Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH **Open Access**

Blockchain enabled task offloading based on edge cooperation in the digital twin vehicular edge network

Chunhai Li[1*], Qiyong Chen[1], Mingfeng Chen[2], Zhaoyu Su[2], Yong Ding[3], Dapeng Lan[4] and Amir Taherkordi[4]

Abstract
The rapid development of the Internet of Vehicles (IoV) along with the emergence of intelligent applications have put forward higher requirements for massive task offloading. Even though Mobile Edge Computing (MEC) can diminish network transmission delay and ease network congestion, the constrained heterogeneous resources of a single edge server and the highly dynamic topology of vehicular edge networks may compromise the efficiency of task offloading, including latency and energy consumption. Vehicular edge networks are also vulnerable to malicious outside attacks. In this paper, we propose a new blockchain-enabled digital twin vehicular edge network (DTVEN) where digital twin (DT) is exploited to monitor network communication, computation, and caching (3C) resources management in real time to provide rich data for offloading decision-making, and blockchain is utilized to secure fair and decentralized offloading transactions among DTs. To ensure 3C resources sharing across edge servers, we design a DT-assisted edge cooperation scheme, which makes full use of edge resources in vehicular networks. Furthermore, a DT-based smart contract is built to achieve a quick and effective consensus process. Then, we apply a task offloading algorithm based on an improved cuckoo algorithm (ICA) and a resource allocation scheme based on greedy strategy to minimize network cost by comprehensively taking into account latency and energy consumption. Numerical results demonstrate that our proposed scheme outperforms the existing schemes in terms of network cost.

Keywords Digital twin, Blockchain, Edge cooperation, Improved cuckoo algorithm

Introduction

The Internet of Vehicles (IoV) is a crucial technology for implementing Intelligent Transportation System (ITS) based on in-vehicle network, inter-vehicle network, and mobile Internet [1, 2]. In recent years, with the continuous development of IoV and the commercialization of the fifth-generation (5G) networks, vehicles become more intelligent and efficient, offering users safe, comfortable, and intelligent travel experiences as well as traffic services [3]. Generally, these computationally intensive and delay-sensitive service applications require a large amount of communication, computation, and caching (3C) resources, but the computing capacity of vehicular terminal devices is difficult to meet the requirements of these applications, which will have an impact on the Quality of Service (QoS) and Quality of Experience (QoE) in IoV, as well as the safety of vehicle driving [4].

Mobile edge computing (MEC) can reduce task transmission time by deploying edge servers (ESs) with computation and caching resources at the edge of the network [5]. With the explosive increase of tasks, a single ES cannot process multiple tasks simultaneously.

*Correspondence:
Chunhai Li
chunhaili@guet.edu.cn
[1] School of Information and Communication, Guilin University of Electronic Technology, Guilin, China
[2] School of Artificial Intelligence, Guilin University of Electronic Technology, Guilin, China
[3] School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin, China
[4] Department of Informatics, University of Oslo, Oslo, Norway

Edge cooperation can effectively make up for the lack of 3C resources of a single ES. At the same time, edge cooperation processes tasks in a distributed and collaborative manner, which can take advantage of idle network resources and save system power consumption and time overhead [6, 7]. However, task offloading in IoV faces some critical challenges such as the randomness of wireless channels, the heterogeneity of collaborative nodes, and the high mobility of vehicles. These challenges seriously restrict the implementation of task offloading, resource allocation, and edge cooperation, and ultimately limit the long-term development of IoV.

Digital Twin (DT) as an emerging technology, which can map the physical world to the digital world, enables communication, collaboration, and information sharing between the physical and digital worlds, thereby monitoring the state of the entire network in real time [8]. DT can collect massive sensing data from IoV, then model and analyze it to obtain a global perspective, and publish control strategies for IoV. IoV can utilize powerful computation, communication, and control functions of DT to conduct unified scheduling management and obtain a vision related to the future of the system, making it possible to truly achieve the collaboration between the physical world and digital world throughout the life cycle of IoV [9–11]. However, frequent wireless communication between IoV nodes makes the network vulnerable to attack. If the sensed data is maliciously falsified, the constructed DT will not reflect the actual status of the physical entity, which may lead to serious accidents.

As a distributed ledger, blockchain can achieve information fusion among all parties in IoV because of its non-deleting, non-tampering, and openness characteristics [12–14]. The information collected from IoV will be uploaded to the blockchain platform that provides data security through encryption technology, forming a decentralized and distributed big data sharing network. Its decentralized and transparent data management approach has led to a significant reduction in transaction costs for vehicle management centers and data management centers [15, 16].

However, there are some important challenges for edge cooperation in IoV. First, the dynamic characteristics in a time-varying IoV environment will have an impact on task offloading decision. Second, edge cooperation in IoV is prone to external attacks and information leakage. Third, how to make full use of the limited resources of heterogeneous edge servers is still worth research.

Therefore, in this paper, we propose a new vehicular edge network based on digital twin and blockchain, which optimizes edge cooperation and secures task offloading. Different from the previous studies, we introduce digital twin technology to map the status of physical entities in IoV into the virtual space and construct a digital twin model of task offloading based on blockchain, thereby improving resource sharing security. To satisfy the dynamic vehicular edge network, we design an edge cooperation-based task offloading scheme, which minimizes network cost under the constraints of latency and energy consumption. The key contributions of this paper can be listed as follows.

1 We propose a novel blockchain empowered digital twin vehicular edge network (DTVEN), in which DTs assist resource scheduling by capturing network dynamics while strengthening the security of resource sharing through blockchain.
2 We design an edge cooperation-based task offloading scheme to better reduce offloading latency and energy consumption, and construct a DT-driven blockchain consensus model to ensure the reliability of vehicles and edge servers.
3 We propose a task offloading algorithm based on an improved cuckoo algorithm (ICA) and a resource allocation scheme based greedy strategy to jointly optimize the latency and energy consumption during the task offloading process. The simulation analysis is conducted to show the effectiveness of our proposed scheme.

The rest of this paper is organized as follows. In "Related works" section, we review the related work. In "System model" section, we propose the system model. In "Performance analysis" section, we describe the performance analysis of the system. In "Algorithms" section, we illustrate the algorithms in detail. In "Performance evaluation" section, we discuss the performance of the proposed scheme. In "Conclusion" section, we summary this paper.

Related works

In order to improve the QoE performance of vehicles, some researchers have studied the advantages of edge cooperation in improving the performance of the IoV system. Literature [17] develops a blockchain-based collective learning framework, in which distributed vehicles can train the machine learning models locally and upload to blockchain network to utilize the collective intelligence while avoiding large amounts of data transmission. Considering there is a large number of parking vehicles with abundant and undeveloped resources in the city, Ma et al. [18] use parking vehicles as virtual edge servers to assist ESs in performing tasks. A task scheduling algorithm combined with ESs selection and resource allocation is designed to provide a more efficient and stable offloading service in the case of a large number of task requests. To solve the load imbalance among the ESs caused by the uneven geographical distribution of the vehicles, the authors in [19] present a joint task offloading and resource allocation method for vehicular edge computing (VEC) via edge cooperation, in which the tasks offloaded to a high-load ES can be further offloaded to the other low-load ESs. Duan et al. [20] construct a hierarchical model with QoS-aware and energy-aware resource management for the cooperative edge computing-based intelligent vehicular network (CEC-IoV). They propose a Minimum Latency with Migration Loads (MLML) scheme for load balance among multiple MECs to obtain lower system delay and higher energy efficiency. Literature [21] studies a new framework of hierarchical multi access edge computing based on blockchain, which can reach effective consensus among blockchain nodes while ensuring the performance of MEC system and blockchain. A trust model is proposed to greatly ensure the security of communication links between vehicles.

Since the integration of DT into IoV is beneficial to the system, some scholars have exploited DT to promote the network performance of IoV. To prevent edge computing device (ECD) overload in DT-empowered IoV, Xu et al. [22] study a multiuser offloading system and propose a service offloading (SOL) method with deep reinforcement learning for optimal offloading decisions. In [23], a vehicular edge computing network combined with DT and artificial intelligence (AI) is presented to match potential edge services, in which a coordination graph-driven task offloading scheme based on Multi-agent Deep Deterministic Policy Gradient is designed to decrease offloading costs. Dai et al. [24] integrate DT into VEC networks to adaptively conduct network management and strategy schedule, thereby minimizing the overall offloading latency of the system. The authors in [25] present a holistic network virtualization model that integrates DT and network slicing, in which an environment-aware offloading method is designed to reduce the total time of the system. Liu et al. [26] propose a DT-assisted edge intelligent collaboration scheme in IoV to realize optimal 3C resources allocation and edge collaboration. Considering the deviations between the physical world and the digital world, they utilize a DRL algorithm to obtain the optimal offloading strategy and diminish the response delay.

