Novel Lagrange Multipliers-Driven Adaptive Offloading for Vehicular Edge Computing

Liang Zhao, *Member, IEEE,* Tianyu Li, Guiying Meng, Ammar Hawbani, Geyong Min, Ahmed Al-Dubai, *Senior Member, IEEE,* Albert Zomaya, *Fellow, IEEE*

Abstract—Vehicular Edge Computing (VEC) is a transportation-specific version of Mobile Edge Computing (MEC) designed for vehicular scenarios. Task offloading allows vehicles to send computational tasks to nearby Roadside Units (RSUs) in order to reduce the computation cost for the overall system. However, the state-of-the-art solutions have not fully addressed the challenge of large-scale task result feedback with low delay, due to the extremely flexible network structure and complex traffic data. In this paper, we explore the joint task offloading and resource allocation problem with result feedback cost in the VEC. In particular, this study develops a VEC computing offloading scheme, namely, a Lagrange multipliers-based adaptive computing offloading with prediction model, considering multiple RSUs and vehicles within their coverage areas. First, the VEC network architecture employs GAN to establish a prediction model, utilizing the powerful predictive capabilities of GAN to forecast the maximum distance of future trajectories, thereby reducing the decision space for task offloading. Subsequently, we propose an real-time adaptive model and adjust the parameters in different scenarios to accommodate the dynamic characteristic of the VEC network. Finally, we apply Lagrange Multiplier-based Non-Uniform Genetic Algorithm (LM-NUGA) to make task offloading decision. Effectively, this algorithm provides reliable and efficient computing services. The results from simulation indicate that our proposed scheme efficiently reduces the computation cost for the whole VEC system. This paves the way for a new generation of disruptive and reliable offloading schemes.

Index Terms—Vehicular Edge Computing, Result Feedback, Task Offloading, Resources Allocation, Swarm Intelligence.

1 INTRODUCTION

Mobile Edge Computing (MEC) is an emerging computational paradigm that aims to push computing and data storage capabilities towards the network edge in order to be closer to end-users and mobile devices for local and faster execution. It aims to alleviate the computational burden on mobile devices [1]. MEC offers advantages such as reduced network latency, alleviated network congestion, and support for local data processing. Due to these advantages, MEC has vast application prospects and potential [2].

Vehicular Edge Computing (VEC) is an emerging paradigm that aims to transfer computing and data processing capabilities closer to the vehicle in order to provide faster, more dependable, and more secure vehicular applications and services [3]. Task offloading is a key technology in VEC, which allows vehicles to offload partial computational tasks to edge nodes or cloud servers for processing, thereby reducing the computational load on vehicles themselves and enhancing their computational efficiency [4]. For VEC task offloading, there are still some critical issues that are often overlooked in current research.

It is worth noting that due to the high-speed mobility of vehicles in the Vehicle to Infrastructure (V2I) scenario, V2I offloading must deal with the network topology varying in time. During the offloading process, vehicles can move out of the coverage range of the current Roadsides Unit (RSU), leading to the failure of task offloading [5]. In addition, one of the most important objectives for offloading tasks is to reduce delay and energy consumption. However, most of the existing work on task offloading assumes that the size of the computational results is small and the feedback delay can be neglected. For example, individual gaming experiences, such as those involving VR or other gaming devices, can be processed locally by vehicles. Meanwhile, RSUs are capable of storing and managing the broader environment of Metaverse games, including elements like maps and roads. They also support collaborative multiplayer gaming tasks and act as platforms for virtual service transactions. Given the limited computational capabilities of vehicles, these tasks necessitate offloading to RSUs for processing. In the case of VR video, each frame contains comprehensive view information, yet a user only perceives a small segment of the image within their field of view when viewing the video. This results in a significant amount of redundancy in each frame of the VR content. Such extensive redundant data requires processing by the RSU.

In reality, the feedback delay poses significant challenges to the offloading decision-making process [6]. Due to the lengthy transmission time of large tasks and the high mobility of vehicles, vehicles may undergo RSU handovers, and the computed results need to be routed back through

- *Liang Zhao, Tianyu Li, Guiying Meng and Ammar Hawbani are with the School of Computer Science, Shenyang Aerospace University, Shenyang, China. Liang Zhao is also with the Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, Shenzhen 518110, China. (e-mail: lzhao@sau.edu.cn, 13998798119@163.com,mengguiying@163.com, anmande@ustc.edu.cn)*
- *Geyong Min is with the Department of Computer Science, University of Exeter, UK. (e-mail: g.min@exeter.ac.uk).*
- *Ahmed Y.Al-Dubai is with the School of Computing, Edinburgh Napier University, UK. (e-mail: a.al-dubai@napier.ac.uk).*
- *Albert Y. Zomaya is with the School of Computer Science, University of Sydney, Australia (e-mail: albert.zomaya@sydney.edu.au).*
- *Tianyu Li and Ammar Hawbani are the corresponding authors.*

the backhaul network, incurring additional delay. Feedback delay also significantly affects the offloading decision and resource allocation schemes [7]. Therefore, it is essential to reconsider the scale of the offloading decision and the allocation of computational resources among all vehicles.

In the context of the joint task offloadinG And resoUrce allocatioN wiTh resuLt fEedback cosT (GAUNTLET) in large-scale VEC scenarios, Reinforcement Learning (RL) is commonly employed for its ability to quickly solve large-scale problems with multidimensional state spaces [8]. However, RL also faces certain challenges. For high-dimensional environment, large-scale VEC scenarios often involve a high number of vehicles, edge nodes, cloud servers, and potential offloading tasks [9]. The resulting state and action spaces can be extremely high-dimensional, making RL training and convergence challenging. In VEC scenarios, this can involve vehicle locations, task requirements, network conditions, and more. Finding the right state representation can be non-trivia. When the number of tasks, RSUs, or load capacities changes, the models need to be retrained, incurring additional cost.

Taking into account the complexity of environmental changes in the dynamic VEC network, we put forward a task offloading scheme, named as a lAgrange multiplierSbaSed adaptIve computinG offloadiNg with prediction model (ASSIGN). This scheme leverages Generative Adversarial Networks (GAN) to predict future vehicle positions based on historical vehicle trajectories. Thus, addressing the issue of unstable links caused by high vehicle mobility and concurrently reducing the decision space for offloading enhances algorithm efficiency. By constructing an adaptive model and employing a heuristic algorithm based on Lagrange multipliers, we address the variability issues in real-world scenarios. Our main goal is to reduce the average cost in the entire VEC network, primarily targeting cost factors related to task offloading, task transmission, and feedback. In the following, we outline our main contributions.

- Develop an adaptive VEC-integrated distributed task offloading scheme that copes with the time-varying network conditions and diverse user requirements in the VEC network. This architecture establishes an adaptive optimization model to handle the dynamic multi-dimensional features of the system and further dynamically adjusts the system optimization objectives.

- To mitigate the influence of vehicle mobility on the offloading efficiency of the overall system, we employ GAN to predict the real-time future locations of vehicles. This GAN-based prediction serves as a pre-processing component, effectively reducing the decision space for V2I offloading decisions while further enhancing the efficiency and stability of the system.

- Formulate the problem of GAUNTLET as a Mixed-Integer NonLinear Program (MINLP) problem and solve optimization problem using the Lagrange Multiplier-based Non-uniform Genetic Algorithm (LM-NUGA). This approach has high computational efficiency to cope with dynamic scenarios. Through extensive simulation experiments, we demonstrate that the proposed scheme achieves efficient computation services at a lower cost for solving the GAUNTLET problem in highly dynamic scenarios.

2 RELATED WORK

This section provides a literature review on task offloading in VEC through the implementation of diverse algorithms. Additionally, we discuss several studies that consider result feedback for task offloading. Finally, we highlight the innovative aspects of our research in comparison to other works in the field.

2.1 Task Offloading in VEC

The offloading approaches include infrastructure-assisted [10], vehicle-assisted [11], and a combination of both methods [12]. In [13], the authors employ convex optimization techniques to decompose the joint optimization problem into two decoupled sub-problems, addressing the issue of synchronous offloading among multiple vehicles. In [14], the authors utilize RL methods to address the problem of vehicle offloading service assistance. Additionally, they establish a migration decision model through Software-Defined Networking to facilitate the offloading process. In [15], they employ Deep Reinforcement Learning (DRL) to solve the task offloading and resource allocation strategies in highly dynamic environments, achieving the minimum total system cost. In [16], the authors use a two-stage Stackelberg game model to address resource pricing and task offloading issues, and they theoretically prove the existence of a unique Nash equilibrium. For specific details, please refer to supplementary file.

2.2 Task Offloading with Result Feedback

Recently, several studies begin to address the issue of result feedback in task offloading [17]. In [18], the authors propose an online learning framework based on Nash equilibrium to address the peer-to-peer offloading problem with delayed feedback in fog networks. In [19], in order to reduce the energy consumption, the authors investigated a joint computation and communication mechanism for computational result downloading in an Orthogonal Frequency-Division Multiple Access (OFDMA) system.

In VEC system, in [20], the authors employ a self-learning algorithm to address the issue of timely service provisioning for tasks subject to deadline constraints in vehicular cloud computing systems. In [21], the authors developed a stable and efficient offloading method for edge computing that does not terminate services despite the high mobility of vehicles. In [22], the authors proposed a self-learning offloading decision algorithm based on multi-armed bandit theory to address the issue of link instability caused by the high-speed mobility of adjacent vehicles during task offloading. The aforementioned studies primarily focus on efficient task offloading algorithms without considering the allocation of computing resources. Additionally, they do not address the joint optimization of delay and energy consumption, nor do they consider the varying emphasis on cost consumption in different scenarios.

