A Novel Energy-Minimization Joint Hungarian-PSO Algorithm for Task Offloading in Vehicular Fog Computing

Lin Chai, Jun Wang, Shuhui Fang, Yumei Yang, and Wu Wang
School of Mathematics and Computer Science, Yunnan Minzu University, Kunming 650504, China

With the rapid development of Internet of Things (IoT) and Artificial Intelligence (AI) technologies, Vehicular Fog Computing (VFC) has emerged as a crucial technology in Intelligent Transportation Systems (ITS). Fog computing is a distributed computing model. VFC is a network architecture formed by applying fog computing technology to the Internet of Vehicles (IoV). In VFC, vehicles can offload tasks to other vehicles or roadside units with more powerful resources, leveraging their idle computing power to boost system performance. As vehicles usually run on batteries, reducing energy consumption can extend battery life, support more computing tasks, and cut down device maintenance and replacement frequency. Therefore, how to efficiently perform task offloading to minimize energy consumption has become a key research challenge. This paper proposes a joint task offloading scheme based on the Hungarian algorithm and Particle Swarm Optimization (PSO) algorithm to optimize energy consumption and latency in fog computing networks. First, the Hungarian algorithm is employed to achieve optimal matching between User Vehicles (UVs) and Fog Vehicles (FVs), ensuring efficient task allocation. Subsequently, the PSO algorithm is utilized to optimize transmission power, further reducing energy consumption and interference. Experimental results demonstrate that the proposed scheme significantly reduces the system's total energy consumption while meeting task latency constraints. Simulation results also show that the scheme exhibits excellent performance and stability in complex network environments, providing an effective solution for task offloading in fog computing networks.

Index Terms—**Vehicular fog computing, Energy-Minimization, Task offloading, Hungarian algorithm, Particle Swarm Optimization algorithm.**

I. Introduction

With the advancement of autonomous driving, Intelligent Transportation Systems (ITS), and 5G/6G communication technologies, vehicle-generated tasks often have high computational density and strict latency constraints. For instance, real-time perception and decision-making tasks in autonomous driving must be completed within an extremely short time frame. Traditional cloud computing, due to its centralized architecture, suffers from high latency and limited bandwidth, making it difficult to meet real-time requirements.To address this issue, Vehicular Fog Computing (VFC) has emerged as a solution.

VFC is a fusion of IoV and Fog Computing, leveraging the computing and storage resources of edge nodes such as vehicles, Road Side Units (RSUs), and base stations [1]. In VFC, task offloading refers to transferring computation tasks from a vehicle to other vehicles or RSUs through wireless communication networks for processing. This compensates for the limitations of onboard computational resources and improves task execution efficiency and quality. Onboard devices often have restricted computing, storage, and battery power, making it hard to handle complex tasks. In contrast, remote computational nodes generally possess stronger computing and storage resources, enabling faster and more efficient task processing to improve execution speed and quality. By offloading tasks to better-resourced vehicles or RSUs, their idle computing power is utilized, boosting system performance.

Vehicles typically rely on battery power, and complex tasks can rapidly drain battery reserves, resulting in interruptions to critical functionalities, such as autonomous driving. Task offloading reduces the computional and storage burden on onboard devices, thereby lowering energy consumption and extending battery life. This allows vehicles to support computing-intensive tasks for longer without frequent charging. By optimizing energy consumption through task offloading, continuous operational uptime is prolonged, ensuring service stability. Simultaneously, this approach reduces maintenance and replacement frequency of devices, further lowering operational costs. Additionally, with the global focus on sustainable development, optimizing energy consumption during task offloading improves energy efficiency. This contributes to reducing carbon emissions, environmental pollution, and negative ecological impacts, aligning with environmental protection requirements. Therefore, reducing energy consumption is critically imperative.