The above studies have proved that digital twin and blockchain are both promising technologies. Different from these studies, we integrate digital twin and blockchain into edge cooperation in IoV, which is conducive to efficient and secure edge computing. In addition, this paper minimizes network cost by jointly optimizing offloading cost and consensus cost.

System model

In this section, We first propose the system model of blockchain-enabled vehicular edge network architecture. Then, we describe the detailed process of blockchain-based vehicular edge computing. The main notations used in this paper are summarized in Table 1.

Network model

As shown in Fig. 1, the proposed system model consists of three layers, namely, an end layer, an edge layer, and a digital twin layer.

On the end layer, vehicles equipped with multiple sensors and onboard units can collect various valuable information about other vehicles, roads, and the surrounding environment, including video music, entertainment games, and traffic situation. Vehicles can locally process their own generated tasks by consuming their computation resources. Each task is represented as $Ta^v = \{D_D, Z_C, Y_{max}\}$. D_D is the data size of the task, Z_C is the CPU cycles required to perform the task, and Y_{max} means the maximum deadline for executing tasks. However, due to the limited computing capacity, vehicles divide a task into several subtasks and offload them to different edge servers on the edge layer through vehicle-to-infrastructure (V2I) communication.

On the edge layer, multiple edge servers with 3C resources are uniformly distributed. We use the set $ES = \left\{ES_1, ES_2, \cdots, ES_j, \cdots, ES_M\right\}$ to represent edge servers in the model. Owing to the constrained and heterogeneous resources, each edge server has different capabilities for performing tasks. By utilizing 3C resources, edge servers can provide high-quality V2I transmission, efficient task execution, and reliable block consensus for vehicles. In the meantime, each edge server is equipped with a blockchain to ensure that untrusted vehicles are able to offload tasks in a secure manner. All edge servers collaborate to maintain the blockchain, store relevant data, and manage the participating nodes of the network architecture.

On the digital twin layer, digital twins that are highly similar to vehicles and edge servers are constructed

Table 1 Main notations

Notation	Description
DT^v	The digital twin of the vehicle v
DT_j^E	The digital twin of the edge server ES_j
T_v,E_v	The local computing time and energy consumption of v
T_j,E_j	The edge computing time and energy consumption of v
T_{TO},E_{TO}	The time and energy consumption for task offloading
U_{TO}	The cost for task offloading
T_{bp},E_{bp}	The time and energy consumption for block production
T_{bb},E_{bb}	The time and energy consumption for block broadcast
T_{bv},E_{bv}	The time and energy consumption for block verification
T_{bc},E_{bc}	The time and energy consumption for block confirmation
U_{BC}	The cost for blockchain consensus
U_{Net}	The network cost

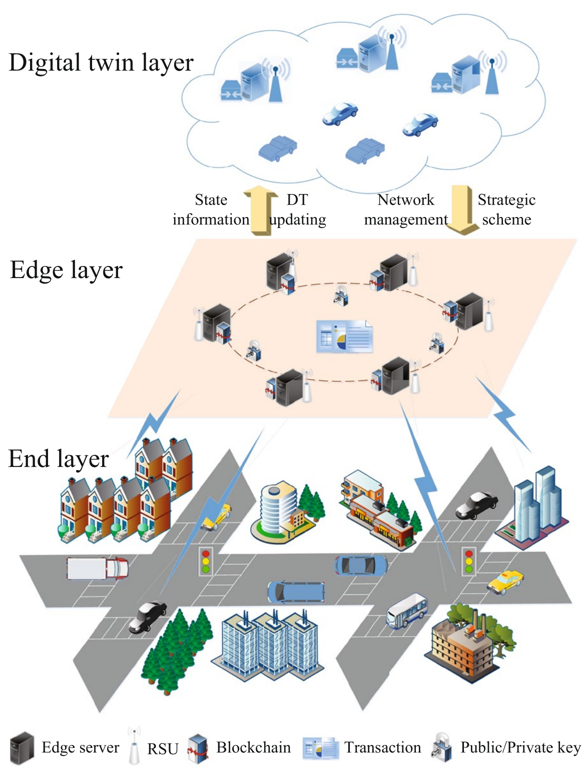

Fig. 1 System model

based on the state information of the physical entities on the end layer and edge layer. To maintain consistency with physical entities, the digital twin maps the physical devices to virtual space and updates them in real time. As shown in Fig. 2, DT can simulate the basic functions of the end layer and edge layer, which realizes more effective offloading decisions during the entire task execution process. The twins on the digital twin layer are modeled and maintained by edge servers on the edge layer. In this paper, we consider two types of digital twins, namely, the digital twins for vehicles and the digital twins for edge servers.

The digital twin of the vehicle is constructed on the digital twin layer with the help of the edge servers. By collecting and processing the operation status of the vehicle, including available computation resources and locations, the historical and current behavior of the vehicle is dynamically presented in digital form.

$$DT^{v} = \left\{Ta^{v}, f^{v}, l^{v}\right\} \tag{1}$$

where f^{v} is the estimated computing capability of the vehicle, $l^{v}=\left(x^{v}, y^{v}\right)$ is the location of the vehicle.

The digital twin of the edge server is a digital replica of the edge server that can reflect the current state of the edge layer. It constantly interacts with edge servers and updates itself based on actual network topology, task requests from vehicles, and so on. At the same time, the digital twin model of the edge server is stored in the digital twin layer through the edge server. For the vehicle ES_j, its DT can be represented as

$$DT_j^{E} = \left\{C_j^{1}, C_j^{2}, C_j^{3}, l_j^{E}\right\} \tag{2}$$

where C_j^1, C_j^2, and C_j^3 are the estimated communication, computing, and caching resources of the vehicle respectively, $l_j^E=\left(x_j^E, y_j^E\right)$ is the location of the edge server.

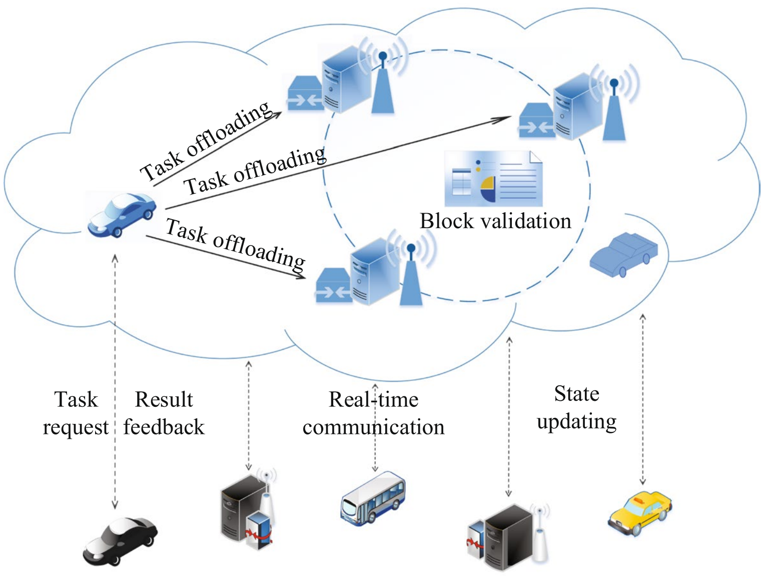

Fig. 2 Task offloading based on digital twin

In terms of data sensing, accurate global sensing is still difficult, because electromagnetic information transmission is subject to losses due to absorption and scattering in wireless channels, which cannot fully reflect the status of physical entities. The loss of state data further leads to deviations between the estimated value of the twins and the true value of the physical entity state. To calibrate the deviation of the digital twin, the resource deviation is represented by a matrix ΔC of size $3 \times M$

$$\Delta C = \begin{bmatrix} \Delta C_1^1 & \dots & \Delta C_j^1 & \dots & \Delta C_M^1 \\ \Delta C_1^2 & \dots & \Delta C_j^2 & \dots & \Delta C_M^2 \\ \Delta C_1^3 & \dots & \Delta C_j^3 & \dots & \Delta C_M^3 \end{bmatrix}$$

where the first, second, and third rows of the matrix ΔC represent the deviation values of communication, computation and caching resources for each edge server, respectively.

Blockchain-based vehicular edge computing

Blockchain technology can prevent data leakage from vehicles and edge servers while ensuring safe and reliable task offloading. Since the digital twin network is built on the edge layer, we can use blockchain to trace the resource allocation between the digital twins.

To achieve task offloading and resource allocation by blockchain technology, vehicles and edge servers must first register their unique accounts and create corresponding keys. The identities of vehicles and edge servers are established through elliptic curve digital signature technology and asymmetric encryption algorithms. Specifically, vehicles and edge servers need to register their legal identities after passing the authentication of the blockchain registration authority. A legal identity consists of a public key, a private key, and a certificate. The public key is used as the source address of the offloading transaction to verify the authenticity of the transaction. The key is regarded as transaction signing, and the certificate is used to identify the unique vehicle and edge server by binding their registration information.

