Proceedings of the 2023 International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/44/20230578

Efficient vehicular networks offloading using Hybrid Localization Algorithm and Deep Reinforcement Learning

Yankai Peng[1,5,*], **Zhiyuan Wang**[2,6,*], **Hailin Li**[3,7], **Ning Dong**[4,8]

[1]Glasgow College, University of Electronic Science and Technology of China, Chengdu, 611731, China
[2]School of Aeronautics and Astronautics, Zhejiang University, Zhejiang, 310058, China
[3]School of Information and Communication Engineering,
Beijing University of Posts and Telecommunications, Beijing, 100876, China
[4]School of Information Science and Engineering, Huaqiao University, Xiamen, 361021, China

[5]2020190902027@std.uestc.edu.cn
[6]3190103755@zju.edu.cn
[7]lihailin027@163.com
[8]Dongning021215@163.com

Abstract. In the context of growing urbanization and increased vehicular traffic, the demand for efficient computation and location-based services is paramount. This paper proposes a pioneering solution to address the challenges of precise location services in Vehicular Networks within urban settings. The system combines a Hybrid Localization Algorithm (HLA) that integrates multiple methods for improved location accuracy with Deep Reinforcement Learning (DRL) for intelligent and adaptive offloading decisions based on real-time traffic conditions. Extensive simulations demonstrate the effectiveness of our approach in reducing response times, optimizing offloading strategies, and alleviating the burden of urban peak vehicle navigation pressure. This research paves the way for enhanced location-based services and intelligent transportation systems in urban areas.

Keywords: Hybrid Localization Algorithm, Vehicular Networks Offloading, Deep Reinforcement Learning

1. Introduction

The exponential growth of urbanization and the escalating vehicular traffic have resulted in a critical demand for efficient computation and location-based services. Urban settings present distinctive challenges for providing precise location services in Vehicular Networks. Accurate and reliable positioning is essential for diverse applications, including navigation, emergency services, and traffic management. This paper introduces an innovative solution designed to address these challenges and revolutionize location services within urban environments.

In response to the complexities of urban Vehicular Networks, we introduce a novel system that synergizes two cutting-edge technologies: the Hybrid Localization Algorithm and Deep Reinforcement Learning. The Hybrid Localization Algorithm integrates multiple methods, harnessing the unique advantages of each to achieve superior location accuracy compared to traditional approaches. Concurrently, Deep Reinforcement Learning facilitates intelligent and adaptive offloading decisions

DOI: 10.54254/2755-2721/44/20230578

grounded on real-time traffic conditions, thereby ensuring resource efficiency and enhanced system performance.

The remainder of the paper is organized as follows. In Section II, we introduce the related work. In Section III, we present the whole system model. In Section IV, we introduced our proposed methods of deep reinforcement learning. In Section V and Section VI, we evaluate the performance of our methods and simulate the whole Vehicle networking system and then introduce our future work. Finally, conclusion is made in Section VII.

2. Related work

With the development of wireless network technology, urban vehicle network puts forward higher requirements for real-time accurate positioning of vehicles. In paper [1] , the mean square error (MSE) helped with the vehicle to be positioned under different dimensions and variances To solve the problem of communication bandwidth and limited computation resource, [2] propose a cooperative location algorithm based on vehicle-road cooperative communication exchange. The algorithm has low computational complexity, easy implementation, stable performance and high reliability. Article [3] propose a distributed cooperative vehicular localization framework with truth discovery, assisting vehicles to learn which neigh- boring nodes they should cooperate with and ignore the others. However, the current research results do not take into account that different tasks have different requirements for positional accuracy. For location information requiring different accu- racy, we can use matching algorithms to make the most efficient use of computational resources. In a congested vehicle environment, each vehicle is an individual, but can be located by means of contact with other vehicles. A new discovery found that adjacent vehicles can cooperate to complete positioning tasks in paper[4], which also propose a approach coined Team Channel-SLAM Evolution (TCSE) to take advantage of the interrelationships between virtual transmitter locations. Paper[5] believed that cooperative vehicles can adapt to the environment and orient themselves better than individual vehicles. In order to improve the accuracy of information connection, an improved centralized and cooperative monocular synchronous localization and mapping (CCM-SLAM) method is proposed. To solve the problem of GNSS (Global Navigation Satellites Systems) outages, page [6] established a fusion localization framework of GNSS/On-board sensors which achieve the accuracy below half a meter during 5s GNSS outage. Another article introduce the basic foundations of intelligent vehicles called High-precision self-localization, in [7] the author used the Monte Carlo localization algorithm to obtain an optimal estimate of the vehicle position.