2.3 Summary

From the analysis of the literature above, many current studies do not offer almost real time services in dynamic VEC network that take into account the result feedback. The main factors contributing to the issue are the significant motion of vehicles, which results in unreliable transmission of links, a wide range of options for offloading decisions that prevent algorithms from reaching a stable state, and variable priorities placed on minimising computation cost in different situations.

As for the challenges, we initially employs GAN to construct a global predictive model, using historical vehicle data to predict future positions and speeds of vehicles. This approach reduces the offloading decision space compared to traditional vehicle mobility models. By integrating the global search capability of the NUGA and the precise solution-finding ability of the Lagrange multiplier method under specific conditions, the decision-making algorithm LM-NUGA is proposed. Additionally, the mentioned adaptive model is integrated into the ASSIGN scheme, which can provide on-the-spot services, simultaneously addressing both task offloading decisions and resource allocation issues.

3 SYSTEM MODEL AND PROBLEM FORMULATION

3.1 System Overview

In this section, the ASSIGN offloading scheme is introduced to address the large-scale deployment of VEC-generated vehicles. As depicted in Fig. 1, vehicles on the road generate various large-scale tasks, such as high-definition video transmission, large online games, and metaverse, among others. Offloading the extensive tasks requires a significant amount of delay. Consequently, multiple RSUs are deployed along the roads to facilitate the offloading of these large-scale tasks. In an effort to minimize the computational resources required for vehicular offloading decision-making, in our ASSIGN architecture, the cloud serves as the central execution platform for the entire predictive model algorithm. This centralized approach enables the processing and analysis of comprehensive vehicle data from various sources, thereby facilitating accurate predictions of vehicle positions and velocities for future time slots. Subsequently, the predictive model is distributed to RSUs to assist in the generation of offloading decisions. Each RSU possesses both computational and communication capabilities. The primary function of the RSU entails the optimization of task offloading decision and computational resource allocations for vehicular entities within its designated coverage area. This optimization is conducted with consideration for both the temporal halting parameters of the vehicles and the prevailing conditions of the underlying network infrastructure [23]. The most common notations are detailed in Table 1.

We investigate multi-task offloading across multiple RSUs without sacrificing generality [24], where $M = \{1, 2, \ldots, m\}$ is a set of tasks generated by vehicles, each of which generates a computational task that can be processed either locally or offloaded to an RSU. If the vehicle stays inside the coverage area of the current RSU after offloading, it can directly get the calculation results from the RSU where the work is offloaded. Nevertheless, in the event that the vehicle departs from the coverage area of the current RSU, the computation results must be communicated from the originating RSU to the RSU located at the vehicle's present position. To illustrate this, Fig. 1 depicts two scenarios of task offloading. For instance, vehicles v_1 and v_4 offload their tasks to the RSU_4 and RSU_8 which situated at their current location and subsequently transfer the offloaded result to the RSU_7 and RSU_1. Vehicles v_2 and v_3 select other available RSUs (i.e., RSU_3 and RSU_4) in their proximity to offload the task and transfer the result to RSU_9 and RSU_2 after offloading.

Assuming that each task is indivisible and can only offload to the local vehicle or RSU. Each vehicle has a computation task defined as a tuple $W_m = (D_m, C_m, T_m)$ which can be processed either by the local vehicle or an RSU, where D_m represents the size of task, C_m represents the required CPU cycle, and T_m sets the maximum tolerate delay. Regarding the decision variables, let $a_{mn} \in \{0, 1\}$ to denote whether the computation task W_m offloads to the RSU. Specifically, $a_{mn} = 1$ indicates that the task W_m is offloaded to the RSU n, while $a_{mn} = 0$ implies it is not. Simultaneously, let $b_m \in \{0, 1\}$ indicate whether task W_m is processed locally, where $b_m = 1$ means that task W_m is processed locally, whereas $b_m = 0$ indicates otherwise. The relationship between the decision variables a_{mn} and b_m is given by (1), ensuring a one-to-one assignment between tasks and offloading destinations.

$$\sum_{n=1}^{N} a_{mn} + b_m = 1, \quad 1 \leq m \leq M. \tag{1}$$

3.2 Transmission Model

3.2.1 Task Offloading

In V2I, a critical concern is the determination of the transmission rate at which tasks are offloaded from vehicles to RSUs. The adoption of OFDMA is shown to be an effective strategy to address this concern [25].

$$R_m^u = \log_2 \left(1 + \frac{P_{\mathrm{u}} \left| h_m \right|^2 l_m^{-\beta}}{N_0 + I_m^{\mathrm{u}}} \right) \tag{2}$$

In (2), P_u signifies the transmission power for the uplink, h_m denotes the coefficient of channel fading, l_m is the distance between RSU to vehicle m, β represents the exponent that characterises the path loss, N_0 and I_m^u denote noise power and intercell interference, respectively. Intercell interference can be mitigated through advanced techniques, including interference alignment and continuous inference elimination [26]. In particular, when $I_m^u = 0$, the interference is entirely nullified. Similarly, the downlink transmission rate can be expressed as :

$$R_m^d = \log_2 \left(1 + \frac{P_{\mathrm{d}} \left| h_m \right|^2 l_m^{-\beta}}{N_0 + I_m^{\mathrm{u}}} \right) \tag{3}$$

For the purposes of this paper, we posit complete elimination of intercell interference. Consequently, the wireless transmission delay for vehicle m is formally expressed as:

$$t_{mn}^u = \frac{D_m}{R_m^u} \tag{4}$$

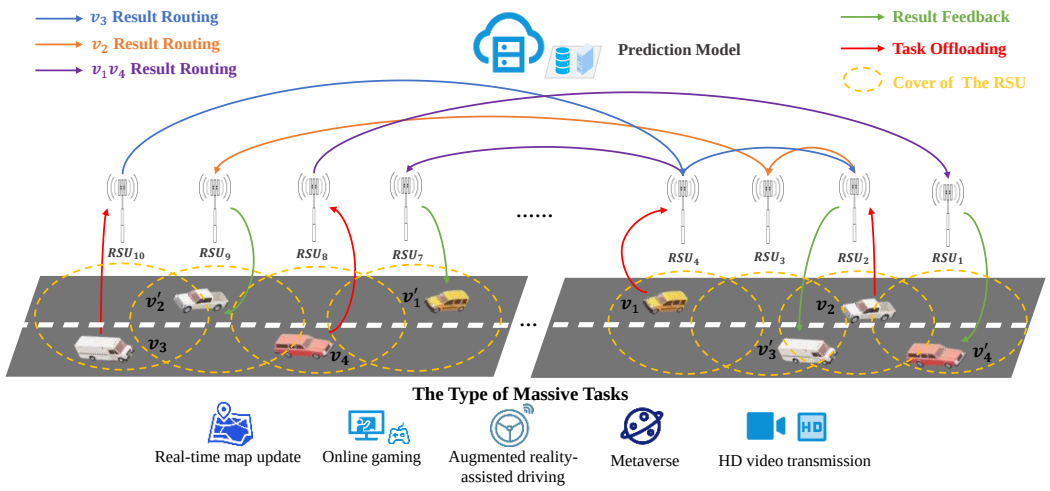


Fig. 1. System illustration. $v_1 - v_4$ and $v_1' - v_4'$ indicate the current position of the vehicle before and after the offloading task.

TABLE 1
TABLE OF NOTATIONS.

Notation/Unit		Description	Notation/Unit		Description
M	——	The number of tasks	t^u_{mn}	s	The upload delay of task m
N	——	The number of RSUs	t^r_{mn}	s	The result routing delay of task m
D_m	MB	The input size of each task	t^{loc}_m	s	The local computation delay of task m
C_m	Gcycle/s	The required CPU cycles	t^R_{mn}	s	The RSU computing delay of task m
T_m	s	The maximum torlerable delay	r^b_m	Mbps	The backhaul tranmission rate
a_{mn}	——	The probability of task offload to RSU	t^q_{mn}	s	The queueing delay for returning the result
b_m	——	The probability of task offload by local	ν	——	The coefficient of energy in loacl offloading
R^u_m	Mbps	The uplink data rate	f^{max}_n	Gcycle/s	The max computing capacity of RSU n
R^d_m	Gcycle/s	The downlink data rate	S	m^2	The road surface area covered by RSU
S^*	m^2	The mean floor space of vehicle	P_u	W	Uplink transmission power
β	——	The path loss exponent	K	——	The number of vehicles and RSUs
N_0	dBm/Hz	Noise power	F^{avg}_{log}	Gcycle/s	The average computing capacity of vehicles
I^u	MHz	The inter-cell interference	F^{avg}_{off}	Gcycle/s	The average computing capacity of RSUs

Here P_u indicates the rated power, and the transmission energy consumption E^u_{mn} is calculated as follows:

$$E^u_{mn} = P_u t^u_{mn} \tag{5}$$

3.2.2 Result Feedback

After finishing the task offloading at the RSU, in a situation where the vehicle has moved outside the boundaries of the present RSU, the result feedback delay, as articulated in (6), is the sum of two components. The first delay is the routeing delay between the offloading RSU and the RSU currently hosting the vehicle. The second delay is the downlink transmission delay from the current RSU to the vehicle [27].

$$t^r_{mn} = \frac{D_m}{r^b_m} + t^d_{mn} \tag{6}$$

First is the delay from the offloading RSU to the RSU presently hosting the vehicle, $r^{b_0}_m$ is the initial backhaul transmission rate. If multiple tasks are transmitted simultaneously between RSUs, it may lead to network congestion, thereby affecting communication efficiency.To simulate the impact of piled tasks on parallel task offloading schemes, we use r^b_m to represent the actual result feedback transmission rate in practice.

$$r^b_m = r^{b_0}_m \times \ell \left(1 + \frac{P_{\mathrm{d}} |h_m|^2 l_m^{-\beta}}{N_0 + I^{\mathrm{u}}_m}\right) \tag{7}$$

where ℓ is the adjustment coefficient which means the impact on network congestion caused by parallel transmission. Keep in mind that after offloading, if vehicle stay within the RSU, the feedback delay is not exist.