As shown in Fig. 3, we use smart contracts to complete each transaction and store it on the blockchain to ensure the traceability of transaction information. Based on digital twin and blockchain, safe and efficient task offloading in vehicular edge networks can be achieved through the following stages.

When a new task is generated, the vehicle sends an encrypted task offloading request to its corresponding digital twin DT^v. The request message for task offloading of the vehicle can be described as

$$Req^{v \to DT^v} = En_{PK_{DT^v}}(PK_v, Sign_v, ICert_v, ts) \tag{3}$$

where $En_{PK_{DT^v}}$ indicates that the message $Req^{v \to DT^v}$ is encrypted with a public key PK_{DT^v}. $Sign_v = Si_{SK_v}(Ta^v, l^v)$ indicates that the digital signature of the task information Ta^v and vehicle location l^v is encrypted with a private key SK^v. $ICert_v$ is the unique v identifier between the vehicle v and its twin DT^v, and ts is the timestamp of the current message.

In the digital twin network, as a task publisher, DT^v broadcasts task details to the digital twin $DT_j^E (j \in [1, M])$ of the edge server ES_j

$$Boc^{DT^v} = En_{PK_{DT_j^E}}(PK_{DT^v}, Sign_{DT^v}, Dep_v, Cert_{DT^v}, ts) \tag{4}$$

where $En_{PK_{DT_j^E}}$ indicates that the message Boc^{DT^v} is encrypted with a public key $PK_{DT_j^E}$. $Sign_{DT^v} = Si_{SK_{DT^v}}(D_D, Z_C, Y_{max}, l^v)$ represents that the digital signature of the detailed status information of the edge server is encrypted with a private

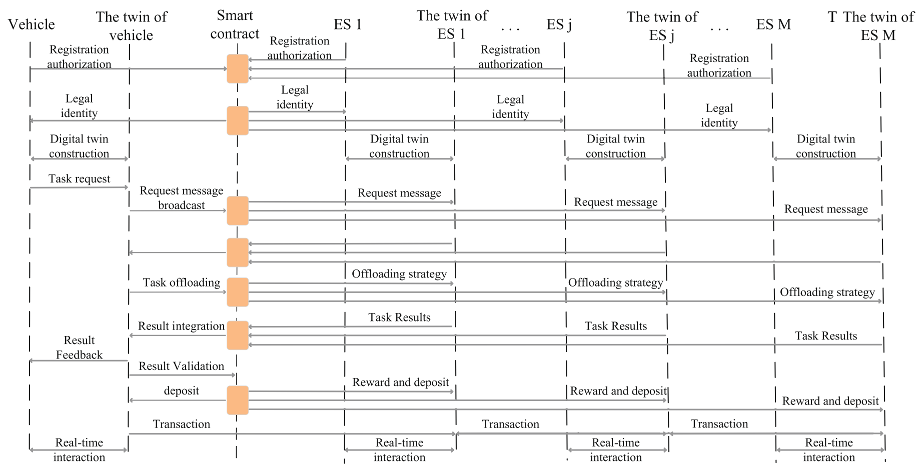

Fig. 3 Smart contract for secure vehicular edge computing

key $SK_{DT_j^E}$. Dep_v is the deposit submitted by the vehicle. $Cert_{DT^v}$ is the identification of the vehicular twin DT^v, i.e., digital certificate, and ts is the timestamp of the current message.

As a task executor, the twin $DT_j^E (j \in [1, M])$ of edge server responds to broadcast information of DT^v and sends its detailed information such as its own resources and location, to DT^v for edge resource allocation

$$Rep^{DT_j^E} = En_{PK_{DT^v}}(PK_{DT_j^E}, Sign_{DT_j^E}, Dep_j, Cert_{DT_j^E}, ts) \quad (5)$$

where $En_{PK_{DT^v}}$ indicates that the message $Rep^{DT_j^E}$ is encrypted with a public key PK_{DT^v}. $Sign_{DT_j^E} = Si_{SK_{DT_j^E}}(C_j^1, C_j^2, C_j^3, l_j^E)$ indicates that the digital signature of detailed status information about the edge server is encrypted with a private key $SK_{DT_j^E}$. Dep_j is the deposit submitted by the edge server. $Cert_{DT_j^E}$ is the identification of the edge server twin DT_j^E, i.e., digital certificate, and ts is the timestamp of the current message.

After receiving the response message, DT^v verifies the identity of the twin $DT_j^E (j \in [1, M])$. To accelerate the verification process, a batch verification process is used to verify the validity of multiple identities simultaneously [27]. Next, DT^v performs subtask offloading and task-resource matching

$$Sol^{DT^v \to DT_j^E} = En_{PK_{DT_j^E}}(PK_{DT^v}, OR_j, Ta^v, Cert_{DT^v}, ts) \quad (6)$$

where $En_{PK_{DT_j^E}}$ indicates that the message $Sol^{DT^v \to DT_j^E}$ is encrypted with a public key $PK_{DT_j^E}$. OR_j represents the proportion of tasks offloaded to the twin DT_j^E. The digital signatures of OR_j and Ta^v are encrypted with a private key SK_{DT^v} through $Si_{SK_{DT^v}}(OR_j, Ta^v)$.

$DT_j^E (j \in [1, M])$ verifies the validity of its identity based on the digital certificate of DT^v, and then executes the task according to the offloading scheme message sent by DT^v and feedbacks the task result message

$$Res^{DT_j^E \to DT^v} = En_{PK_{DT^v}}(PK_{DT_j^E}, RES_j, Cert_{DT_j^E}, ts) \quad (7)$$

where $En_{PK_{DT^v}}$ indicates that the message $Res^{DT_j^E \to DT^v}$ is encrypted with a public key PK_{DT^v}. The digital signature of the subtask result RES_j is encrypted with a private key $SK_{DT_j^E}$ through $Si_{SK_{DT_j^E}}(RES_j)$.

After receiving the feedback results of all subtasks, DT^v integrates them into the total task result RES_{all} and feedbacks them to the vehicle v

$$Feb^{DT^v \to v} = En_{PK_v}(PK_{DT^v}, RES_{all}, ICert_{DT^v}, ts) \quad (8)$$

where En_{PK_v} indicates that the message $Feb^{DT^v \to v}$ is encrypted with a public key PK_v. The digital signature of total task results RES_{all} is encrypted with a private key SK_{DT^v} through $Si_{SK_{DT^v}}(RES_{all})$. $ICert_{DT^v}$ is the unique DT^v identifier between the vehicle v and its twin DT^v, and ts is the timestamp of the current message.

The smart contract verifies the validity of the task results. If the result is verified successfully, the rewards in the smart contract will be sent to the digital twins of the edge servers and the deposits submitted by them will be returned. If the task is not completed or its execution deadline is exceeded, the deposit of DT^v will be refunded. Furthermore, the deposit of DT_j^E will be confiscated as a penalty to prevent DT_j^E from inaction or passive action after receiving the offloading scheme.

At the same time, DT^v generates a transaction to record the task offloading event, and sends the transaction to DT_j^E. DT_j^E validates the received transaction and cryptographically broadcasts it to the blockchain network

$$Tra^{DT^v \to DT_j^E} = En_{PK_{DT_j^E}}(HA, Sol^{DT^v}, Cert_{DT^v}, Cert_{DT_j^E}, ts) \quad (9)$$

where $En_{PK_{DT_j^E}}$ indicates that the message $Tra^{DT^v \to DT_j^E}$ is encrypted with a public key $PK_{DT_j^E}$. HA is its hash value. The digital signatures of all offload schemes sent by DT^v are encrypted with the private key $SK_{DT_j^E}$ through $Si_{SK_{DT_j^E}}(Sol^{DT^v})$

Newly generated transactions are broadcast across the blockchain network for review and verification. Validated transactions are sorted and batch processed into encrypted and tamper-proof blocks. Blocks are connected in linear chronological order via hash pointers to form a blockchain. During the consensus process, the digital twin DT_j^E creates the corresponding blocks. Then, DT_j^E broadcasts the block with a timestamp for block auditing while verifying the correctness of the newly created block.

Performance analysis

In this section, The partial offloading technology based on blockchain is used to perform tasks. We will describe the local computing model, edge computing model, and blockchain consensus model in detail.