Because of the limited computing resources can no longer fully meet the needs of vehicles communication, there is a high latency, high energy consumption and low feasibility in the process of information transmission. With the explosive growth of global mobile traffic, traditional fixed cloud cal- culators may not quickly give instructions to vehicle communication methods. Therefore deep reinforcement learning algorithms combined with a variety of technologies to form a new model to solve computation offloading problem. In article[8], a joint task offloading and task migration optimization (JCOTM) algorithm based on deep reinforcement learning is proposed to reduce task processing delay and optimize computing offloading. In [9] authors proposed a shared unloading strategy based on deep reinforcement learning to reduce task unloading delay and energy consumption in complex networked vehicle computing environment. For the problem of limited communication resources, [10] propose a model named vehicular fog computing (VFC) based on the vehicle is designed for communication and computing infrastructure, which help enrich the communication resources and better use of individual vehicle's computing offloading. [11] authors prioritize the experience to have low task service time and high load balance, achieving high Quality of Experience (QoE). In paper[12], a hybrid task offloading scheme (HyTOS) based on deep reinforcement learning is proposed to consider the delay constraints and resource requirements. To improve vehicle communication method, a cooperative computation offload-ing and resource management approach is proposed in [13], meanwhile a deep reinforcement learning algorithm (DRL), namely Asynchronous Advantage Actor-Critic (A3C) is used to optimize the system

DOI: 10.54254/2755-2721/44/20230578

model. And [14], A virtual platform for vehicle trajectory prediction based on deep neural network is developed to achieve reasonable allocation of computing resources.

To cope with tasks with different localization accuracies, we use a hybrid localization algorithm. In order to meet the demand of computing offloading in the heavy traffic flow, we consider a decision-making method to guide the vehicle to choose whether to communicate with the vehicle or the base station, and select the most efficient communication link. We design to deploy the Deep Reinforcement Learning in local data server to reasonable distribute computing and communication resources for the whole vehicle networking system. Experiments show that our proposed algorithm can effectively optimum the decision, achieving rational utilization of computing resources as well as Maximizing spectrum utilization and its improvement is proved by simulation. The main contributions of our work are as follows:

- We present a comprehensive system model that integrates localization, offloading, and vehicular communication to optimize the performance of localization and edge computing system.
- We propose a data-driven approach using hybrid localization and DRL algorithm, which considers both localization and offloading strategy, enabling efficient and adaptive decision-making in the face of the dynamic vehicular environment.
- We demonstrate the effectiveness of our approach through extensive simulations, showing significant improvements in system performance compared to existing methods.

3. System model

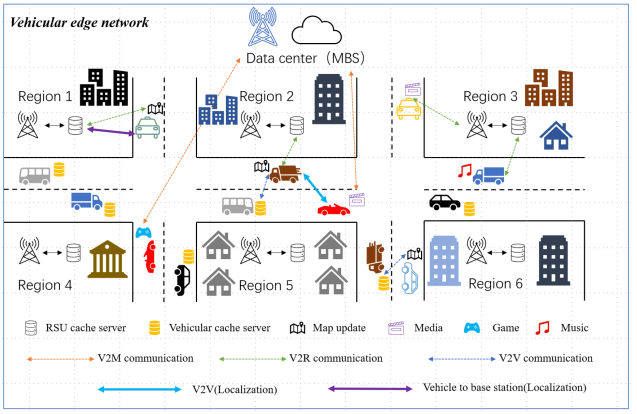

Figure 1: Vehiclular network framework

3.1. Vehicular network model

Vehicular edge computing network has different types of caching nodes, including vehicles, RSUs and remote data center, shown in Fig.1. Suppose there are k vehicles and the computational power of each vehicle can be expressed as f_k^{veh}. There are N RSUs in total and the computation rate of each RSU assigned to the task is $f_{n,l}^{RSU}$. The computation rate of the system for a task assignment is the sum of the local computation rate and the transmission rate of V to RSU. Let $\{C_1, C_2, C_3, ..., C_L\}$ represent L types of tasks and each content includes three features $\{s_l, \tau_l, \beta_l\}$, s_l and τ_l are the size of content and maximum allowed access latency to obtain content, and β_l is the popularity of content. The contents can be computed in three different types of nodes: computing in the local vehicle, in other vehicles and in RSUs.

DOI: 10.54254/2755-2721/44/20230578

3.2. Communication and computation model

In the vehicle communication system, there are uplink and downlink communication between vehicles and RSU. For the communication between vehicle i and vehicle m, the following equation is used to calculate the upstream signal-to-noise ratio $\gamma_{i,m}$

$$\gamma_{i,m} = \frac{p_i g_i}{\xi_{i,m} d_{i,m}^{\kappa} \sigma_{i,m}^2} \tag{1}$$

where p is the transmission power between the two vehicles, g is the antenna gain at the car i, ξ is transmission loss at a reference unit distance, d is the distance between the two vehicles, $\sigma_{i,m}^2$ is the energy of Gaussian white noise introduced during transmission. The downlink signal-to-noise ratio $\gamma_{m,i}$, satisfies the following equation:

$$\gamma_{m,i} = \frac{p_m g_m}{\xi_{i,m} d_{i,m}^{\kappa} \sigma_{i,m}^2} \tag{2}$$