Numerous studies have been conducted on the energy consumption issue in VFC. Liang et al. in [2] proposed a two-stage hybrid heuristic energy consumption optimization algorithm, employing the stable matching algorithm for fog vehicle selection and introducing a heuristic algorithm based on decision tree technology to minimize energy consumption and network delay. Zhang et al. designed an optimization algorithm based on a fair scheduling index, applying convex optimization and heuristic algorithms to address task offloading in fog computing networks, effectively lowering offloading energy consumption and enhancing network efficiency, but without prioritizing and allocating resources based on task urgency [3]. Kim et al. used a joint optimization model based on task popularity and convex optimization methods to tackle the energy consumption issue of user equipment and fog servers in fog computing networks [4]. Lin et al. employed convex

Manuscript received May 30, 2025; revised August 26, 2025. Corresponding author: Wu Wang (email: wangwu@ymu.edu.cn).

optimization, successive convex approximation—the primal interior-point method, and coalition game methods to resolve energy consumption issues in NOMA-assisted multi-fog-node VFC networks [5]. Yadav et al. proposed a new latency-sensitive and energy-efficient task offloading scheme based on clustering and particle swarm optimization, achieving joint minimization of latency and energy consumption [6]. Hussain et al. proposed a multi-objective optimization framework for VFC: it models the delay and energy consumption of the vehicle–fog–cloud hierarchy using probability and queuing theory, formulates the task-offloading decision as an MINLP, and solves it with a Modified Differential Evolution [7]. Targeting the facility-location problem in VFC, Hussain et al. proposed a multi-objective optimization model that simultaneously minimizes service delay and energy consumption, and designs a hybrid evolutionary algorithm called SONG (blending NSGA-II with SMPSO) to provide an efficient, low-latency, and low-energy deployment plan for VFC planning [8]. These studies have made significant contributions to reducing energy consumption. However, as vehicle-generated tasks may have varying latency requirements, it is necessary to consider these requirements and optimize them using particle swarm optimization.

The Particle Swarm Optimization (PSO) algorithm is a swarm intelligence-based optimization algorithm inspired by collective biological behaviors in nature, such as bird flock foraging and fish school swimming. It simulates the movement and information-sharing of particles in multi-dimensional space to optimize target functions and is widely used in function optimization, neural network training, combinatorial optimization, etc. The Hungarian algorithm is a combinatorial optimization algorithm for solving assignment problems, primarily used to find the minimum weight matching in bipartite graphs and widely applied in task and resource allocation. The PSO algorithm has many advantages in solving optimization problems, such as strong global search ability, fast convergence speed, and high adaptability. Therefore, this paper considers using the Hungarian algorithm and the PSO algorithm to solve the problem of reducing energy consumption.

The contributions of this study can be summarized as follows:

- This paper proposes a stepwise collaborative optimization framework. This framework decomposes the NP-hard Mixed-Integer Nonlinear Programming (MINLP) problem into UV-FV matching optimization (sub-problem P1) and transmission power allocation (sub-problem P2), which are then efficiently solved using the Hungarian algorithm and the Particle Swarm Optimization algorithm, respectively. By performing stepwise optimization to reduce computational complexity, it lowers the system's energy consumption while ensuring task latency constraints.
- This paper presents a normalized weighted model combining channel gain and FVs' computational capability to define the UVs' preference for FVs, achieving global-optimal task-fog node matching by considering communication quality and computational power.
- Rigorous theoretical analysis and extensive simulations confirm the marked superiority of the proposed joint framework over conventional methods. It guarantees globally optimal resource allocation and, across diverse task loads and network scales, consistently delivers lower total system energy consumption and shorter task-processing delays.