Local computing model

Since partial offloading can collaboratively utilize the computation resources of the vehicle and edge servers, it also benefits from both parallel computing [28]. Therefore, when vehicles with constrained resources cannot perform tasks with large amounts of data, a partial offloading model is used to divide the task into several subtasks. It is assumed that tasks are data partition oriented and can be divided arbitrarily on their input bits [29]. Similar to the literature [30], we ignore the task partitioning cost and divide vehicular tasks. Some subtasks are processed locally on the vehicle, while the remaining subtasks are offloaded to the edge servers for execution. Considering the mapping of the digital twin of the vehicle, the generated estimated value has a deviation from the vehicular real value. When the proportion of tasks to be processed by the vehicle is θ_v, the local computing time is

$$T_v = \frac{\theta_v Z_C}{f^v + \Delta f^v} \tag{10}$$

The energy consumption generated by executing the task locally on the vehicle is

$$E_v = \kappa_v (f^v + \Delta f^v)^2 \theta_v Z_C \tag{11}$$

where Δf^v is the computation resources deviation value of the vehicle. κ_v is the energy coefficient depending on the vehicle chip structure [31].

Edge computing model

When the vehicle offloads the subtask to the edge server for processing, the time required includes (1) the upload time for transmitting the subtask to the edge server, (2) the processing task time of the edge server, and (3) the downlink time for the edge server to feed the task result back to the vehicle. The results of the task will be compressed and are much smaller than when they are input, so the downlink time can be ignored in the calculation [32].

Due to the heterogeneity of resources, the edge servers have different communication resources, resulting in different transmission times for vehicles offloading subtasks to different edge servers. When the vehicle offloads a subtask with a ratio of θ_j to the edge server ES_j, the transmission time is

$$T_j^1 = \frac{\theta_j D_D}{R_j} \tag{12}$$

R_j represents the data transmission rate of vehicle v to edge server ES_j, which can be calculated according to the Shannon's formula

$$R_j = (C_j^1 + \Delta C_j^1) \log_2 (1 + \frac{p^v h^v d_{vj}^{-\alpha}}{\sigma_0}) \tag{13}$$

where p^v is the transmitting power of the vehicle, h^v is the current channel gain, σ_0 is power of the Gaussian white noise, α is path loss exponent. $d_{vj} = \sqrt{\left(x^v - x_j^E\right)^2 + \left(y^v - y_j^E\right)^2}$ indicates the distance between the vehicle v and the edge server ES_j.

When the task ratio processed by the edge server ES_j is θ_j, the processing time is

$$T_j^2 = \frac{\theta_j Z_C}{C_j^2 + \Delta C_j^2} \tag{14}$$

Therefore, when the vehicle offloads a subtask with a ratio of θ_j to the edge server ES_j, the time includes transmission time and processing time, which can be calculated

$$T_j = T_j^1 + T_j^2 \tag{15}$$

The energy consumption generated by this process includes vehicle transmission subtask energy consumption and edge server ES_j processing energy consumption, which can be calculated

$$E_j = p^v \frac{\theta_j D_D}{R_j} + \kappa_E (C_j^2 + \Delta C_j^2)^2 \theta_j Z_C \tag{16}$$

where κ_E is the energy coefficient depending on the edge server chip structure.

With the help of advanced wireless communication technology, the vehicle cooperates with multiple edge servers to process tasks in parallel through full-duplex mode, so the completion time of a task is the maximum of the vehicular local computing time or the computing time of multiple edge servers. Therefore, the total time for the vehicle v offloading task is expressed as

$$T_{TO} = \max\{T_v, T_1, \ldots, T_j, \ldots, T_M\} \tag{17}$$

The total energy consumption of offloading task includes local computing and edge computing energy consumption, which can be calculated

$$E_{TO} = E_v + \sum_{j=1}^{M} E_j \tag{18}$$

By introducing a delay factor ϑ_1 and energy consumption factor ϑ_2, the optimization of the offloading delay and the energy consumption is transformed into the optimization of the offloading cost

$$U_{TO} = \vartheta_1 T_{TO} + \vartheta_2 E_{TO} \tag{19}$$

Blockchain consensus model

By dividing and offloading vehicular tasks to edge servers, the service quality and service experience of vehicles are improved, but there exist some malicious edge servers that may privately steal vehicle user data and even disclose the privacy of vehicle users. Therefore, to ensure the security of the vehicular edge network, we use blockchain technology to verify task transactions. Edge servers are regarded as blockchain network nodes to perform consensus because of their 3C resources. They will form a verification set, where one edge server acts as the leader and is responsible for block production and the other edge servers act as verifiers to audit and verify the blocks. The blockchain consensus phase can be divided into four processes: block production process, block broadcast process, block verification process, and block confirmation process.

During the block production process, the edge server collects transactions after the task is offloaded and broadcasts them to the entire network. The edge server, as a leader, packages its collection and computes a correct hash to create an unverified block. According to Delegated Proof of Stake (DPoS), the time T_{bp} for collecting transactions and computing hashes can be predefined, for example the time is 0.5s in a commercial distributed design blockchain operating system. In the meantime, the energy consumption during the block production process can be expressed as

$$E_{bp} = \kappa_E (C_j^3 + \Delta C_j^3)^3 T_{bp} \tag{20}$$

During the block broadcast process, the leader sends a broadcast message to the other verifier edge servers. The format of the broadcast message is $Brm = (block, PK_{sou}, PK_{des}, ts_{Brm})$, where *block* represents the block it created, PK_{sou} is the source address of the message, PK_{des} is the destination address of the message, and ts_{Brm} is the timestamp of the broadcast message. The leader simultaneously broadcasts the generated blocks to the other verifiers. Since the location of each edge server is different, the block broadcast time is determined by the block transmission time with the longest distance. The time and energy consumption of the block broadcast process are respectively

$$T_{bb} = \max_{j,m \in M} \{ \frac{D_{bb} d_{jm}}{R_E} \} \tag{21}$$

$$E_{bb} = \sum_{j=1}^{M-1} p^E \frac{D_{bb} d_{jm}}{R_E} \tag{22}$$

where D_{bb} is the size of the block before validation. d^{jm} is the distance between the edge server j and m. P^E is the transmit power of the edge server. R_E is the wired transmission rate between edge servers.

The block verification process consists of three stages. First, each verifier that receives a block performs a local verification, i.e., it audits the accuracy of the block by comparing the hash value calculated by using a hashing algorithm with the digital signature of the received broadcast message. Then, the verifier broadcasts its local audit result with the digital signature to other verifiers in a distributed manner. Finally, the verifier performs a secondary audit of the received local results. The time and energy consumption during the block verification process can be expressed as

$$\begin{aligned} T_{bv} &= \max_{m,m' \in M, m \neq m'} \{ T_{lv} + T_{rb} + T_{sa} \} \\ &= \max_{m,m' \in M, m \neq m'} \{ \frac{D_{bb} Z_0}{C_m^3 + \Delta C_m^3} + \frac{D_{lr} d_{mm'}}{R_E} + \frac{D_{lr} Z_0}{C_{m'}^3 + \Delta C_{m'}^3} \} \end{aligned} \tag{23}$$

$$\begin{aligned} E_{bv} = &\sum_{m=1}^{M-1} \kappa_E (C_m^3 + \Delta C_m^3)^2 D_{bb} Z_0 + \sum_{m=1}^{M-1} p^E \frac{D_{lr} d_{mm'}}{R_E} \\ &+ \sum_{m'=1}^{M-1} \kappa_E (C_{m'}^3 + \Delta C_{m'}^3)^2 D_{lr} Z_0 \end{aligned} \tag{24}$$

where T_{lv} is the block local verification time, T_{rb} is the broadcast time of the block local verification results, and T_{sa} is the secondary audit time of the block local verification results. Z_0 represents the size of caching resources per bit used to verify blocks, $C_m^3 + \Delta C_m^3$ and $C_{m'}^3 + \Delta C_{m'}^3$ represent the caching resources provided by the verifier edge servers m and m' for block verification respectively. $d_{mm'}$ is the distance between the verifier m and m'. D_{lr} is the data size of the local verification result.

During the block confirmation process, each verifier compares its audit results with those received from other verifiers and sends a confirmation message to the leader. The format of the validation message is $Com = (Res_{com}, Aut_{own}, Aut_{oth}, PK_{sou}, PK_{des}, ts_{Com})$, where Res_{com} is the audit comparison result between the verifiers, Aut_{own} is the verifier's audit result, Aut_{oth} is the audit result received from other verifiers, and ts_{Com} is the timestamp of the feedback message. The block confirmation time depends on the distance between edge servers, so the block confirmation time and energy consumption are expressed as

$$T_{bc} = \max_{j,m \in M} \{ \frac{D_{sr} d_{jm}}{R_E} \} \tag{25}$$

$$E_{bc} = \sum_{j=1}^{M-1} p^E \frac{D_{sr} d_{jm}}{R_E} \tag{26}$$

where D_{sr} is the data size of the secondary audit result.