The uplink and downlink transmissions need to satisfy Shannon's formula. Using the following formula to calculate the uplink transmission rate $r_{i,m}$, where $B_{i,m}$ is the uplink channel bandwidth between vehicle i and vehicle m.

$$r_{i,m} = B_{i,m} \log_2 (1 + \gamma_{i,m}) \tag{3}$$

The downlink transmission rate satisfies the following equation, where $B_{m,i}$ is the downlink channel bandwidth between vehicle i and vehicle m:

$$r_{m,i} = B_{m,i} \log_2 (1 + \gamma_{m,i}) \tag{4}$$

The uplink and downlink communication indexes between vehicle and RSU have similar relationship with each parameter, and the signal-to-noise ratio $\gamma_{i,n}$, The uplink and downlink communication indexes between vehicle and RSU have similar relationship with each parameter, and the signal-to-noise ratio $\gamma_{i,n}$, $\gamma_{n,i}$ of the uplink and downlink channels can be calculated by the following equations:

$$\gamma_{i,n} = \frac{p_i g_i}{\xi_{i,n} d_{i,n}^{\kappa} \sigma_{i,n}^2} \tag{5}$$

$$\gamma_{n,i} = \frac{p_n g_n}{\xi_{i,n} d_{i,n}^{\kappa} \sigma_{i,n}^2} \tag{6}$$

where p is the transmission power between the vehicle and the RSU, g is the antenna gain at the vehicle, ξ is the transmission loss per unit distance, d is the distance between the vehicle and the RSU, and $\sigma_{i,n}^2$ is the energy of the Gaussian white noise introduced during transmission. The upstream and downstream transmission rates between the vehicle and the RSU are $r_{i,n}$ and $r_{n,i}$, which satisfies the following equation, where $B_{i,n}$ is the uplink channel bandwidth and $B_{n,i}$ is the downlink channel bandwidth:

$$r_{i,n} = B_{i,n} \log_2 (1 + \gamma_{i,n}) \tag{7}$$

$$r_{n,i} = B_{n,i} \log_2 (1 + \gamma_{n,i}) \tag{8}$$

The task content to be computed in vehicle communication is divided into different types and offloaded to vehicle local computation, other vehicle computation or RSU computation, using the following equation to compute the delay incurred in completing the computation of task l locally in the vehicle, where f_k^{veh} is the computation speed of vehicle k:

$$d_{l,i,i} = \eta_l \frac{\lambda_{i,i} s_{l,i,i}}{f_i^{veh}} \tag{9}$$

DOI: 10.54254/2755-2721/44/20230578

where η_l obeys a two-point distribution $\Pr(\eta_l = 1) = \beta_L, \Pr(\eta_l = 0) = 1 - \beta_L$ and β_L is the probability of generation, $\lambda_{i,i}$ is the proportion of task l that car i offloads to local computation, $s_{l,i,i}$ is the size of task l offloaded by vehicle i. When computational task l is offloaded to other vehicles, the delay $d_{l,i,m}$ satisfies the following equation:

$$d_{l,i,m} = \eta_l \left(\frac{\lambda_{i,m} s_{l,i,m}}{f_m^{veh}} + \frac{\lambda_{i,m} s_{l,i,m}}{r_{i,m}} + \frac{\lambda_{i,m} s_{l,i,m}}{r_{m,i}} \right) \tag{10}$$

When computational task l is offloaded to the RSU, $f_{n,l}^{RSU}$ is the computation rate assigned to task l by the RSU, the resulting delay satisfies the following equation:

$$d_{l,i,n} = \eta_l \left(\frac{\lambda_{i,n} s_{l,i,n}}{f_{n,l}^{RSU}} + \frac{\lambda_{i,n} s_{l,i,n}}{r_{i,n}} + \frac{\lambda_{i,n} s_{l,i,n}}{r_{n,i}} \right) \tag{11}$$