The resulting downlink channel is unknown when the decision is made about task offloading. The downlink transmission rate R^d_m is used as a proxy for the downlink channel in order to keep the system model feasible. The delay required for vehicle m to receive the results of its download from RSU via wireless transmission can be expressed as:

$$t^d_{mn} = \frac{D_m}{R^d_m} \tag{8}$$

3.3 Computation Model

3.3.1 Local Computation

We define f^{loc}_m as the local computation resources of vehicle m, with units in cycles per second, C_m is the total number of CPU cycles required to compute one task, measured in cycles. Therefore, the energy consumption E and the square of the frequency f^{loc}_m are indeed proportional. The delay and

energy consumption of vehicle m when it offloads a task to itself can be expressed as by (9), and (10), respectively.

$$t_m^{loc} = \frac{C_m}{f_m^{loc}} \tag{9}$$

$$E_m^{loc} = \nu {f_m^{loc}}^2 C_m \tag{10}$$

where ν represents the coefficient of energy utilization in the local offloading, which means the proportion of computational resource f_m^{loc} require task m to be computed locally. In binary offloading usually set the coefficient to 1.

3.3.2 Computation at RSU

Here we address the allocation of computational resources at RSUs for processing tasks generated by vehicles. Let $f_r > 0$ denote the computational cost that RSU allocates to the vehicle m in its coverage. Accordingly, we can define the delay and energy consumption by (11) and (12), respectively.

$$t_{mn}^R = \frac{C_m}{f_m^r} \tag{11}$$

$$E_{mn}^r = {f_m^r}^2 C_m \tag{12}$$

We define the decision set $S = \{a_{m0}, a_{m1}, \ldots, a_{mn}\}$. Drawing from the equations (4), (6), (8), (9) and (11) above, we introduce t_{mn}^{off} in (13).

$$t_{mn}^{off} == \begin{cases} t_m^{loc} & \text{if } a_{mn} = 0 \\ t_{mn}^u + t_{mn}^r + t_{mn}^R & \text{if } a_{mn} = 1 \end{cases} \tag{13}$$

Next, we compute the maximum offloading delay T_{mn}^{off} as in (14). It reflects the collection of t_{mn}^{off}, indicates the maximum offloading delay.

$$T_{mn}^{off} = \max\left\{t_{m1}^{off}, t_{m2}^{off}, \ldots, t_{mn}^{off}\right\} \tag{14}$$

Similarly, we derive the formula for energy consumption:

$$E_{mn}^{off} == \begin{cases} E_m^{loc} & \text{if } a_{mn} = 0 \\ P^u t_{mn}^u + P^d t_{mn}^d + P^R t_{mn}^R + E_{mn}^r & \text{if } a_{mn} = 1 \end{cases} \tag{15}$$

Finally, based on (14) and (15), we can express the total cost associated with the edge execution scheme for a given RSU n and task m as below, where β^T and β^E are the tuning parameters.

$$C_{mn} = \beta^T T_{mn}^{off} + \beta^E E_{mn}^{off} \tag{16}$$

3.4 Adaptive Model

In the pursuit of optimizing the system while considering Quality of Service (QoS) and energy-related factors, we devise an adaptive relational model to assign appropriate weights β^T (i.e., QoS significance) and β^E (i.e., energy efficiency significance). The relationship between these two weights is encapsulated in the following equation:

$$\beta^T + \beta^E = 1 \tag{17}$$

Our proposed model dynamically balances these two tuning parameters. We base our weight adjustments on a set of key criteria.

- **Consideration of overall QoS**: Our fundamental approach is to holistically assess QoS, with varying aspects of QoS represented by $\varpi \in [0, 1]$. We propose a metric, traffic density ϖ in (18), that takes into account the road coverage area, the number of RSUs, and the number of vehicles. S_i and S^* represent the road surface area covered by RSU and the mean floor space of vehicle. N is the number of total RSUs and n is the number of vehicles in the range of RSU.

$$\varpi = \frac{\sum_{i=1}^{N} 1 - \frac{S_i - nS^*}{S_i}}{N} \tag{18}$$

- **Evaluation of RSU load**: We evaluate the load degree of RSUs. According to the relationship between RSUs and the calculation capacity of vehicle numbers, the load degree primarily determines the magnitude of the influential factor $\sigma \in [0, 1]$. In (19), the parameter k governs the steepness of the sigmoid curve. L denotes the load level of RSUs, quantifiable through metrics like CPU utilization or the length of the task queue. $L_{\text{threshold}}$ serves as a critical threshold, identifying when RSUs are deemed to be under high load, adjustable according to system needs. This equation portrays σ as a sigmoid function, with σ increasing as L exceeds the threshold. indicating a higher load degree and a greater influence of load on the QoS weight. It can adjust the parameters and the threshold value to fit any scenario.

$$\sigma = \frac{1}{1 + \exp(-k(L - L_{\text{threshold}}))} \tag{19}$$

- **Comparison of computing capacity**: Vehicles typically have less computing capacity compared to RSUs. By evaluating the average computing capabilities of both vehicles and RSUs, the influence weight is established as $\varrho \in [0, 1]$. The factors σ and ϱ are contrasting influences that fulfill the condition $\sigma + \varrho = 1$. As the QoS requirements increase, so does the demand for higher computing capabilities in RSUs.

$$\beta^E = \varpi \left\{ \sigma \left(1 - \frac{\sum_{n=1}^{N} f_n^{max}}{\sum_{n=1}^{K} f_n^{max}} \right) + \varrho \frac{F_{loc}^{avg}}{F_{off}^{avg}} \right\} \tag{20}$$

In (20), K is the total number of RSUs and vehicles. f_n^{max} is the maxium computing capacity of the RSU n. F_{loc}^{avg} denotes the mean computing capacity of the vehicles when locally computing. F_{off}^{avg} denotes the mean computing capacity of the vehicles when offloading tasks to RSUs. There are numerous elements that influence the link between the number of RSUs and vehicles. For instance, when the RSUs' number increases, the percentage of weight β^E decreases, and the corresponding percentage of weight β^T is higher, so delay has a large impact on this situation.

3.5 Problem Formulation

In the whole VEC system, minimizing the total cost by jointly the task offloading decision set S and the computation resource allocation f is our main objective, and in order to deal with the GAUNLTLET problem, the objective function C is given by (21):

$$C(a,b,f_m^r,f_m^{loc}) = \sum_{m=1}^{M}\sum_{n=1}^{N} a_{mn}C_{mn}^{r\_total} + b_m C_m^{loc\_total} \tag{21}$$

where $C_{mn}^{r\_total}$ and $C_m^{loc\_total}$ denote the total cost at RSU and local for task offloading. Thus, the problem with optimization can be formulated as follows:

$$\min_{a,b,f_m^r,f_m^{loc}} \quad C(a,b,f_m^r,f_m^{loc}) \tag{22}$$

$$\text{s.t.} \quad \sum_{m=1}^{M} a_{mn} f_m^r \le F_n^r, \quad 1 \le n \le N \tag{22a}$$

$$\sum_{m=1}^{M} b_m f_m^{loc} \le F^{loc}, \tag{22b}$$

$$a_{mn}T_{mn}^{r\_total} \le T_m, \quad 1 \le m \le M, 1 \le n \le N \tag{22c}$$

$$b_m T_m^{loc\_total} \le T_m, \quad 1 \le m \le M \tag{22d}$$

$$\sum_{n=1}^{N} a_{mn} + b_m = 1, \quad 1 \le m \le M \tag{22e}$$

$$f_m^r \ge 0, \quad 1 \le m \le M, 1 \le n \le N \tag{22f}$$

$$f_m^{loc} \ge 0, \quad 1 \le m \le M \tag{22g}$$

$$a_{mn}, b_m \in \{0,1\} \tag{22h}$$

This formulation is suitable for addressing the complexities of joint task offloading, resource allocation, and result feedback cost in large-scale VEC scenarios. Constraints (22a) and (22b) ensure that the sum of computing resources allocated to tasks by the cloud server does not exceed its computing capacity, where F^{loc} and F_n^r are the total computing resource of the vehicle m and the RSU n. Constraints (22c) and (22d) indicate that the total delay to complete a computing task cannot exceed the maximum delay tolerance of tasks in order to maintain balance and stability of the system, and the constraint (22e) ensures that a computational task is only offloaded to a single RSU to complete the computation to ensure the completeness of each task.