Similarly, the optimization of consensus delay and the energy consumption is transformed into the optimization of consensus cost by introducing a delay factor ϑ_1 and energy consumption factor ϑ_2. The expression is

$$\begin{aligned} U_{BC} =& \vartheta_1 T_{BC} + \vartheta_2 E_{BC} \\ =& \vartheta_1\{T_{bp} + T_{bb} + T_{bv} + T_{bc}\} + \vartheta_2\{E_{bp} + E_{bb} + E_{bv} + E_{bc}\} \end{aligned} \tag{27}$$

In summary, network cost consists of task offloading cost and blockchain consensus cost, which can be calculated

$$U_{Net} = U_{TO} + U_{BC} \tag{28}$$

Problem formulation

The purpose of task offloading and blockchain consensus is to minimize network cost, including offloading cost and consensus cost. Therefore, the optimization model can be formulated as

$$\begin{aligned} & \min \quad U_{Net} \\ s.t. \quad & C1: \quad \theta_v + \sum_{j=1}^{M} \theta_j = 1, \quad \forall j \in M \\ & C2: \quad \vartheta_1 + \vartheta_2 = 1 \\ & C3: \quad f_v < C_j^2, \quad \forall j \in M \\ & C4: \quad T_{TO} \leq T_{max}, \quad \forall j \in M \end{aligned} \tag{29}$$

C1 is a requirement for task offloading ratios to ensure that tasks can be fully executed. C2 represents a linear combination of delay and energy consumption. C3 ensures that the computing power of each edge server is greater than the computing power of the vehicle. C4 indicates that the task offloading time cannot exceed the maximum deadline for the task.

Algorithms

In this section, we divide the optimization model into two problems to solve, namely, using an improved cuckoo algorithm to minimize task offloading cost, and using a greedy strategy to minimize blockchain consensus cost, thereby jointly optimizing to minimize network cost.

ICA based task offloading algorithm

The offloading ratio of a task determines its offloading time and energy consumption, i.e., task offloading cost. Since the advantages of Cuckoo algorithm (CA) include fewer parameters and fast convergence speed [33], we design a task offloading algorithm based on improved Cuckoo algorithm (ICA). The probability of cuckoo eggs being discovered P_a and control step ε_0 are adaptively changed to balance the relationship between global search ability and local search ability. At the same time, the preferred random walking mode with global optimal guidance can better improve the development ability of ICA algorithm and jump out of a local optimal solution, thereby obtaining the optimal task offloading ratio and minimizing task offloading cost. The task offloading algorithm based on ICA is shown in Algorithm 1.

```
Require: algorithm parameters, the parameters of task and edge servers
Ensure: global optimal nest location $X_{best}$, the optimal fitness value $U_{TO}^{min}$
 1: begin
 2: Use Latin hypercube sampling to generate the initial nest locations
 3: Calculate the fitness value of all nests by (19), and update initial optimal $X_{best}^{ini}$ and $U_{TO}$
 4: while $t \leq T_I$ do
 5:     Update dynamically nest location with improved Levy flights by (30), (34)
 6:     Compare with the previous generation, and update better $U_{TO}$ nest location
 7:     if $\rho_2 < P_a$ then
 8:         Search randomly nest locations by (36)
 9:         Update better nest location
10:     end if
11:     Update present optimal nest location and optimal solution
12:     $t = t + 1$
13: end while
14: return $X_{best}$, $U_{TO}^{min}$;
15: $end$
```

Algorithm 1 ICA based task offloading algorithm

To prevent uneven distribution of individuals within a randomly generated population, we adopt the Latin hypercube sampling method to generate the initial population [34]. Based on the space-filling technique, Latin hypercube sampling ensures that the projection of sample points in the variable space is uniformly distributed in each dimension. The population locations after Latin hypercube sampling not only realize non-overlap throughout the entire space-filling and sampling to make the initial population uniformly distributed but also allow more full searching of the entire variable space with a small number of sample points. The specific steps for population initialization in the ICA are as follows

(1) Determine the population size and dimension number of the ICA,
(2) Determine the interval of variables $X \in [0, 1]$,
(3) Divide the interval of variables $X \in [0, 1]$ into equal subintervals with the same number of population sizes,
(4) Select a point within each sub-interval of each dimension randomly,
(5) Combine the extracted points of each dimension to generate the initial population position of the ICA.

Based on the above three rules, the CA uses both the Levy flight mode and the random walking mode to jointly update the location of the nest.

Levy flight mode (global search). The update formula for the location and path of the cuckoo bird searching for its host nest is

$$X_i^{t+1} = X_i^t + \varepsilon \bigoplus L(\lambda), i = 1, 2, \ldots, N \tag{30}$$

where X_i^{t+1} and X_i^t represent the nest location in the $t + 1$ and t generation respectively. $\varepsilon = \varepsilon_0(X_i^t - X_{best})$ is the control step length, X_{best} represents the optimal nest location in the current population. $\bigotimes$ stands for point-to-point multiplication; N is the number of bird nests; $L(\lambda)$ represents the random search path for Levy flight, which follows the Levy probability distribution $L(\lambda) \sim s^{-\lambda}, 1 < \lambda \leq 3$, and its random step s can be calculated by

$$s = \frac{u}{|\mu|^{1/\beta}} \tag{31}$$

where $u \sim N(0, \delta_u^2)$, $\mu \sim N(0, 1)$, $\beta = 1.5$, δ_u can be calculated by

$$\delta_u = \{\frac{\Gamma(1/\beta)\sin(\pi\beta/2)}{\Gamma[(1+\beta)/2]\beta 2^{(\beta-1)/2}}\}^{1/\beta} \tag{32}$$

The control step ε_0 in the early stage of iterative optimization should be larger to search more regions to improve the global optimum performance, which is more favorable to jump out of the local optimum. In the later stage of iterative convergence, the control step ε_0 should be gradually reduced to a smaller value, which is more conducive to the careful search in the local region and to improve the solution accuracy while accelerating the convergence speed. As a result, the relationship between the control step ε_0 and the number of iterations t is

$$\varepsilon_0 = \varepsilon_0^{min} + (\varepsilon_0^{max} - \varepsilon_0^{min})(\frac{T_I - t}{T_I})^{\omega} \tag{33}$$

where the maximum and minimum values of control step ε_0 represent ε_0^{max} and ε_0^{min} respectively, T_I represents the maximum number of iterations, and ω is a non-linear factor with a value greater than 0 to control the rate of decline of control step ε_0.

Preferred random walking mode (local search). The host bird will discover and discard cuckoo eggs with a certain probability P_a, $P_a \in [0, 1]$. The new nest location is updated by

$$X_i^{t+1} = X_i^t + \rho_1 \bigoplus H(\rho_2 - P_a) \bigoplus (X_{i'}^t - X_{i''}^t) \tag{34}$$

where $i \neq i' \neq i''$, and $i, i', i'' \in [1, N]$. ρ_1 and ρ_2 are random numbers subject to uniform distribution [0, 1]; H is the Heaviside step function; $X_{i'}^t$ and $X_{i''}^t$ represent two randomly selected nest locations in the T generation respectively.

The discovery probability P_a of cuckoo eggs is used to generate new individuals. If $\rho_2 < P_a$, the original individuals are saved to the next generation. If $\rho_2 > P_a$, the original individuals are eliminated and new individuals are generated. However, in the early iterative stage, the discovery probability P_a should be kept at a large value to increase the diversity of the population and avoid falling into a local optimum, and a smaller value in the later stage to ensure better convergence of the algorithm. The appropriate discovery probability P_a should be gradually reduced as the iteration progresses to ensure that new individuals can be generated more easily in the late iterations, so we adopt a cosine decreasing strategy to achieve dynamic changes in the discovery probability:

$$P_a = P_a^{max}\cos(\frac{\pi}{2} \cdot \frac{t-1}{T_I + 1}) + P_a^{min} \tag{35}$$

To greatly improve the population diversity and the efficiency of ICA algorithm, we combine the particle swarm algorithm (PSO) and the golden sine optimization algorithm [35] for the preferred random walking mode with global optimal guidance, which can be calculated by

$$\begin{aligned} X_i^{t+1} = X_i^t|\sin\varphi_1| + \rho_1 \bigoplus H(\rho_2 - P_a) \bigoplus \varphi_2 \sin\varphi_1 |\phi_1 X_{i'}^t - \phi_2 X_{i''}^t| \\ + \zeta(X_{best} - X_{i'''}^t) \end{aligned} \tag{36}$$

where $i \neq i' \neq i''$, and $i, i', i'' \in [1, N]$. $\varphi_1 \in [0, 2\pi]$ and $\varphi_1 \in [0, \pi]$ are random numbers that determine the distance and direction of the location update during the iteration respectively. $\phi_1 = -\pi + 2\pi(1-\gamma)$ and $\phi_1 = -\pi + 2\pi\gamma$ are compression factors that can be used to compress the space to find the best nest location more quickly. $\gamma = (\sqrt{5} - 1)/2$ is the golden section ratio coefficient, which can be obtained

$$\zeta = \eta\gamma \tag{37}$$

where the offset coefficient η is a random number uniformly distributed on the interval $(-\eta, \eta)$, and the value is $(\pi/6)$ [36]. According to (36), the preferred random walking mode with global optimal guidance generates new solutions based on the golden ratio coefficient close to the global optimal solution, because the new solutions obtained by the golden ratio coefficient are superior to those generated only within the interval [0, 1]. Compared to generating new solutions linearly between [0, 1], the offset coefficient allows the ICA to search for more location information with a certain offset angle.