The maximum delay generated by the three computational allocation methods is taken as the total delay generated by the computational task l in vehicular communication, which means the total delay d_l satisfies the following equation:

$$d_l = \max \{ d_{l,i,i}, d_{l,i,m}, d_{l,i,n} \} \tag{12}$$

The total system delay d_{total} is the sum of the total delay d_l of all tasks and all vehicles of the system:

$$d_{total} = \sum_{i=1}^{K} \sum_{l=1}^{L} d_l \tag{13}$$

The computational resource allocation obtained by feeding the tasks to be processed into the deep reinforcement learning algorithm to obtain the delay minimizing computational offloading scheme with the lowest latency, and the resulting computational resource allocation needs to satisfy $\max \{ d_{l,i,i}, d_{l,i,m}, d_{l,i,n} \} \leqslant \tau_l$, where τ_l is the maximum delay. Therefore, the proposed vehicular edge computing problem can be represented as minimize the total delay of the system. The mathematical of the problem can be expressed as follows:

$$\begin{aligned} \mathbf{P0}: & \min \sum_{i=1}^{K} \sum_{l=1}^{L} d_l \\ \text{s.t.} \quad & \mathrm{C1}: \max \{ d_{l,i,i}, d_{l,i,m}, d_{l,i,n} \} \leqslant \tau_l \\ & \mathrm{C2}: f_{n,l}^{RSU} \leqslant f_n^{RSU} \\ & \mathrm{C3}: s_l = \lambda_{i,i} s_{l,i,i} + \lambda_{i,m} s_{l,i,m} + \lambda_{i,n} s_{l,i,n} \\ & \mathrm{C4}: 0 \leqslant \beta_L \leqslant 1 \end{aligned} \tag{14}$$

4. Algorithm design

4.1. Hybrid Localization algorithm

A novel localization algorithm that incorporates both multilateration and Extended Kalman Filter techniques. This hybrid approach dynamically switches between the two methods based on a user-defined threshold. When the localization accuracy falls below the threshold, the algorithm automatically switches to the multilateration algorithm, which excels in providing accurate position estimations under certain conditions. Conversely, when the accuracy surpasses the threshold, the algorithm switches to the more sophisticated Extended Kalman Filter technique, which offers improved performance and robustness in challenging localization scenarios. This adaptive approach ensures optimal localization results in a wide range of environments and conditions, striking a balance between accuracy and computational efficiency.

DOI: 10.54254/2755-2721/44/20230578

4.1.1. Multilateration algorithm The distance between the anchor i and tag t can be expressed as:

$$d_i^2 = (x_i - x_t)^2 + (y_i - y_t)^2 + (z_i - z_t)^2 \tag{15}$$

Organizing this nonlinear equation, we can obtain a linear equation of the following form:

$$\mathrm{AX} = \mathrm{B} \tag{16}$$

$$\mathrm{A} = \begin{bmatrix} x_1 - x_t & y_1 - y_t & z_1 - z_t \\ x_2 - x_t & y_2 - y_t & z_2 - z_t \\ \vdots & \vdots & \vdots \\ x_n - x_t & y_n - y_t & z_n - z_t \end{bmatrix} \tag{17}$$

$$x = \begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix} \tag{18}$$

Solving the above equation by the least squares method:

$$\mathrm{X} = \left(A^T A\right)^{-1} A^T b \tag{19}$$

4.1.2. EKF algorithm The Extended Kalman Filter (EKF) has emerged as an effective choice for localization, particularly when dealing with nonlinear motion models and sensor measurements. The EKF is an extension of the Kalman Filter, which is applicable only to linear systems. Its key idea is to linearize the nonlinear system at each time step using Taylor series expansion. By doing so, the nonlinear system can be transformed into a linear system, allowing the use of the standard Kalman Filter for state estimation. This makes the EKF well-suited for estimating the state of a vehicle in real-time, taking into account nonlinearities in the motion model and sensor measurements, and providing accurate localization even in challenging environments where GPS signals may be unreliable.

The state equation of the EKF can be expressed as

$$x_k = f(x_{k-1}, u_k) + w_k \tag{20}$$

where x_k is the state vector at time step k, $f()$ is a nonlinear function representing the system's state transition, u_k is the control input vector at time step k, w_k is the process noise, representing the uncertainty in the system model. The observation equation can be expressed as

$$z_k = h(x_k) + v_k \tag{21}$$

where z_k is the measurement vector at time step, $h()$ is a nonlinear function representing the observation equation, v_k is the measurement noise, representing the uncertainty in the measurement process.

The EKF algorithm proceeds through two main steps: the prediction step (time update) and the update step (measurement update).

At each time step k, the EKF predicts the next state estimate and error covariance based on the state transition model:

$$\hat{x}_{k|k-1} = f(\hat{x}_{k-1}, u_k) \tag{22}$$

The linearization process involves calculating the Jacobian matrix F_k of the function $f()$ with respect to the state vector x evaluated at the predicted state $\hat{x}_{k|k-1}$. This matrix is used to update the error covariance $P_{k|k-1}$ as follows:

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \tag{23}$$

DOI: 10.54254/2755-2721/44/20230578

where $P_{k-1|k-1}$ is the error covariance matrix at the previous time step $k-1$, Q_k is the process noise covariance matrix.