Since the optimization problem (22) contains nonlinear functions and integer variables, the optimization problem is MINLP [28]. There are no polynomial-time algorithms available to solve the MINLP exponential explosion problem. To overcome these obstacles, we use LM-NUGA, which concludes the heuristic algorithm and the Lagrange multiplier method in the next section to solve the offloading scheme and the computing resource allocation, respectively.

4 PROPOSED SOLUTION

The preceding optimization problem (22) is a non-convex optimization problem, which is challenging to solve with conventional optimization techniques. In this paper, the optimization problem is solved by using the combination of an improved NUGA and the Lagrange multipliers method. In simple terms, we encode only the offloading scheme. Using the Lagrange multiplier method can determine the near-optimal resource allocation scheme under the offloading scheme when a specific scheme is provided. In light of the preceding, we employ LM-NUGA to conduct a global search for better solutions.

4.1 Pre-prossessing of ASSIGN

We introduce the GAN-based pre-processing method, and build the pre-processing architecture through prediction model. The obtained prediction model can judge the number of target RSUs for task offloading, minimize the decision space and improve the overall effectiveness in the whole VEC system.

4.1.1 *Preliminaries*

GAN is a conventional model that differs from GSNs and Boltzmann machines in that it relies solely on backpropagation without employing complex Markov chains [29].

The objective of GAN is to establish a predictive model for future traffic conditions based on the current traffic situation [30]. In this context, the driving conditions of vehicles are denoted as $V = \{V_c \cup V_r\}$, V_c is the vehicle state and V_r is the road state. The data set for V_c is denoted as $V_c = \{v_{id}, x, y, v_s\}$, where v_{id} represents the quantity of vehicles. (x, y) reflects the vehicle's location, v_s represents the current velocity. Moreover, the road condition data set $V_r = \{P_s, P_e, p_f, p_d, p_v, N\}$. P_s and P_e are the starting and ending places of the road, respectively. p_f is a binary variable that indicates if another vehicle is moving ahead of the vehicle, p_d is the distance from the vehicle to the vehicle in front, and p_v is the velocity of the vehicle ahead. N is the total number of vehicles.

$$[V_{t+1}, V_{t+2}, \ldots, V_{t+T}] = F(V_t) \tag{23}$$

The relationship between current and predicted traffic condition is defined in (23). The mapping function for prediction model is defined as $V_t = F(V_t)$, where V_t is vehicle condition data and F is the mapping function at time t. V_{t+1} is the driving condition date of the vehicle predicted by the generated network 1 s later. Besides, to assess the accuracy of the prediction model, we designate the loss function l as below:

$$l = \hat{F}(V_t) - F(V_t) \tag{24}$$

where $\hat{F}(V_t)$ denotes the actual data on traffic conditions. Based on the above definition, the prediction model can solve the mapping function F, which is equivalent to minimize the loss function l.

1) *Generative Model*: The function of the generator is to generate the predicted vehicle trajectory and other parameters in order to deceive the discriminator. The mapping function G of the generative model is based on the vehicle's driving state. Here, $\hat{V}_{t_g}$ represents the set of vehicle states for the generative model, denoted as $\hat{V}_{t_g} = G_\varsigma(V_{t-1})$.

2) *Distribution Model*: The responsibility of the discriminator in GAN is to differentiate. If the calculated trajectories closely resemble real trajectories, then discriminator's output is near to 1, indicating difficulty in distinguishing them. The discriminator cannot evaluate the output of the generator based solely on criteria V_{t-1} or $\hat{V}_{t_g}$. The output data of the generator regarding vehicle trajectories should

Algorithm 1: Pre-processing process for ASSIGN

Input : generator G_ς, discriminator D_τ, learning rate ω_g and ω_d, date set μ,

1 Initialize G_ς, D_τ;
2 $o \leftarrow \varphi$;
3 **repeat**
4 **for** *Generator* **do**
5 $\hat{Z}_t = \left\{V_{t-1} \cup \hat{V}_{t_g}\right\}$;
6 $\nabla = \log\left(1 - D_\tau\left(\hat{Z}_t\right)\right)$;
7 $\varsigma \leftarrow \varsigma - \omega_g \nabla$;
8 **end**
9 **for** *Discriminator* **do**
10 $\nabla = \log\left(D_\tau\left(Z_t\right)\right) + \log\left(1 - D_\tau\left(\hat{Z}_t\right)\right)$;
11 $\tau \leftarrow \tau + \omega_d \nabla$;
12 **end**
13 $o \leftarrow \varsigma$;
14 Return o;
15 **until** *GAN converge*;

Output: mapping function G_o

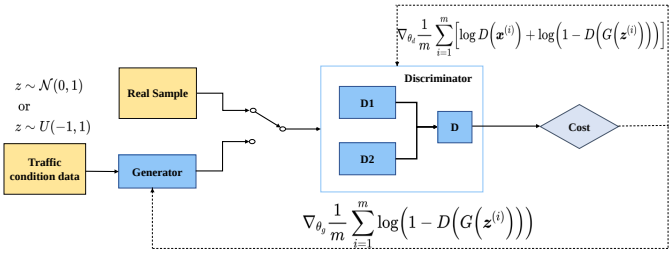


Fig. 2. Diagram of the Framework for GAN.

not be used as the sole input for the discriminator to check accuracy.

Fig. 2 shows the system framework diagram of GAN, which includes a generator and a discriminator. The complete flow of GAN is presented in Algorithm 1. In essence, the pre-processing process can be regarded as the vehicle trajectory prediction process, which involves solving mapping function F, with the objective of minimizing the loss function l. G_ς and D_τ undergo training subsequent to the generator and discriminator being initialized. The output mapping function G_0 is assigned of G_ς after training. After the values of the generator G_ς and discriminator D_τ has been initialized, lines 4–8 and 9–12 show the training process of them, respectively.

4.1.2 Prediction Model of GAN

This section describes the GAN-based prediction model for vehicle trajectory prediction. On the one hand, the prediction model estimates the maximum distance that vehicles will travel in the future, which helps to determine the number of RSUs selected for offloading decision and narrows the decision space. It enhances the precision of the ultimate offloading decision solution. On the other hand, the model also provides the future average vehicle velocity as a discriminatory metric for mobility detection. It serves as an evaluation criterion for real-world scenarios by incorporating real-time vehicle positions into the penalty function of the offloading decision algorithm. The following are the primary evaluation metrics for the prediction model:

$$t_{max}^{trans} = \frac{D_{max}^{n}}{R_m^{u_{min}}} \tag{25}$$

$$t_{max}^{comp} = \frac{C_{max}^{n}}{f_{min}^{r}} \tag{26}$$

$$t_{max}^{total} = t_{max}^{trans} + t_{max}^{comp} \tag{27}$$

where (25) and (26) denote the maximum transmission delay the computational offloading delay of the task. D_{max}^{n} denotes the maximum data size of all tasks whose unit is usually upto GB, $R_m^{u_{min}}$ is the minimum speed for transmission, which depends on the largest distance between RSUs. The specific explanation of them is mentioned in subsection 3.2. C_{max}^{n} is the maximum number of CPU cycles, which means the amount of computation required for task offloading, and f_{min}^{r} is the minimum computing capacity of the RSUs. t_{max}^{total} is the maximum time that typically represents the total time it takes to complete all tasks contained in an application. Thus, we can obtain the maximum time and predict the maximum distance traveled by the vehicle within the maximum time, and select the RSUs within the range of the distance as the offloading object to achieve the purpose of reducing the decision space.

4.2 Offloading Scheme with LM-NUGA

In this section, we employ an improved Non-Uniform Genetic Algorithm called LM-NUGA to deal with the problem of offloading decision-making and resource allocation. We focus solely on encoding the offloading scheme and utilized the Lagrange multiplier method to determine the near-optimal resource allocation scheme under a given specific offloading scheme. This approach makes the entire system better suited for highly dynamic VEC scenarios.

4.2.1 Preliminaries

Genetic Algorithm (GA) is a mathematical model of how evolution happens in living things [31]. For specific details, please refer to supplementary file.

First, we obtain the preliminary offloading decision by using NUGA. By applying the Lagrange multiplier method, we can determine the near-optimal resource allocation under these decisions. Combining the advantage of NUGA in global search that avoids getting trapped in local optima, we can further optimize the process of solving offloading decision within NUGA. This iterative process allows us to obtain a more accurate offload decision.

4.2.2 Population Initialization

Population initialization is the process of creating an initial group of random individuals at the onset of GA. It serves as a starting point within the search space of the algorithm. During initialization, various parameters are typically set for each individual, including chromosome code, gene number, range of values, and fitness function. The choice of population initialization method directly impacts the search

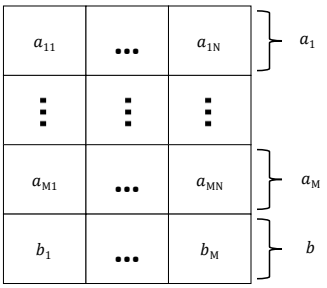


Fig. 3. The Encoding of an individual.

space, initial conditions, and search paths, thereby influencing the overall global and local search performance of GA.

1) *Chromosome Encoding*: the specific encoding for individuals is illustrated in Fig. 3. Since the offloading scheme is a two-dimensional binary number, it can serve as a coding scheme. It is important to note that only the offloading scheme is involved in the crossover and mutation operations of GA. Once the offloading scheme has been identified, the optimization problem (22) and the remaining resource allocation problem become convex optimization problems. These problems may be effectively handled by using the Lagrange multiplier method. If constants such as a_{mn} and b_m are involved and the other constraints are linear, the optimization problem (22) turns into a convex optimization problem. This approach significantly enhances the computational efficiency of GA.