Greedy strategy based resource allocation scheme

DPoS is a fast and efficient blockchain consensus mechanism that utilizes voting and selection to protect blockchains from the impact of centralization and malicious use [37]. Compared with traditional consensus algorithms, the number of entities involved in DPoS consensus is few, which can effectively reduce the time and energy consumption to reach consensus. Therefore, the DPoS consensus is combined with a greedy strategy to reasonably allocate resources, thereby minimizing the consensus cost.

The first is the leader selection. Task offloading is mainly performed between edge servers, so they store relevant transaction information and can be considered as leader candidates. In each selection, the edge servers vote based on their respective caching resources. The caching resources can be used for block consensus, including block production and verification. As a measure of the quality of the edge servers, more caching resources means that the edge servers have more processing power to produce and verify blocks. At the end of the voting, the edge servers with the highest number of votes will form a verification set.

```
Require: block parameters, the parameters of edge servers
Ensure: validation set
 1: begin
 2: Rank the edge servers according to $C^3$
 3: Select $ES_j$ with the largest $C^3$ as the leader
 4: for $i = 1 : M - 1$ do
 5:     Calculate the distance between $ES_j$ and other verifiers
 6:     Retrieve the edge servers with the shortest distance from $ES_j$
 7: end for
 8: while new $ES_m$ add do
 9:     Repeat step 1  6
10:     if $d_{jj'} > d_{jm}$, $C^3_{j'} < C^3_m$ then
11:         $d_{jj'} \Rightarrow d_{jm}$, $C^3_{j'} \Rightarrow C^3_m$
12:     end if
13: end while
14: return validation set;
15: $end$
```

Algorithm 2 Greedy strategy based resource allocation scheme

There are two roles in the verification set, the leader and the verifier. The leader is responsible for transaction collection and block production, and the verifier is responsible for block verification. In each block production process, one edge server in the verification set acts as a leader and other edge servers act as verifiers. Leaders are generated in a round-robin manner in the verification set, indicating that each edge server can become a leader to produce blocks. In the specific block consensus process, the leader first collects a certain amount of task-offloading transactions and then computes a correct hash value to produce an unverified block. The verifier audits the block broadcasted by the leader and sends a confirmation message to the leader. The leader analyzes the received confirmation message and audits the block for accuracy. If more than two-thirds of the verifiers agree on the block, the leader sends the block to all edge servers in the verification set for storage. Edge servers that are not in the verification set will periodically synchronize the latest blockchain information in their vicinity.

When a newly generated block is successfully added to the blockchain, the edge servers involved in the block consensus process are rewarded to compensate for their resource consumption. If all edge servers in the verification set become leaders, the order of leaders is changed, and they then produce new blocks again in a round-robin manner. If a leader fails to create a block, the block is skipped and the transactions in the skipped block are transferred to the next block. The blockchain resource allocation scheme based on greedy strategy is shown in Algorithm 2.

Performance evaluation

In this section, simulation experiments are conducted to evaluate the performance of our proposed scheme. The comparison schemes in this paper include the cuckoo algorithm scheme (CA), distribute maximum scheme (DM), average offloading scheme (AVER), local computing scheme (LOCA), and random offloading scheme (RAN).

Simulation setup

We considered an IoV scenario where vehicles drive randomly on the road and edge servers are randomly distributed on both sides of the road. The transmission power of each edge server is 10 W, and the transmission rate between edge servers is 100 Mbit/s. We assume that vehicular computing power is 19 GHz and transmission power is 0.3 W. The current channel gain is 60 dB, the power of the Gaussian white noise is 2×10^{-12} W, and the path loss exponent is 3. The energy coefficient of the vehicle chip structure is 10^{-26}, while the energy coefficient of the edge server chip structure is 10^{-23}. For the convenience of analysis, the DT deviation value is set to 1. The block size is [0, 30] MB, the data size of the local audit result is [0, 3] MB and the data size of the secondary audit result is [0, 300] KB.

In addition, the number of iterations is 100. The maximum and minimum values of discovery probability P_a are 0.3 and 0.1, respectively. The maximum and minimum values of control step ε_0 are 1.5 and 1, respectively. The non-linear factor ω is taken as 3. Other parameters in the simulation are shown in Table 2.

All simulations are performed by MATLAB R2016a on a computer configured with 8GB RAM and Intel Core i5-9500 3.0 GHz CPU with a 64-bits Microsoft Windows 10 operating system. Besides, in order to eliminate the error caused by randomness, each simulation test is run 20 times to obtain an average result.

Table 2 Simulation parameters

Parameter	Value
The data size of the task D_D	[0.2,1] MB
The CPU cycles of the task Z_C	$[10,20] \times 10^8$
The communication resources of edge server C_j^1	[20,30] GHz
The computation resources of edge server C_j^2	[100,110] GHz
The storage resources of edge server C_j^3	[5,10] GHz

Security analysis

The DT-based blockchain provides secure and reliable edge cooperation for vehicular edge networks.

1) Efficient operation: Physical entities transmit information through physical-to-physical (P2P) communications, including vehicles and edge servers. The DTs in the proposed DTVEN achieve information interaction and resource sharing through twin-to-twin (T2T) communications, to reflect the state behavior in the actual physical network. T2T communications depend on the computation resources of the edge server to model and simulate, without consuming any communication resources. Compared with time-consuming P2P communications, T2T communications only need to simulate and predict the operational state of the network in a relatively short time, and send the results to the physical entities through twin-to-physical (T2P) communications to realize task offloading and resource allocation.

2) Transaction traceability: In the proposed scheme, the verifier selection process is based on a greedy strategy, where all blocks and transactions are publicly audited and mutually verified by all verifiers in a round-robin manner. Broadcasted transactions recorded with timestamps cannot be maliciously modified by a single entity, since timestamps in the blockchain can be used to keep transactions intact. At the same time, any vehicle and edge server can quickly verify and trace previous records by accessing the digital twin layer.

3) Privacy Protection: Vehicles, edge servers, and their corresponding DTs transfer messages about task offloading and blockchain in an anonymous manner. For example, the twin DT^v of a vehicle v uses its public key as a pseudonym to ensure the anonymity of its true identity. DT^v sends the messages (i.e., $Boc^{DT^v}, Sol^{DT^v \to DT_j^E}, Feb^{DT^v \to v}$), and transactions (i.e., $Tra^{DT^v \to DT_j^E}$), which are signed and can only be accessible by vehicle v and the digital twin DT_j^E of the edge server ES_j with the correct private key. If the malicious $DT^{v'}$ and DT^E want to forge the signature of DT^v to pass the authentication process, the attackers must forge a signature. However, attackers cannot obtain the private key from the public key of DT^v, so they cannot access the private key to forge the signature information of the legitimate DT^v. In edge collaboration, anonymity and digital signatures can protect the privacy of physical entities and twins in DTVEN.

Simulation and results analysis

As can be seen from Fig. 4, the ICA proposed in this paper can achieve effective convergence compared to the traditional CA. The solution space can be explored more fully by generating uniformly distributed initial populations through Latin hypercube sampling. In the global search process using Levy flight, the number of control step ε_0 changes dynamically with the number of iterations to improve the accuracy of the search. In the process of local search, the cosine decreasing strategy is used to realize the dynamic change of discovery probability P_a, which can increase the strength of population evolution and thus avoid falling into local optimum. At the same time, the global optimization guidance is added to strengthen the development ability of the algorithm, and the search process generates new solutions toward the global optimum to improve the convergence speed.