After obtaining a new measurement z_k at time step k, the EKF uses the observation model to compute the predicted measurement $\hat{z}_{k|k-1} = h(\hat{x}_{k|k-1})$. The Jacobian matrix H_k of the function $h()$ is evaluated at $\hat{x}_{k|k-1}$ and used to calculate the Kalman gain K_k and update the state estimate:

$$K_k = P_{k|k-1}H_k^T(H_kP_{k|k-1}H_k^T + R_k)^{-1} \tag{24}$$

$$hatx_k = \hat{x}_{k|k-1} + K_k(z_k - \hat{z}_{k|k-1}) \tag{25}$$

Finally, the error covariance matrix is updated using the Kalman gain and the observation model:

$$P_k = (I - K_kH_k)P_{k|k-1} \tag{26}$$

where R_k is the measurement noise covariance matrix.

4.2. Deep reinforcement learning algorithm

4.2.1. Algorithm Overview Deep Reinforcement learning is widely used in computation resource offloading, especially in multi-edge computing. We proposed a Random-generated Deep Reinforcement Learning(RG-DRL) to provide proportions of three offloading decisions: V2V computing, local computing in RSUs, and offloading to the Cloud. It consists of online and offline stages.

Algorithm 1 Localization algorithm

Input: The total tasks generate at each time slot
Output: Original tasks with location information

Initialize: Generate L tasks with different priority
for $i = 1, 2, ..., L$ **do**
  if $Priority > Upper - threshold$ **then**
    Implement EKF algorithm and predict the position
  end if
  if $Lower - threshold < Priority < Upper - threshold$ **then**
    Implement multilateration algorithm and predict the position
  end if
  if $0 < Priority < Lower - threshold$ **then**
    Remain the same
  end if
end for

In online stage, a two-dimension state matrix $\mathbf{S}$ of a timeframe is obtained from dataset. Required delay of each type of task $\tau_1, \tau_2, ..., \tau_L$ are claimed in the matrix. It is then expanded into a state vector $\mathbf{s}_t$ as the input of policy network(PN). PN predict a best action vector $\hat{\mathbf{x}}_t$ of the timeframe. Then $\hat{\mathbf{x}}_t$ is quantized into R nearby action vectors for Q-calculator to compute. The vector with highest Q-value Q_t^* is select and is regarded as $\mathbf{x}_t^*$. The corresponding vector pair $\{\mathbf{s}_t, \mathbf{x}_t^*\}$ is combined and stored in replay memory as input and target output of PN.

When a dozen of timeframes pass, online stage pauses, then offline stage starts. A batch of combined vectors are extracted from replay memory to update PN coefficients. The loss function is mean square error(MSE).

DOI: 10.54254/2755-2721/44/20230578

4.2.2. Policy Network In our RG-DRL, the predicted action vector $\hat{x}_t$ is obtained by policy $\pi(\theta_i)$, which is a deep neural network. It consists of one input layer, two linear hidden layers and an output layer that are fully connected. The input layer corresponds with the state vector $\mathbf{s}_t$, and the output layer corresponds with the predicted vector $\hat{\mathbf{x}}_t$. Hidden layers are in size of 1024×1, with ReLU as activation function. The output layer, instead, is activated by Sigmoid in order to be normalized.

4.2.3. Random Generator and Replay Memory Conventional DRL Algorithms are compensate for binary-choice policies. In offloading, however, without a foreseen action space, continuous policies are difficult to be constructed. To create an action space for Q-value calculating, we applied a random action generator(RAG) in the algorithm.

When an action vector $\hat{\mathbf{x}}_t$ is given by the policy network, it is copied into the RAG as the first element $\mathbf{x}_{t,1}$ of action space. Then RAG randomly generates a series of new vectors $\mathbf{x}_{t,i}$ near the former vector. Each proportion in action vector is changed within a maximum range w. As shown in Fig.2, vectors on the top on the RAG is similar to $\mathbf{x}_{t,1}$, whereas those at the bottom are different. After the action space is filled up, the vectors are send to Q-value calculator. Q-value is defined as

$$Q = Q_0 - P_{delay} - P_{miss} - P_{overflow} \tag{27}$$

It consists of one initializing value Q_0 and three penalties: delay penalty P_{delay} reflecting delay caused by the action vector, miss penalty P_{miss} for missing the required delay, and overflow penalty $P_{overflow}$ for allocating too much resources.Penalties are calculated by

$$\begin{aligned} P_{delay} &= \sum_{i=1}^{K}\sum_{l=1}^{L} \lambda_d d_{i,l} \\ P_{miss} &= \sum_{i=1}^{K}\sum_{l=1}^{L} \lambda_m \mathrm{ReLU}\left(d_{i,l} - \tau_l\right) \\ P_{overflow} &= \sum_{k=1}^{L} \lambda_o \mathrm{ReLU}\left(x_{2k} + x_{2k+1}\right) \end{aligned} \tag{28}$$

Each penalty has a weight $\lambda_d, \lambda_m, \lambda_o$ to note its importance. Thus the best nearby action in RAG is

$$\mathbf{x}_t^* = argmax\ Q^*(\mathbf{s}_t, \mathbf{x}_i) \tag{29}$$

The vector $\mathbf{x}_t^*$ indicates that a better proportion exists, hence it's regarded as the target proportion in the timeframe for PN to train and update coefficients θ_i.