2) *Space Searching*: according to constraints (22e) and (22h), we set $a_{mn} = \{a_{m1}, a_{m2}, \ldots, a_{mN}\}$ and $b = \{b_1, b_2, \ldots, b_M\}$ as the smallest unit genetic factors, which ensures that the constraints are met during the crossover and mutation processes. For convenience, $K^i = \{\mathbf{a_1^i}, \ldots, \mathbf{a_N^i}, \mathbf{b^i}\}^\top$ is used to represent an individual. For $T_{mn}^{loc\_total} \geq 0$ and $T_{mn}^{r\_total} \geq 0$ it is easily obtained from constraint (22c) and (22d).

$$a_{mn}(t_{mn}^u + t_{mn}^r + t_{mn}^d + t_{mn}^R) \leq T_m, \quad 1 \leq m \leq M, 1 \leq n \leq N \tag{28}$$

$$b_m(t_m^{loc}) \leq T_m, \quad 1 \leq m \leq M \tag{29}$$

Here we will demonstrate that the search space $\mathcal{C}$, which satisfies constraints (28) and (29), can serve as the feasible region for the mutation operation. Since the delay in (28) and (29) are only affected by environmental variables, we can preliminarily judge whether the offload scheme a_{mn} or b_m can satisfy constraints (22c) and (22d) when the computational resource allocation scheme is unknown, because if constraints (28) and (29) are not established, then constraints (22c) and (22d) must not hold. It also improves the efficacy of the algorithm, i.e., $\forall K^i \in \mathcal{C}$.

3) *Fitness Function*: the fitness function is a critical component of NUGA used to assess the fitness or character of each individual during the evolutionary process. The selection and reproduction of superior individuals depend on their fitness function. The fitness function plays a vital role in the performance and convergence speed of NUGA. The objective is to minimize total cost consumption in this work. Once an offloading scheme is determined, the problem can be formulated as below:

$$\min_{f^r, f^{loc}} \quad \sum_{m=1}^{M}\sum_{n=1}^{N}(a_{mn}C_{mn}^{R\_total} + b_m C_m^{L\_total}) \tag{30}$$

$$\text{s.t.} \quad \sum_{m=1}^{M} a_{mn} f_{mn}^r = F_{mn}^r, \quad 1 \leq m \leq M, 1 \leq n \leq N \tag{30a}$$

$$\sum_{m=1}^{M} b_m f_m^{loc} = F_m^{loc}, \quad 1 \leq m \leq M \tag{30b}$$

$$0 < f_{mn}^r \leq F_{mn}^r, \quad 1 \leq m \leq M, 1 \leq n \leq N \tag{30c}$$

$$0 < f_m^{loc} \leq F_m^{loc}, \quad 1 \leq m \leq M \tag{30d}$$

where the constraints (30b) and (30c) are equality constraints because our objective function is to minimize the total cost, and allocating all computing resources can reduce the total delay.

(a) *sufficiency condition*: Problem (30) is a convex optimization problem.

For simplicity, first we write $g_{mn}(f_{mn})$ as problem (30), then the objection function can be written as $\sum_{m=1}^{M} g_{mn}(f_{mn})$. For $g_{mn}(f_{mn})$, when $a_{mn} \neq 0$, its second derivative is as follow:

$$\frac{\partial^2 g_{mn}(f_{mn})}{\partial f_{mn}^2} = \frac{2\hat{a}_{mn} c_m d_m}{f_{m,n}^3} > 0, \forall f_{mn} > 0 \tag{31}$$

$g_{mn}(f_{mn})$ is a convex function. Specially, when $a_{mn} = 0$, $g_{mn}(f_{mn}) = 0$, which is also a convex function. Therefore, $g_{mn}(f_{mn})$ is a convex function with respect to f_{mn}, and the objective function is a sum of convex functions. In conculsion, the objective function is a convex optimization problem.

The equality constraint $\sum_{m=1}^{M} a_{mn} f_{mn}^r = F_{mn}^r$ defines a hyperplane in terms of f_{mn}, which is a convex set. The constraint (30b) is in the similar way. Constraint (30c) and (30d) are the intersection of an open interval and a closed interval, which is also convex set.

(b) *necessary condition*: The first derivative of the Lagrange function is 0.

we can construct the Lagrange function as follow:

$$\begin{aligned} &L(f_{mn}^r, f_m^{loc}, \lambda, \psi) = \\ &\sum_{m=1}^{M}\sum_{n=1}^{N}(a_{mn}C_{mn}^{R\_total} + b_m C_m^{L\_total}) \\ &- \sum_{n=1}^{N}\sum_{m=1}^{M}\lambda_n(a_{mn}f_{mn}^r - F_{mn}^r) \\ &- \psi\sum_{m=1}^{M}(b_m f_m^{loc} - F_m^{loc}) \end{aligned} \tag{32}$$

ψ are the Lagrange multiplier. Then the partial derivatives are calculated for the parameters of the Lagrange function respectively, and finally the near-optimal solutions of f^{r^*} and f^{loc^*} are obtained.

$$f_{mn}^{r^*} = \frac{F_{mn}^{r}}{\sum\limits_{\substack{i=1 \\ i \neq m}}^{M} \sqrt{a_{in} C_i}} \tag{33}$$

$$f_{m}^{loc^*} = \frac{F_{m}^{loc}}{\sum\limits_{\substack{i=1 \\ i \neq m}}^{M} \sqrt{b_i C_i}} \tag{34}$$

To evaluate the fitness of an individual, the fitness function is defined as follow:

$$Fitness = -C(a, b, f_{mn}^{r^*}, f_{m}^{loc^*}) - Penalty \tag{35}$$

where $Penalty$ is the penalty function defined as

$$\begin{aligned} &Penalty = \\ &\alpha \sum_{m=1}^{M} \max\left(0, b_m C_m^{L\_total} - C_m\right) \\ &+ \alpha \sum_{m=1}^{M} \sum_{n=1}^{N} \max\left(0, a_{mn} C_{mn}^{R\_total} - C_m\right), \end{aligned} \tag{36}$$

α is a penalty factor which is generally a large positive number, usually set to 10^4. When $\alpha = 0$, this implies that there is no need to adhere to the constraints. Its role is to add a penalty term to the fitness function to force the search process during optimization to obey the constraints (22c) and (22d).

4.2.3 The Basic Operation of LM-NUGA

In this section, we will introduce some basic operations of NUGA, including selection, crossover, mutation, and the design of non-uniform probability distribution functions.

1) *Selection*: the tournament selection method is employed to choose individuals with higher fitness values as parents, generating the next generation of individuals [32]. To enhance the performance of NUGA, the concept of elite preservation is employed, guaranteeing the participation of individuals with the highest fitness in the reproduction of the next generation. Furthermore, the individual with the highest fitness is updated at each iteration. The detailed process is presented in Algorithm 2.

2) *Crossover*: For detailed information regarding the crossover, please refer to the supplementary file.

3) *Mutation*: For detailed information regarding the mutation, please refer to the supplementary file.

4) *Probability Distribution Function*: the conventional GA explores the search space in a uniform manner, which can lead to the search getting trapped in local optima. In contrast, NUGA is an evolutionary computation algorithm that enhances the diversity of the search space by introducing non-uniform mutation operations. By modifying the search scheme within the GA, the NUGA introduces a non-uniform variation rate [33]. In our approach, the non-uniform probability distribution function adopts an exponential decay function, where the mutation probability exponentially decreases with the increase in iteration count.

$$p_m(t) = p_m^{(0)} e^{-\lambda \frac{t}{t_{\max}}} \tag{37}$$

Algorithm 2: Near-optimal individual selection

Input : population K, elite K^{i^*}, population size S
for $s = 1$ **to** S **do**
  Randomly choose two individuals K^i and K^j;
  Calculate $f_{mn}^{r^*}$ and $f_m^{loc^*}$ corresponding to K^i and K^j according to (33) and (34);
  Calculate the fitness of two individuals according to the (35);
  Select an individual K^* with better fitness to insert into K';
end
if K^* *not in* K' **then**
  Replace the individual with the lowest fitness with K^{i^*};
end
Output: better population K'

where $p_m(t)$ denotes the mutation probability at iteration t, $p_m^{(0)}$ represents the initial mutation probability, $t_{\max}$ is the maximum number of iterations, and λ is a parameter controlling the rate of exponential decay. In the experiment, the values of λ and $p_m^{(0)}$ need to be designed according to the parameters in the adaptive model to achieve the best algorithm performance and search performance.

4.2.4 Overview of LM-NUGA

The LM-NUGA mainly includes initial population, selection, crossover, and mutation operations, which are shown in Algorithm 3. In the initial population stage, a set of solutions is randomly generated according to the search space $\mathcal{C}$, which is the first generation population K. In the selection stage, the algorithm selects excellent and elite individuals as parents of the next generation population according to the fitness function (35) and updates the optimal individual K^{i^*}. If the optimal individual is not in the population, replace the worst individual in the population with the optimal individual K^{i^*}. In the crossover and mutation stage, the algorithm designs the probability of crossover and mutation through the non-uniform probability distribution function. At the same time, the algorithm can always satisfy constraints (22e) and (22h) in the process of crossover and mutation, which reduces the search space and improves the efficiency of the Algorithm 3.