Figure 5 illustrates the comparison of offloading time with increasing data size for different schemes. It can be seen that offloading time increases with increasing data size for most of the schemes. The time fluctuation of the AVER scheme is the smallest because offloading time depends on the maximum of both local computing time and edge computing time when the task is partially offloaded. Since the computing power of the edge server is stronger than that of the vehicle, vehicular local

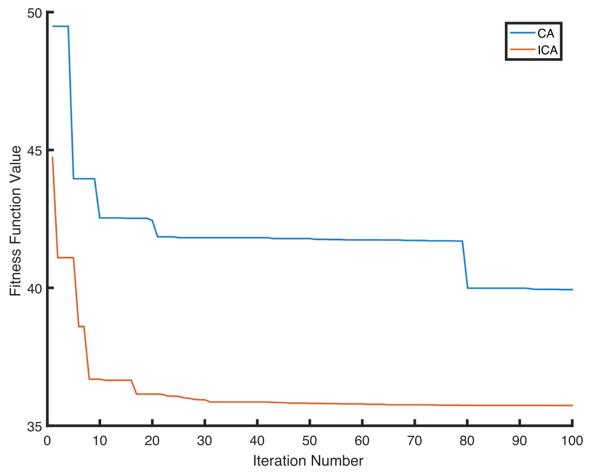

Fig. 4 Comparison of convergence performance

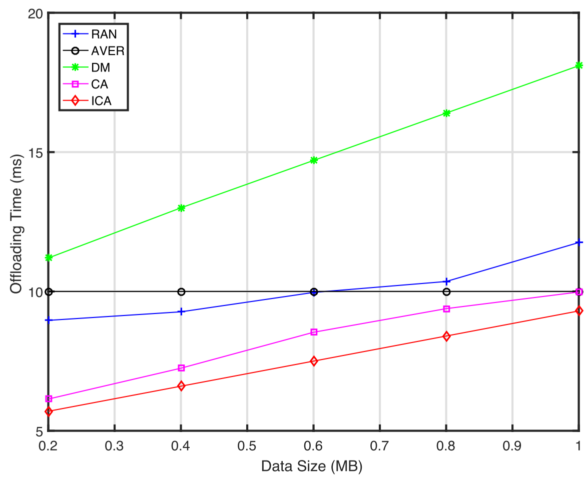

Fig. 5 Offloading time with respect to data size under different schemes

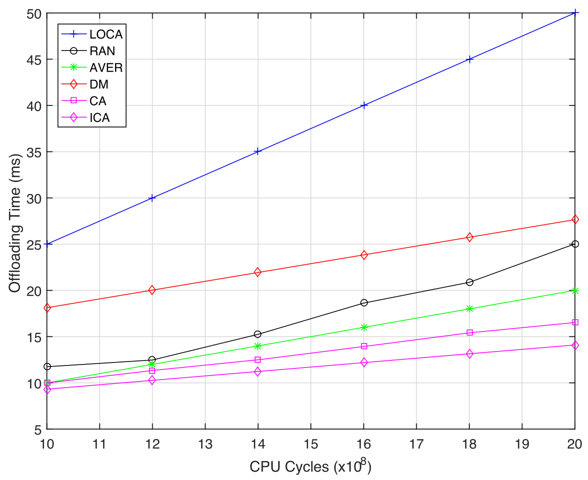

Fig. 6 Offloading time with respect to CPU cycles under different schemes

computing time is often higher than the time offloaded to the edge server for task execution, so the offloading time of the AVER scheme is vehicular local computing time. The DM scheme represents offloading a single task to the edge server with the maximum computing power without any task segmentation operation. Even if the task is offloaded to the edge server with the maximum computing power, the transmission time of the task grows due to the increase in the data size, so the offloading time of the DM scheme also increases linearly. The RAN scheme randomly offloads tasks to the edge server, and the offloading ratio of the tasks is random, resulting in a large offloading time. The comparison results show that the ICA scheme proposed in this paper consistently outperforms the other schemes in terms of offloading time. Compared with other schemes, the ICA scheme can more reasonably allocate communication and computation resources between the vehicles and the edge servers to efficiently execute the tasks.

Figure 6 compares the offloading time for task execution under six schemes. As the number of CPU cycles required for computing tasks increases, the offloading time also grows. The time spent in the LOCA scheme is the largest since the task is executed directly on the vehicle without any offloading operation. Due to the limited computation resources of the vehicle, the LOCA scheme is prone to the risk of vehicular overload. The task is only offloaded to the edge server with the highest computation resources in the DM scheme. Although the task offloading time also increases, it is less than the LOCA scheme because the edge server has higher computing power than that of the vehicle. Compared with the average task offloading in the AVER scheme and the random task offloading in the RAN scheme, the ICA-based task offloading scheme maintains good performance. Considering the resource constraints of the edge server, the appropriate subtasks are offloaded based on the distance from the edge server, thereby reducing the offloading time.

Figure 7 shows how the offloading cost varies with the CPU cycles. It can be seen that when the number of CPU cycles required to process the task increases, the corresponding offload cost grows as well. The offloading cost of the LOCA scheme is the largest since the offloading time required by the LOCA scheme is the largest and computing tasks locally consume vehicular computation resources. The RAN scheme randomly divides and offloads tasks to different edge servers, resulting in relatively high offloading cost. In Fig. 6, although the offloading time of the DM

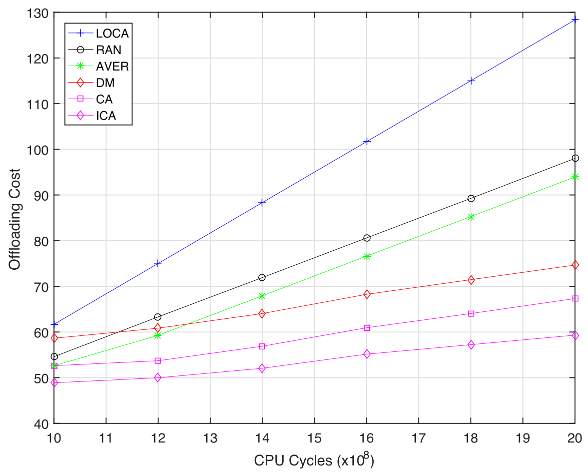

Fig. 7 Offloading cost with respect to CPU cycles under different schemes

scheme is larger than that of the AVER scheme, its offloading cost is lower than that of the AVER scheme. This is because the DM scheme offloads a single task to the edge server with the strongest computing power, and the offloading time increases with the number of CPU cycles, but it only consumes the computation resources of one edge server. In contrast, the AVER scheme requires dividing and offloading tasks equally to the vehicle and the edge server, which consumes not only the computation resources of the vehicle, but also the communication and computation resources of the edge server. Our proposed ICA scheme can measure the energy consumption of offloading tasks based on the communication and computation resources between the vehicle and the edge servers, and reasonably divide the offloading tasks to minimize the offloading cost.

Figure 8 presents the network cost over the varying CPU cycles under six schemes. The network cost of all schemes grows correspondingly with the increase in the number of CPU cycles. During the task offloading process, the task computing time and energy consumption increase accordingly as the number of CPU cycles for processing the task grows. Furthermore, as the CPU cycle of a task increases, more transactions are generated, resulting in a larger number of generated blocks. This increases the time and energy consumption of the blockchain consensus process, and the network cost also grows. We adopt a resource allocation scheme based on greedy strategy, which reduces the blockchain consensus cost by selecting the edge server with the highest caching resources as the leader and reducing the distance of block broadcast. From Fig. 7, it can be seen that our proposed ICA scheme can obtain the minimum offloading cost. Therefore, the proposed scheme in this paper can achieve the minimum network cost during the entire task offloading process.

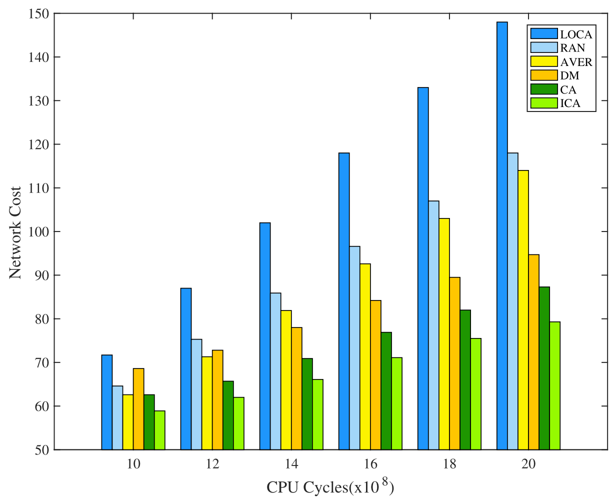

Fig. 8 Comparison of network cost with respect to CPU cycles under different schemes

Conclusion

In this paper, an edge cooperation scheme based on task offloading and resource allocation is proposed to minimize the network cost. We propose a novel blockchain-empowered digital twin vehicular edge network architecture, which utilizes the DT-based smart contract to realize efficient and secure edge computing. To jointly optimize the latency and energy consumption, we first design an ICA-based task offloading algorithm to reduce offloading cost by reasonably offloading subtasks to edge servers. Then, while introducing blockchain for safe data sharing, a greedy strategy-based resource allocation scheme is present to select appropriate edge server verifiers, thereby diminishing consensus cost. Simulation results show that the proposed edge cooperation scheme has a lower network cost significantly than other schemes during the entire task offloading process.