4.2.4. Adaptive Settings In the first third of timeframes, learning rate $\alpha = 0.01$ as default, randomize range $w = 0.1$, both can be set manually. Then at each third of timeframes, learning rate and randomize range are both divided by 2. In addition, RAG size R_t is adaptive basing on historical $\{R_1, R_2, ..., R_{t-1}\}$ to avoid unnecessary choices. Consequently, nearby actions in RAG differs from the predicted one become less and PN is trained more finely as training going on.

4.2.5. Evaluation Note that new occasions may always appear in new time frames, bringing unforeseen proportions to PN-DRL, along with possibility that exists in the real best action because of random processes may be missed, which means conventional evaluations such as MSE no longer fit our algorithm.

Hence, we use Q-ratio instead of loss to evaluate our RG-DRL

DOI: 10.54254/2755-2721/44/20230578

$$Q\text{-ratio} = \frac{|\hat{Q}_t - Q_t^*|}{|Q_t^*|} \times 100\% \tag{30}$$

In which $\hat{Q}_t$ refers the Q-value corresponds with $\hat{\mathbf{x}}_t$.

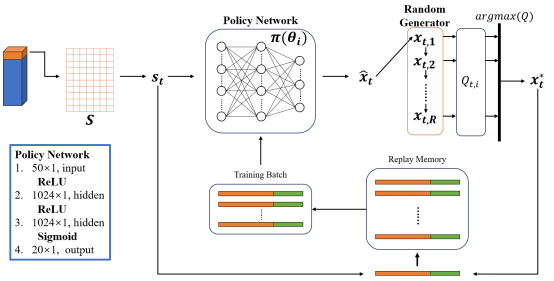

Figure 2: The framework of deep reinforcement learning algorithm

Algorithm 2 DRL algorithm

Input: the state matrix $\mathbf{S}$ at each time-frame t, randomize coefficient ω for random generator, the size of random action space R, total training epochs E, training interval Δ.

Output: Offloading proportion vector $\mathbf{x}_t^*$ with corresponding state vector $\mathbf{s}$

Initialize: empty replay memory, empty random generator, randomize the parameters θ_i in the policy network π_θ.
for $i = 1, 2, ..., E$ **do**
  Predict the action vector $\hat{x}_t$
  Copy $\hat{x}_t$ to the random generator
  Generate $(K-1)$ action vector $\{\mathbf{x}_2, \mathbf{x}_3, ... \mathbf{x}_K\}$ within ω in each step
  Compute Q corresponding each $\mathbf{x}$ in random generator
  Select the optimal action $\mathbf{x}_t^*$ by $\mathbf{x}_t^* = argmax(Q)$
  Storage $\mathbf{s}_t$ and $\mathbf{x}_t^*$ in replay memory
  if i mod $\Delta = 0$ **then**
    Randomly sample a batch of data set from replay memory
    Train the the policy network π_θ with the batch, update θ_i with Adam optimizer
  end if
end for

5. Numerical results

We validate our hybrid localization algorithm using a real dataset, the algorithm is implemented using Matlab. We randomly generate a series of tasks and assign a value between zero and one. When this value is greater than 0.3, it means that this task needs to get the real-time position. When the value is between 0.3 and 0.6, it represents that the location of this task is realized by the multipoint localization algorithm, and when the value is greater than 0.6, it represents that the location of this task is realized by the EKF algorithm. A value less than 0.3 means that the task does not need location information. We then send the results to the DRL algorithm for task offloading. Fig.3-4 shows the location maps of the vehicles corresponding to the required location information after randomly generating 10 and 20 tasks, respectively.

DOI: 10.54254/2755-2721/44/20230578

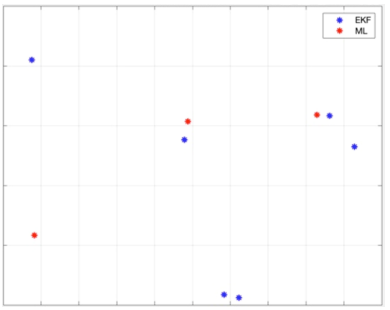

Figure 3: The position of vehicles when 10 tasks were randomly generated

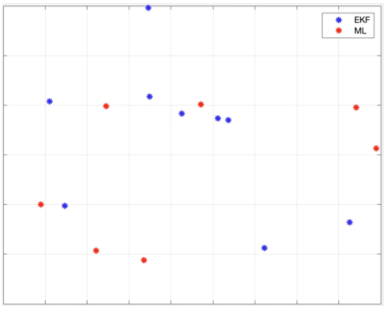

Figure 4: The position of vehicles when 20 tasks were randomly generated

For the result of tasks offloading, The Q-ratio of 100,000 and 1,000,000 timeframes are shown in Fig.6 and Fig.7. Its MSE loss in 1,000,000 timeframes is shown in Fig.5.