For detailed information regarding the complexity analysis of LM-NUGA, please refer to the supplementary file.

4.3 The Complete Process of ASSIGN

we describe the pre-processing of ASSIGN in 4.1, the offloading decision-making algorithm with LM-NUGA in 4.2 in order to integrate the complete ASSIGN offloading scheme. Algorithm 4 outlines the entire ASSIGN task offloading process. It takes whole VEC network as input, the total cost required for system task offloading as output, along with the computation resource allocation and offloading schemes for each vehicle and RSU. The whole implementation of ASSIGN is outlined as below.

Firstly, ASSIGN initializes its required parameters. Secondly, based on the extracted parameters, it calculates the

Algorithm 3: LM-NUGA

Input : population size S, iteration times T, crossover probability p_c, mutation probability p_m, number of tasks from vehicles m, number of RSU n, tasks W_m

According to the constraints (28) and (29) to get the search space $\mathcal{C}$;
Generate a new population K according to the search space $\mathcal{C}$;
for $t = 1$ **to** T **do**
  According to Algorithm 2, update population K to K';
  Perform the crossover operation on population K' with probability p_c to get population K'';
  To generate population K''' from population K'' with a probability p_m using the mutation operation;
  Calculate the fitness of K''' and update the elite individual K^{i^*};
  $K = K'''$;
  $t = t + 1$;
end
Use K^{i^*} to calculate $f_{mn}^{r^*}$ and $f_{mn}^{loc^*}$ by (33) and (34);

Output: K^{i^*}, $f_{mn}^{r^*}$, $f_{mn}^{loc^*}$

adaptive weights λ^E for different scenarios using (20). Thirdly, ASSIGN constructs a prediction model. It calculates the maximum delay required for vehicle task offloading and determines the current location of vehicles. The information is then used to determine the maximum number of RSUs for task offloading, reducing the decision space and enhancing overall system efficiency. Finally, ASSIGN employs the elite selection scheme from Algorithm 2 to choose suitable individuals for optimizing the performance of LM-NUGA. It utilizes Algorithm 3 to calculate task offloading decision and allocate resources between RSUs and vehicles.

Algorithm 4: ASSIGN

Input: The entire VEC network

Extracts the raw data to initialize the parameters in VEC network;
Calculate the adaptive model parameter β^E by (20) based on the parameters in the scene;
Calculates the maximum delay of vehicle to complete the maximum task offloading by using Algorithm 1 to constructs the prediction model.
Perform Algorithm 2 and 3 to compute the allocation of computing resources and offloading decision for RSUs and vehicles.

Output: The total cost to complete the offloading task, including task decision-making and resource allocation results of RSUs and vehicles.

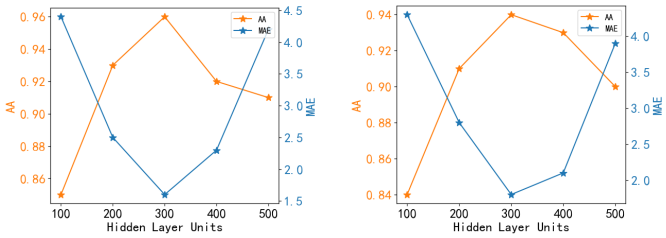
5 EXPERIMENTS

This section outlines the performance evaluation of ASSIGN which can be divided into four subsection. The system scenario and key evaluation parameters are introduced in 5.1. Subsection 5.2 evaluates the performance of the prediction model based on GAN. Subsection 5.3 evaluates the adaptive ability of ASSIGN in various scenarios, and finally 5.4 compares the efficiency of ASSIGN task offloading to that of other mainstream schemes.

5.1 Evaluation Parameters and Scenarios

We evaluate a scenario where a 20-kilometer road is selected and several RSUs are placed in VEC. The positions of each vehicle are produced using the Simulation of Urban Mobility (SUMO). A predictive model is employed to select a subset of RSUs as examples for running the simulation. The programming language employed is Python 3.7, while the experimental simulation platform is a Windows 11 64-bit system, incorporating libraries such as numpy, torch, and matplotlib. We have uploaded the open source code of the platform available on Github [1]. For detailed information regarding the evaluation parameters, please refer to the supplementary file.

5.2 Evaluation of Prediction Models



(a) (b)

Fig. 4. A comparison of the effectiveness of (a) test sets and (b) training sets.

We evaluate the outcome of GAN-based predictive models in this section, and we will introduce the following two performance metrics to evaluate the effectiveness of the predictive model.

- *Mean Absolute Error* (MAE) : The average absolute error is the difference between the predicted and actual values. The method of calculating the average absolute error is to add the absolute errors of all samples and then divide by the number of samples. The MAE represents the mean squared error between the actual and predicted speeds, with units in meters per second.

$$\mathrm{MAE} = \frac{1}{S}\sum_{s=1}^{S}|\hat{y}_s - y_s| \tag{38}$$

- *Average Accuracy* (AA) : Average accuracy is a measure of the average performance of models in different classes, and it is also the average accuracy of multiple classes. It provides an overall performance measure, which here represents the average accuracy of vehicle trajectory predictions.

1. https://github.com/NetworkCommunication/ASSIGN

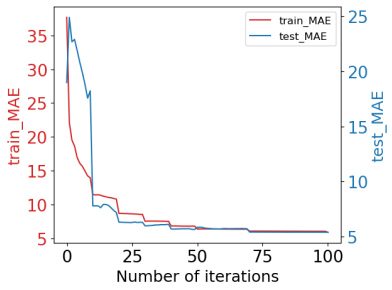


Fig. 5. Performance evaluation of GAN-based predictive model.

$$\mathrm{AA} = \frac{1}{S}\sum_{s=1}^{S}\left(1 - \frac{|\hat{y}_s - y_s|}{S}\right) \tag{39}$$

In our GAN model, both the generator and discriminator are based on a seven-layer neural network architecture. Initial learning rates of 0.02 and 0.01 are set for the generator and discriminator, respectively. In determining the effectiveness of GAN adversarial learning, the number of hidden neurons is crucial. The generator may produce samples with low fidelity if the number is too small. Overfitting may occur if the quantity is excessively high. In our experiments, between 100 and 500 hidden neurons were selected for training. The final number of hidden neurons was determined by selecting the values that yielded the highest AA and the lowest MAE. In Fig. 4(a) and 4(b), the adversarial learning process and performance of model are optimal when the number of hidden neurons reaches 300.

Fig. 5 depicts the training set and test set evaluation metrics for the GAN-based prediction model. MAE measures the accuracy of our predictions of model. When the number of hidden layer neurons is set to 300, the MAE value reaches its minimum after approximately 75 iterations, signifying that the model's training performance satisfies the specified criteria.

5.3 Assessment of Adaptability in Different Scenarios

In this section, we evaluate the effectiveness of adaptive models in three distinct situations. Fig. 6 depicts the impact of the value of delay and energy consumption for various scenarios on the experimental outcomes of total cost.

To address various scenarios in the VEC network, we develop a more realistic scenario model to simulate task offloading. We consider three different scenarios to verify the effectiveness of adaptability of our model: Commercial Trunk Road (CTR) , Village Trunk Road (VTR), and Residential Trunk Road (RTR). In the CTR scenario, there is a high volume of traffic and vehicle density. Task offloading among vehicles requires a high level of QoS to meet the demands. On the other hand, in VTR scenario, the vehicle density is lower, and vehicles are traveling at higher speed. In this case, task offloading prioritizes energy efficiency considerations to reduce energy consumption. The VTR scenario lies between the CTR and VTR scenarios, encompassing both aspects.

We analyse the overall computational cost in three distinct environments, considering various factors such as task's number, data size, and task's complexities. In Fig. 6(a), 6(b) and 6(c), the trends of curves are stability under various conditions. This indicates that our algorithm allows the adaptive model to consider the trade-off between delay and energy consumption in real-world dynamic scenarios.

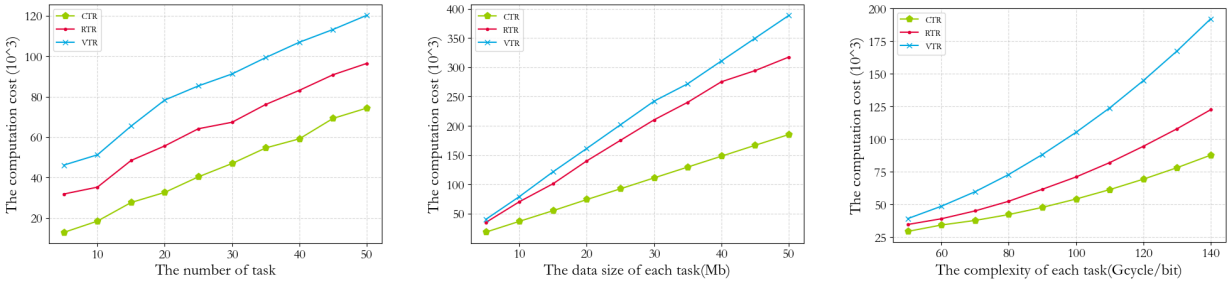
5.4 Comparative Performance Evaluation

We list six baseline schemes to compare with our proposed ASSIGN scheme in this section, including Lagrange multipliers-based NUGA (LM-NUGA), Prediction Model-assisted GA (PM-GA), Prediction Model-assisted PGL (PM-PGL), Prediction Model-assisted PSO (PM-PSO), Prediction Model-assisted Loacl (PM-LOC), and Prediction Model-assisted Greedy (PM-GRE). Our scheme is denoted as ASSIGN, and we set the pre-processing parameters to the same values and controlled variables to conduct comparative experiments. For detailed information regarding the aforementioned baseline schemes, please refer to the supplementary file.