Acknowledgements
We would like to express our sincere thanks to the editors and reviewers for their insightful comments and suggestions on improving this paper.

Authors' contributions
Chunhai Li, Qiyong Chen designed the research and wrote the paper. Mingfeng Chen, Zhaoyu Su analyzed the proposed research. Yong Ding, Dapeng Lan and Amir Taherkordi made suggestions on research methodology and experimental evaluation. All authors reviewed and edited the manuscript.

Funding
This research was supported in part by the Guangxi Natural Science Foundation under Grant 2023GXNSFAA026294, in part by the National Natural Science Foundation of China under Grants 61871466.

Availability of data and materials
Not applicable.

Declarations

Ethics approval and consent to participate
This article does not contain any studies with human participants or animals performed by any of the authors.

Competing interests
The authors declare no competing interests.

Received: 30 March 2023 Accepted: 28 July 2023
Published online: 11 August 2023

References

1. Guo Z, Yu K, Bashir AK, Zhang D, Al-Otaibi YD, Guizani M (2022) Deep information fusion-driven poi scheduling for mobile social networks. IEEE Netw 36(4):210–216
2. Li Y, Ma H, Wang L, Mao S, Wang G (2020) Optimized content caching and user association for edge computing in densely deployed heterogeneous networks. IEEE Trans Mob Comput 21(6):2130–2142
3. Zhu L, Yu FR, Wang Y, Ning B, Tang T (2018) Big data analytics in intelligent transportation systems: A survey. IEEE Trans Intell Transp Syst 20(1):383–398

4. Zhang Q, Yu K, Guo Z, Garg S, Rodrigues JJ, Hassan MM, Guizani M (2021) Graph neural network-driven traffic forecasting for the connected internet of vehicles. IEEE Trans Netw Sci Eng 9(5):3015–3027
5. Liu Y, Peng M, Shou G, Chen Y, Chen S (2020) Toward edge intelligence: Multiaccess edge computing for 5g and internet of things. IEEE Internet Things J 7(8):6722–6747
6. Gong C, Lin F, Gong X, Lu Y (2020) Intelligent cooperative edge computing in internet of things. IEEE Internet J 7(10):9372–9382
7. Liu L, Feng J, Mu X, Pei Q, Lan D, Xiao M (2023) Asynchronous deep reinforcement learning for collaborative task computing and on-demand resource allocation in vehicular edge computing. IEEE Trans Intell Transp Syst 1–14. https://doi.org/10.1109/TITS.2023.3249745
8. Wu Y, Zhang K, Zhang Y (2021) Digital twin networks: A survey. IEEE Internet Things J 8(18):13789–13804
9. Minerva R, Lee GM, Crespi N (2020) Digital twin in the iot context: A survey on technical features, scenarios, and architectural models. Proc IEEE 108(10):1785–1824
10. Zhao L, Han G, Li Z, Shu L (2020) Intelligent digital twin-based software-defined vehicular networks. IEEE Netw 34(5):178–184
11. Fu Y, Li C, Yu FR, Luan TH, Zhao P, Liu S (2022) A survey of blockchain and intelligent networking for the metaverse. IEEE Internet Things J 10(4):3587–3610
12. Sharma V (2018) An energy-efficient transaction model for the blockchain-enabled internet of vehicles (iov). IEEE Commun Lett 23(2):246–249
13. Du J, Cheng W, Lu G, Cao H, Chu X, Zhang Z, Wang J (2022) Resource pricing and allocation in mec enabled blockchain systems: An a3c deep reinforcement learning approach. IEEE Trans Netw Sci Eng 9(1):33–44. https://doi.org/10.1109/TNSE.2021.3068340
14. Zhang P, Zhou M, Zhao Q, Abusorrah A, Bamasag OO (2021) A performance-optimized consensus mechanism for consortium blockchains consisting of trust-varying nodes. IEEE Trans Netw Sci Eng 8(3):2147–2159
15. Lu Z, Liu W, Wang Q, Qu G, Liu Z (2018) A privacy-preserving trust model based on blockchain for vanets. IEEE Access 6:45655–45664
16. Jian X, Leng P, Wang Y, Alrashoud M, Hossain MS (2021) Blockchain-empowered trusted networking for unmanned aerial vehicles in the b5g era. IEEE Netw 35(1):72–77
17. Fu Y, Yu FR, Li C, Luan TH, Zhang Y (2020) Vehicular blockchain-based collective learning for connected and autonomous vehicles. IEEE Wirel Commun 27(2):197–203
18. Ma C, Zhu J, Liu M, Zhao H, Liu N, Zou X (2021) Parking edge computing: parked-vehicle-assisted task offloading for urban vanets. IEEE Internet Things J 8(11):9344–9358
19. Fan W, Hua M, Zhang Y, Su Y, Li X, Wu F, Liu Y (2023) Game-based task offloading and resource allocation for vehicular edge computing with edge-edge cooperation. IEEE Trans Veh Technol 72(6):7857–7870
20. Duan W, Gu X, Wen M, Ji Y, Ge J, Zhang G (2021) Resource management for intelligent vehicular edge computing networks. IEEE Trans Intell Transp Syst 23(7):9797–9808
21. Zhang D, Yu FR, Yang R (2021) Blockchain-based multi-access edge computing for future vehicular networks: A deep compressed neural network approach. IEEE Trans Intell Transp Syst 23(8):12161–12175
22. Xu X, Shen B, Ding S, Srivastava G, Bilal M, Khosravi MR, Menon VG, Jan MA, Wang M (2020) Service offloading with deep q-network for digital twinning-empowered internet of vehicles in edge computing. IEEE Trans Ind Inf 18(2):1414–1423
23. Zhang K, Cao J, Zhang Y (2021) Adaptive digital twin and multiagent deep reinforcement learning for vehicular edge computing and networks. IEEE Trans Ind Inf 18(2):1405–1413
24. Dai Y, Zhang Y (2022) Adaptive digital twin for vehicular edge computing and networks. J Commun Inf Netw 7(1):48–59
25. Gong Y, Wei Y, Feng Z, Yu FR, Zhang Y (2022) Resource allocation for integrated sensing and communication in digital twin enabled internet of vehicles. IEEE Trans Veh Technol 72(4):4510–4524
26. Liu T, Tang L, Wang W, He X, Chen Q, Zeng X, Jiang H (2022) Resource allocation in dt-assisted internet of vehicles via edge intelligent cooperation. IEEE Internet Things J 9(18):17608–17626
27. Camenisch J, Hohenberger S, Pedersen MØ (2007) Batch verification of short signatures. In: Advances in Cryptology-EUROCRYPT 2007: 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007. Proceedings 26. Berlin, Springer, pp 246–263
28. Mach P, Becvar Z (2017) Mobile edge computing: A survey on architecture and computation offloading. IEEE Commun Surv Tutorials 19(3):1628–1656
29. Mao Y, You C, Zhang J, Huang K, Letaief KB (2017) A survey on mobile edge computing: The communication perspective. IEEE Commun Surv Tutorials 19(4):2322–2358
30. Liu Q, Luo R, Liang H, Liu Q (2023) Energy-efficient joint computation offloading and resource allocation strategy for isac-aided 6g v2x networks. IEEE Trans Green Commun Netw 7(1):413–423
31. Li C, Wang S, Huang X, Li X, Yu R, Zhao F (2018) Parked vehicular computing for energy-efficient internet of vehicles: A contract theoretic approach. IEEE Internet Things J 6(4):6079–6088
32. Yang Y, Liu Z, Liu Z, Chan KY, Guan X et al (2022) Joint optimization of edge computing resource pricing and wireless caching for blockchain-driven networks. IEEE Trans Veh Technol 71(6):6661–6670
33. Yang XS, Deb S (2009) Cuckoo search via lévy flights. In: 2009 World congress on nature & biologically inspired computing (NaBIC). Piscataway, IEEE, pp 210–214
34. Stein M (1987) Large sample properties of simulations using latin hypercube sampling. Technometrics 29(2):143–151
35. Tanyildizi E, Demir G (2017) Golden sine algorithm: A novel math-inspired algorithm. Adv Electr Comput Eng 17(2):71–78
36. Walton S, Hassan O, Morgan K, Brown M (2011) Modified cuckoo search: a new gradient free optimisation algorithm. Chaos Solitons Fractals 44(9):710–718
37. Xu G, Liu Y, Khan PW (2019) Improvement of the dpos consensus mechanism in blockchain based on vague sets. IEEE Trans Ind Inf 16(6):4252–4259

Publisher's Note