for performance evaluation of edge computing systems," in *Proc. 2nd Int. Conf. Fog Mobile Edge Comput. (FMEC)*, May 2017, pp. 1–6.
[32] G. H. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers," in *Proc. 11th Conf. Uncertainty Artif. Intell.* San Mateo, CA, USA: Morgan Kaufmann, 1995, pp. 338–345.
[33] J. Platt, "Fast Training of Support Vector Machines using Sequential Minimal Optimization," in *Advances in Kernel Methods–Support Vector Learning*, B. Schoelkopf, C. Burges, and A. Smola, Eds. Cambridge, MA, USA: MIT, 1998.
[34] M. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)—A review of applications in the atmospheric sciences," *Atmos. Environ.*, vol. 32, nos. 14–15, pp. 2627–2636, Aug. 1998.
[35] W.-Y. Lin, M.-W. Li, K.-C. Lan, and C.-H. Hsu, "A comparison of 802.11 a and 802.11 p for V-to-I communication: A measurement study," in *Quality, Reliability, Security and Robustness in Heterogeneous Networks*. Berlin, Germany: Springer, 2012, pp. 559–570.
[36] Z. Xu, X. Li, X. Zhao, M. H. Zhang, and Z. Wang, "DSRC versus 4G-LTE for connected vehicle applications: A study on field experiments of vehicular communication performance," *J. Adv. Transp.*, vol. 2017, Aug. 2017, Art. no. 2750452.

Cagatay Sonmez (Member, IEEE) received the B.S. degree in computer engineering from Dokuz Eylul University, Izmir, in 2008, and the M.S. and Ph.D. degrees in computer engineering from Bogazici University, Istanbul, in 2012 and 2020, respectively. He has been working as the Technical Leader of the Research and Development Software Department, Arcelik Electronics, Istanbul, since 2008. His research interests include design and performance evaluation of communication protocols, cloud computing, edge computing, and the IoT.

Can Tunca received the B.S. degree in computer science and engineering from Sabanci University, Istanbul, Turkey, in 2010, and the M.S. and Ph.D. degrees from Bogazici University, Istanbul, in 2012 and 2019, respectively. He is currently working as the Research and Development Manager with Pointr: The Deep Location Company, Istanbul. His research interests include network protocol design, the Internet of Things, body area networks, wearable computing, pervasive healthcare, and indoor localization.

Atay Ozgovde (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees from Bogazici University, Istanbul, in 1995, 1998, and 2009, respectively. He has worked for Nortel Networks as a Research and Development Engineer in various telecommunications projects from 1998 to 2001. In 2002, he started working as a Research Assistant with the Department of Computer Engineering, Bogazici University. He is currently an Assistant Professor with the Department of Computer Engineering, Galatasaray University. His research interests include wireless sensor networks, embedded systems, distributed systems, pervasive computing, SDN, and mobile cloud computing.

Cem Ersoy (Senior Member, IEEE) received the B.S. and M.S. degrees in electrical and electronics engineering from Bogazici University, Istanbul, in 1984 and 1986, respectively, and the Ph.D. degree in electrical engineering from Polytechnic University, Brooklyn, NY, in 1992. He has worked as a Research and Development Engineer with Northern Telecom subsidiary NETAS A. S. Since then, he has been a Professor with the Department of Computer Engineering, Bogazici University. His research interests include 5G and beyond wireless networks, SDN/NFV, multitier edge/cloud, pervasive health with wearables, the IoT, and green networking. He was the Chairman of the IEEE Communications Society Turkish Chapter for eight years.