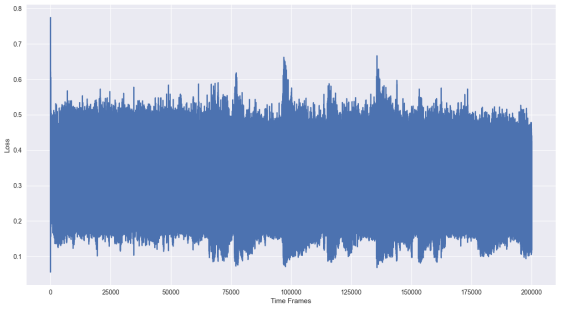

Figure 5: MSE loss

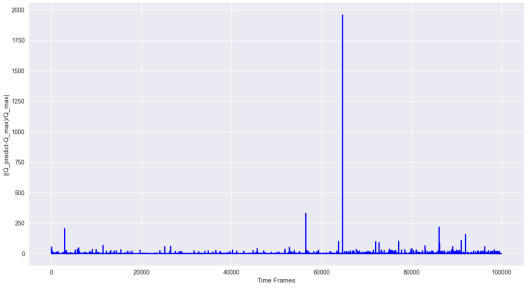

Figure 6: The Q-ratio of 100,000 timeframes

DOI: 10.54254/2755-2721/44/20230578

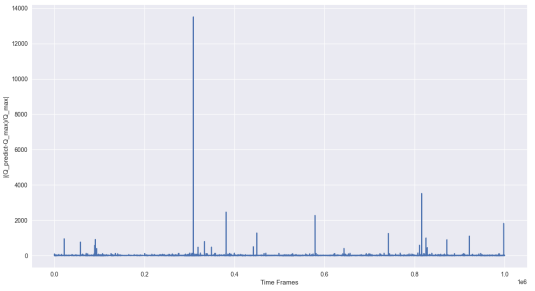

Figure 7: The Q-ratio of 1,000,000 timeframes

It is clear that outliers always appear due to RAG, most possibly because the maximum Q-value among RAG vectors happened to be close at 0. Additionally, MSE loss oscillates even at the end of training, demonstrating our former analysis. However, after 1,000,000 timeframes,despite outliers, most of the Q-ratio given by the trained PN are less than 0.2 in testing results, as shown in Table 1

Table 1: Distribution of output.

Q-ratio	Proportion
$0 \leq$ Q-ratio ≤ 0.2	95.62%
other	4.38%

6. Future work

In the future, we hope to find or collect more accurate and comprehensive data to further validate our methods. In addition, for the localization algorithm, we hope to design a fusion localization algorithm that combines both GPS and UWB technologies.

For our DRL algorithm, one apparent problem is that the Q-ratio in RG-DRL does not converge very fast and robust due to the instability caused by RAG as a sacrifice for continuous proportion prediction. A practical solution might be some improvement in RAG generating mechanisms. storing some of the nearby actions generated by RAG in replay memory. Another access to better results can be a change in algorithm architecture, by replacing RAG and Q-value calculator with another DNN with random generator, thus an actor-critic-like DRL will be constructed. This actor-critic RG-DRL shall be more robust to outliers, as well as setting more appropriate weights to Q-value calculation.

7. Conclusion

This research proposes an innovative solution to address the challenges of precise location services in Vehicular Networks within urban settings. The system combines a Hybrid Localization Algorithm that integrates multiple methods for improved location accuracy with Deep Reinforcement Learning for intelligent and adaptive offloading decisions based on real-time traffic conditions. Extensive simulations demonstrate the effectiveness of our approach in reducing response times, optimizing offloading strategies, and alleviating the burden of urban peak vehicle navigation pressure. This research paves the way for enhanced location-based services and intelligent transportation systems in urban areas, providing a foundation for future advancements in smart mobility and urban management.

DOI: 10.54254/2755-2721/44/20230578

Acknowledgement

This paperhas two co-authors and their equivalent contribution to this paper are: Yankai Peng(design and realization of hybrid locolization) and Zhiyuan Wang(design and realization of RG-DRL).

We would like to express our sincere gratitude to all those who have supported and assisted us throughout this project. First and foremost, we would like to extend my heartfelt appreciation to our supervisor, Professor Danijela Cabric. Throughout the course of this project, your unwavering guidance and invaluable advice have been instrumental in helping us overcome challenges and achieve significant progress. Your expertise and patient mentorship have had a profound impact on us, and we are grateful for the lifelong benefits. We would also like to thank our teaching assistants for the help they have provided us with, giving us a lot of academic advice and allowing us to develop a careful and rigorous approach to our research! Lastly, I extend my appreciation to the institution that provided resources and support for this project.