Fig. 7(a) shows the computation cost of PM-LOC, LM-NUGA, PM-PSO, PM-GA, PM-PGL, PM-GRE, and ASSIGN with different data sizes for each task. The experimental results show that when the number of tasks is set between 10 and 50, as the decision space required by the algorithm increases, the computational complexity also increases. In this scenario, the ASSIGN scheme has the lowest cost among the seven schemes. Furthermore, as the number of tasks increases and the dimensionality of the problem increases, the search space grows exponentially, and each dimension may contain critical information. This makes it difficult for PSO to cover sufficient search space within a reasonable timeframe, thereby hindering its ability to find the global optimum after 25 tasks.

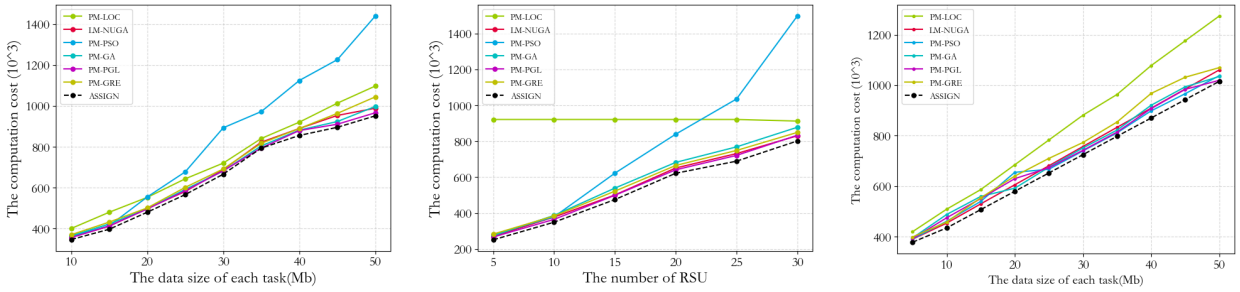
Fig. 7(b) illustrates the effect of different schemes with varying numbers of RSUs, further demonstrating the expandability of ASSIGN. Except for local computation, the overall cost of all schemes increases with an increase in the number of RSUs. Due to limited computational resources, uplink transmission and task offloading computations incur higher cost, and task offloading introduces additional feedback cost. It is evident that ASSIGN shows the fastest convergence speed and the lowest computational cost. The reason is that the use of a pre-processing process based on the predictive model computes the location of the maximum displacement of vehicles during their future movement, thereby determining the number of RSUs for task offloading and reducing the decision space. In addition, GA improved with the Lagrange multiplier method, allowing ASSIGN to converge faster and avoid being trapped in local optima.

Next, we consider the overall computational cost for different scenarios and varying task sizes. In Fig. 7(c), x-axis denotes task size, the total computational and communication resources increase with the growing task scale. It is evident that the ASSIGN scheme exhibits significantly lower computational cost compared to other schemes. Due to the continuous increase in task size, ASSIGN demonstrates superior convergence compared to other schemes, resulting in lower computational cost. In addition, the adoption of the pre-processing method in ASSIGN leads to a more precise decision space, contributing to its better performance compared with other schemes.



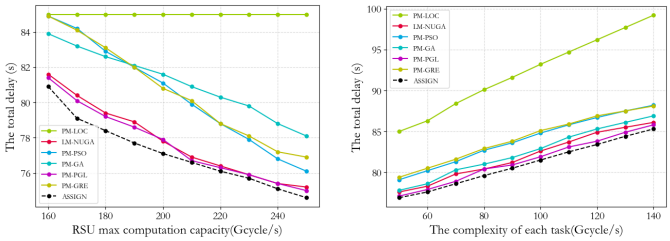
(a) (b) (c)

Fig. 6. Performance for total cost with different task's number (a), task's size (b) and task's complexity (c)



(a) (b) (c)

Fig. 7. Performance for total cost as the number of tasks (a), the number of RSUs (b), and the maximum computation capacity of vehicles (c).



(a) (b)

Fig. 8. The performance of total delay based on different complexity of RSU (a) and complexity of each task (b).

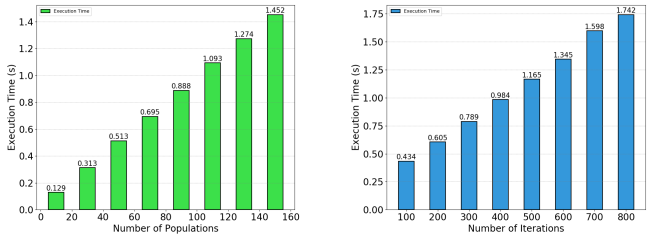
Fig. 8(a) illustrates the impact of the maximum computation resources of RSUs on the overall scenario, comparing the total computation delay for the vehicles in the seven scenarios with different maximum computing capabilities of the RSUs. As observed, the total delay for all scenarios tends to decrease with increasing RSU maximum computation capacity, except the local computation scheme. The increase in computing resources may reduce the offloading delay for tasks, leading to an overall reduction in delay. The ASSIGN scheme achieves the minimum total delay among vehicles as it excels in accurate space-based offloading decision without pre-processing.

In Fig. 8(b), we evaluate the influence of task complexity for the total delay. The total delay for every single scheme escalates as the elevate in task complexity. The reason is that tasks with higher computational complexity need more computation delay compared with tasks with lower computational complexity. Notably, the ASSIGN scheme significantly outperforms with other schemes. This is attributed to the adoption of the Lagrange multiplier method to improve GA, enabling it to achieve fast convergence without getting trapped in local or global optima. Moreover, integration of the prediction model further enhances the performance of ASSIGN compared with other schemes.

In Fig. 9, the execution time of ASSIGN is effected by time complexity of the GAUNTLET problem, which consequently determined by the number of populations and iterations. We list the execution times of the ASSIGN scheme under different numbers of populations and iterations. Initially, both the number of iterations and populations were fixed at 300 and 60, respectively. The actual execution time is clearly less than one second when the problem converges. When the number of iterations reaches 300 and the population reaches 60, the problem has already converged and the minimum actual execution time recorded is about 0.79 seconds.

6 CONCLUSION

We explore the integrated challenges of task offloading and resource allocation within VEC framework in this paper. We proposed an adaptive offloading scheme assisted by a prediction model to address the total computation cost as a combined optimization problem, taking into account the result feedback cost. The proposed scheme intends to offer an efficient offload decision for large-scale V2I offloading in practical vehicular scenarios. The entire scheme can be



(a) (b)

Fig. 9. The performance of execution time based on different number of populations (a) and iteration (b).

split into several components. First of all, we establish a prediction model using GAN to pre-process the vehicles and RSUs in the VEC scenario, reducing the decision space while assisting the task offloading process. Then, we apply an effective offloading scheme based on a Lagrange multiplier-improved non-uniform genetic algorithm to various dynamic scenarios. Next, we develop an adaptive model that adjusts to change in density of vehicles, the load of network, and computing capabilities of vehicles to handle the overall cost in different scenarios. Comprehensive experimental assessments are undertaken to ascertain the adaptability, flexibility, and reliability of ASSIGN scheme. The comparison of the experimental results suggests that our ASSIGN scheme exceeds the alternative offloading scheme with respect to the cumulative overall delay and computation cost. As a future direction, it is interesting to study more realistic scenarios under a wider range of operating conditions.

ACKNOWLEDGMENTS

This work is supported in part by the National Natural Science Foundation of China under Grant 62372310, and in part by the Liaoning Province Applied Basic Research Program under Grant 2023JH2/101300194, and in part by the LiaoNing Revitalization Talents Program.

REFERENCES

[1] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, pp. 450–465, Feb 2018.

[2] K. S. Roy, S. Deb, and H. K. Kalita, "A novel hybrid authentication protocol utilizing lattice-based cryptography for iot devices in fog networks," *Digital Communications and Networks*, 2022.

[3] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," *Mobile Networks and Applications*, vol. 26, pp. 1145–1168, Jul 2020.

[4] Q. He, Z. Feng, H. Fang, X. Wang, L. Zhao, Y. Yao, and K. Yu, "A blockchain-based scheme for secure data offloading in healthcare with deep reinforcement learning," *IEEE/ACM Transactions on Networking*, vol. 32, no. 1, pp. 65–80, 2023.

[5] L. Zhao, T. Li, E. Zhang, Y. Lin, S. Wan, A. Hawbani, and M. Guizani, "Adaptive swarm intelligent offloading based on digital twin-assisted prediction in vec," *IEEE Transactions on Mobile Computing*, 2023.

[6] F. Zeng, Z. Zhang, and J. Wu, "Task offloading delay minimization in vehicular edge computing based on vehicle trajectory prediction," *Digital Communications and Networks*, 2024.

[7] N. Eshraghi and B. Liang, "Joint offloading decision and resource allocation with uncertain task computing requirement," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, IEEE, Apr 2019.

[8] L. Huang, X. Feng, L. Qian, and Y. Wu, "Deep reinforcement learning-based task offloading and resource allocation for mobile edge computing," in *Machine Learning and Intelligent Communications*, pp. 33–42, Springer International Publishing, 2018.

[9] X. An, Y. Li, Y. Chen, and T. Li, "Joint task offloading and resource allocation for multi-user collaborative mobile edge computing," *Computer Networks*, p. 110604, 2024.

[10] J. Du, Y. Sun, N. Zhang, Z. Xiong, A. Sun, and Z. Ding, "Cost-effective task offloading in NOMA-enabled vehicular mobile edge computing," *IEEE Systems Journal*, vol. 17, pp. 928–939, Mar 2023.

[11] F. Liu, J. Chen, Q. Zhang, and B. Li, "Online mec offloading for v2v networks," *IEEE Transactions on Mobile Computing*, 2022.

[12] W. Fan, Y. Su, J. Liu, S. Li, W. Huang, F. Wu, and Y. Liu, "Joint task offloading and resource allocation for vehicular edge computing based on v2i and v2v modes," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 4, pp. 4277–4292, 2023.

[13] K. Tan, L. Feng, G. Dán, and M. Törngren, "Decentralized convex optimization for joint task offloading and resource allocation of vehicular edge computing systems," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 12, pp. 13226–13241, 2022.

[14] Y. Ren, X. Chen, S. Guo, S. Guo, and A. Xiong, "Blockchain-based vec network trust management: A drl algorithm for vehicular service offloading and migration," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 8, pp. 8148–8160, 2021.

[15] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 11, pp. 11158–11168, 2019.

[16] Y. Chen, J. Wu, J. Han, H. Zhao, and S. Deng, "A game-theoretic approach based task offloading and resource pricing method for idle vehicle devices assisted vec," *IEEE Internet of Things Journal*, 2024.

[17] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, 2017.

[18] M. Yang, H. Zhu, H. Qian, Y. Koucheryavy, K. Samouylov, and H. Wang, "Peer offloading with delayed feedback in fog networks," *IEEE Internet of Things Journal*, vol. 8, no. 17, pp. 13690–13702, 2021.

[19] W. Wen, Y. Fu, T. Q. Quek, F.-C. Zheng, and S. Jin, "Joint uplink/downlink sub-channel, bit and time allocation for multi-access edge computing," *IEEE Communications Letters*, vol. 23, no. 10, pp. 1811–1815, 2019.

[20] L. Chen and J. Xu, "Task replication for vehicular cloud: Contextual combinatorial bandit with delayed feedback," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 748–756, IEEE, 2019.

[21] M. Li, J. Gao, L. Zhao, and X. Shen, "Deep reinforcement learning for collaborative edge computing in vehicular networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 4, pp. 1122–1135, 2020.

[22] Y. Sun, X. Guo, J. Song, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Transactions on vehicular technology*, vol. 68, no. 4, pp. 3061–3074, 2019.

[23] Z. Ning, K. Zhang, X. Wang, L. Guo, X. Hu, J. Huang, B. Hu, and R. Y. K. Kwok, "Intelligent edge computing in internet of vehicles: A joint computation offloading and caching solution," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, pp. 2212–2225, Apr 2021.

[24] B. Yang, X. Cao, J. Bassey, X. Li, and L. Qian, "Computation offloading in multi-access edge computing: A multi-task learning approach," *IEEE Transactions on Mobile Computing*, vol. 20, pp. 2745–2762, Sep 2021.

[25] R. Zhao, F. Zhu, Y. Feng, S. Peng, X. Tian, H. Yu, and X. Wang, "OFDMA-enabled wi-fi backscatter," in *The 25th Annual International Conference on Mobile Computing and Networking*, ACM, Aug 2019.

[26] Q. Wu and R. Zhang, "Common throughput maximization in UAV-enabled OFDMA systems with delay consideration," *IEEE Transactions on Communications*, vol. 66, pp. 6614–6627, Dec 2018.

[27] Z. Nan, S. Zhou, Y. Jia, and Z. Niu, "Joint task offloading and resource allocation for vehicular edge computing with result feedback delay," *IEEE Transactions on Wireless Communications*, pp. 1–1, 2023.

[28] J. Kronqvist, D. E. Bernal, A. Lundell, and I. E. Grossmann, "A review and comparison of solvers for convex MINLP," *Optimization and Engi-*

neering, vol. 20, pp. 397–455, Dec 2018.
[29] E. D. L. Rosa and W. Yu, "Restricted boltzmann machine for nonlinear system modeling," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, IEEE, Dec 2015.
[30] S. Eiffert, K. Li, M. Shan, S. Worrall, S. Sukkarieh, and E. Nebot, "Probabilistic crowd GAN: Multimodal pedestrian trajectory prediction using a graph vehicle-pedestrian attention network," *IEEE Robotics and Automation Letters*, vol. 5, pp. 5026–5033, Oct 2020.
[31] M. Srinivas and L. Patnaik, "Genetic algorithms: a survey," *Computer*, vol. 27, pp. 17–26, Jun 1994.
[32] J. Tufto, "Genetic evolution, plasticity, and bet-hedging as adaptive responses to temporally autocorrelated fluctuating selection: A quantitative genetic model," *Evolution*, vol. 69, pp. 2034–2049, Aug 2015.
[33] M. V. B. Delgado and P. P. Marino, "Using non-uniform probability distribution past to improve identification performance in dense RFID reader environments," in *2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, IEEE, Jul 2013.

Liang Zhao (Member, IEEE) is a Professor at Shenyang Aerospace University, China. He received his Ph.D. degree from the School of Computing at Edinburgh Napier University in 2011. Before joining Shenyang Aerospace University, he worked as associate senior researcher in Hitachi (China) Research and Development Corporation from 2012 to 2014. He is also a JSPS Invitational Fellow (2023). He was listed as Top 2 % of scientists in the world by Standford University (2022 and 2023). His research interests include ITS, VANET, WMN and SDN. He has published more than 150 articles. He served as the Chair of several international conferences and workshops, including 2022 IEEE BigDataSE (Steering Co-Chair), 2021 IEEE TrustCom (Program Co-Chair), 2019 IEEE IUCC (Program Co-Chair), and 2018-2022 NGDN workshop (founder). He is Associate Editor of Frontiers in Communications and Networking and Journal of Circuits Systems and Computers. He is/has been a guest editor of IEEE Transactions on Network Science and Engineering, Springer Journal of Computing, etc. He was the recipient of the Best/Outstanding Paper Awards including 2015 IEEE IUCC, 2020 IEEE ISPA, 2022 IEEE EUC and 2013 ACM MoMM, etc.

Tianyu Li is a student at Shenyang Aerospace University, China. He is currently studying for his M.S. degree in Shenyang Aerospace University. His research interests include mobile edge computing, computation offloading and digital-twins.

Guiying Meng is a lecturer at the School of Computer Science, Shenyang Aerospace University, and graduated from Northeastern University in 2006 with a master's degree in computer application technology. Her main research direction is computer network application.

Ammar Hawbani is a Full Professor at the School of Computer Science at Shenyang Aerospace University. He earned his B.S. in Computer Software and Theory from the University of Science and Technology of China (USTC) in 2009. His academic journey continued with an M.S. in 2012 and a Ph.D. in 2016, all from USTC. Following his Ph.D. completion, he served as a Postdoctoral Researcher in the School of Computer Science and Technology at USTC from 2016 to 2019. Later, he worked as an Associate Researcher in the School of Computer Science and Technology at USTC from 2019 to 2023. Currently, he holds the position of Full Professor at the School of Computer Science in Shenyang Aerospace University. His research interests span IoT, WSNs, WBANs, WMNs, VANETs, and SDN.

Geyong Min is a Professor of High-Performance Computing and Networking in the Department of Computer Science at the University of Exeter, UK. He received his PhD in Computing Science from the University of Glasgow, UK, in 2003, and a B.Sc. in Computer Science from Huazhong University of Science and Technology, China, in 1995. His research interests encompass Computer Networks, Wireless Communications, Parallel and Distributed Computing, Ubiquitous Computing, Multimedia Systems.

Ahmed Y. Al-Dubai is a Professor of Networking and Communication Algorithms in the School of Computing at Edinburgh Napier University, UK. He received his PhD degree in Computing from the University of Glasgow in 2004. He leads interdisciplinary research and initiatives in Group Communications, High-performance Networks, Internet of Things, Edge Computing, Future Networks, E-Health, Smart Cities, and Security. His research has received support from EU, Universities UK, the Royal Society, Carnegie Trust, EPSRC, and Scottish Funding Council. Ahmed has received several international awards and has published widely in top-tier journals and prestigious international conference proceedings. He has also served on editorial boards and chaired and co-chaired over 40 international conferences and workshops.

Albert Y. ZOMAYA is the Peter Nicol Russell Chair Professor of Computer Science in the School of Computer Science, Sydney University, and serves as the Director of the Centre for Distributed and High-Performance Computing. Professor Zomaya has published over 800 scientific papers and articles and is the author, co-author, or editor of more than 30 books. He is the past Editor in Chief of the IEEE Transactions on Computers, the IEEE Transactions on Sustainable Computing, and the ACM Computing Surveys. Professor Zomaya received numerous accolades, including Fellowships of the IEEE, AAAS, and the IET. Also, he is a Fellow of the Australian Academy of Science, a Fellow of the Royal Society of New South Wales, a Foreign Member of Academia Europaea, and a Member of the European Academy of Sciences and Arts. His research interests are in parallel and distributed computing, networking, and complex systems.