References

[1] F. Wang, G. Yin, L. Xu, W. Zhuang, Y. Liu and J. Liang, "Distance-Based Cooperative Localization of Connected Vehicles Via Convex Relaxation Under Extreme Environments," 2021 5th CAA International Conference on Vehicular Control and Intelligence (CVCI), Tianjin, China, 2021, pp. 1-5, doi: 10.1109/CVCI54083.2021.9661208.

[2] J. Li and N. Ma, "Design of Vehicle Cooperative localization System Based on Cooperative Communication Switching Strategy," 2021 17th International Conference on Computational Intelligence and Security (CIS), Chengdu, China, 2021, pp. 237-241, doi: 10.1109/CIS54983.2021.00057.

[3] F. Wen and T. Svensson, "Collaborative Localization with Truth Discovery for Heterogeneous and Dynamic Vehicular Networks," 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring), Antwerp, Belgium, 2020, pp. 1-5, doi: 10.1109/VTC2020-Spring48590.2020.9128766.

[4] X. Chu et al., "Joint Vehicular Localization and Reflective Mapping Based on Team Channel-SLAM," in IEEE Transactions on Wireless Communications, vol. 21, no. 10, pp. 7957-7974, Oct. 2022, doi: 10.1109/TWC.2022.3163071.

[5] S. Wen, J. Chen, F. R. Yu, F. Sun, Z. Wang and S. Fan, "Edge Computing-Based Collaborative Vehicles 3D Mapping in Real Time," in IEEE Transactions on Vehicular Technology, vol. 69, no. 11, pp. 12470-12481, Nov. 2020, doi: 10.1109/TVT.2020.3019061.

[6] L. Gao, L. Xiong, X. Xia, Y. Lu, Z. Yu and A. Khajepour, "Improved Vehicle Localization Using On-Board Sensors and Vehicle Lateral Velocity," in IEEE Sensors Journal, vol. 22, no. 7, pp. 6818-6831, 1 April1, 2022, doi: 10.1109/JSEN.2022.3150073.

[7] Z. Wang, J. Fang, X. Dai, H. Zhang and L. Vlacic, "Intelligent Vehicle Self-Localization Based on Double-Layer Features and Multilayer LIDAR," in IEEE Transactions on Intelligent Vehicles, vol. 5, no. 4, pp. 616-625, Dec. 2020, doi: 10.1109/TIV.2020.3003699.

[8] Z. Wu and D. Yan, "Deep reinforcement learning-based computation offloading for 5G vehicle-aware multi-access edge computing network," in China Communications, vol. 18, no. 11, pp. 26-41, Nov. 2021, doi: 10.23919/JCC.2021.11.003.

[9] X. Peng et al., "Deep Reinforcement Learning for Shared Offloading Strategy in Vehicle Edge Computing," in IEEE Systems Journal, vol. 17, no. 2, pp. 2089-2100, June 2023, doi: 10.1109/JSYST.2022.3190926.

[10] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin and S. Chen, "Vehicular Fog Computing: A Viewpoint of Vehicles as the Infrastructures," in IEEE Transactions on Vehicular Technology, vol. 65, no. 6, pp. 3860-3873, June 2016, doi: 10.1109/TVT.2016.2532863.

[11] J. Park and K. Chung, "Collaborative Computation Offloading Scheme Based on Deep Reinforcement Learning," 2023 International Conference on Information Networking (ICOIN), Bangkok, Thailand, 2023, pp. 110-115, doi: 10.1109/ICOIN56518.2023.10048957.

[12] C. Wu, Z. Huang and Y. Zou, "Delay Constrained Hybrid Task Offloading of Internet of Vehicle: A Deep Reinforcement Learning Method," in IEEE Access, vol. 10, pp. 102778-102788, 2022, doi: 10.1109/ACCESS.2022.3206359.

[13] L. Lu, X. Li, J. Sun and Z. Yang, "Cooperative Computation Offloading and Resource Management for Vehicle Platoon: A Deep Reinforcement Learning Approach," 2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys), Hainan, China, 2022, pp. 1641-1648, doi: 10.1109/HPCC-DSS-SmartCity-DependSys57074.2022.00249.

[14] G. Ma, X. Wang, M. Hu, W. Ouyang, X. Chen and Y. Li, "DRL-Based Computation Offloading With Queue Stability for Vehicular-Cloud-Assisted Mobile Edge Computing Systems," in IEEE Transactions on Intelligent Vehicles, vol. 8, no. 4, pp. 2797-2809, April 2023, doi: 10.1109/TIV.2022.3225147.