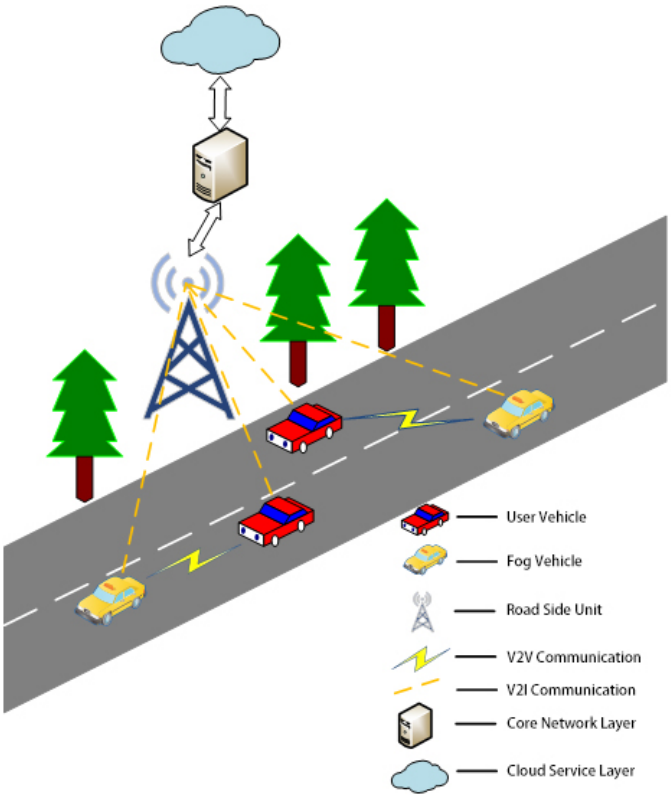

Fig. 1. The Architecture of VFC.

The remainder of this paper is organized as follows. Section II introduces the system model and formulates the energy minimization problem. Section III presents the optimization algorithms used in this paper. Section IV provides simulation results and compares them with other methods. Section V concludes the paper.

II. System Model and Problem Formulation

A. Task Offloading Model

VFC is a network architecture formed by integrating fog computing technology into the IoV, aiming to provide vehicles with low-latency, high-bandwidth, and real-time computing services by deploying computing resources at the network edge [9]. It meets the needs of delay-sensitive and computationally intensive applications in vehicular networks. Fig. 1 presents a VFC architecture, which consists of four layers: the terminal layer, fog node layer, core network layer and cloud service layer. In this architecture, user vehicles serve as the terminal layer, while fog vehicles and road side units act as the fog node layer. There are various communication modes in VFC, such as Vehicle-to-Vehicle (V2V) communication and Vehicle-to-Infrastructure (V2I) communication.

Due to the distance between UVs and data centers, VFC emerges as a promising task offloading paradigm that effectively utilizes the idle computational resources of vehicles. Vehicles equipped with dedicated short-range communication devices, high-speed computers, and Long-Term Evolution (LTE) communication equipment are referred to as Fog vehicles (FVs) [10]. These FVs can be managed directly by RSUs and provide task offloading services to UVs when they are idle.

In the task offloading scenario, when UVs generate computational tasks, they can choose to process the tasks locally or offload them to FVs or cloud servers via the uplink. By leveraging Cellular Vehicle-to-Everything (CV2X) technology, UVs and FVs can establish V2V connections when they are in close proximity. UVs request fog computing services and obtain available radio resources to offload tasks to FVs. Before offloading tasks, UVs need to select FVs within their coverage area to establish a connection. We assume that limited radio resources, referred to as Physical Resource Blocks (PRBs), can be shared by multiple UVs. UVs can utilize PRBs to offload their tasks. PRBs are the smallest units of resource allocation in the network, representing a 0.5 ms time slot and a 180 kHz bandwidth [11].

B. Communication Model

Consider a task offloading model comprising the set $\mathbf{K}' = \{1,\ldots,\mathrm{K}\}$ representing K FVs, the set $\mathbf{N}' = \{1,\ldots,\mathrm{N}\}$ representing N PRBs, and the set $\mathbf{S}' = \{1,\ldots,\mathrm{S}\}$ representing S UVs. A binary variable X_s^k indicates whether FV k is connected to UV s, with 1 representing a connection and 0 representing no connection. We assume that within a time slot, each UV can only select one FV for task offloading, and each FV can only serve one UV, meaning they are in a one-to-one correspondence.

The various tasks offloaded by UVs may have different Quality of Service (QoS) requirements, which depend on factors such as latency, the size of the requested data, and computational density. These requirements are outlined in Table I [12]. Specifically, tasks are divided into three specific service classes based on delay requirements [13].

Given the limited availability of PRBs, different UVs may reuse the same PRB for transmission, potentially causing interference. To mitigate this, each PRB is assigned to only one UV.

The transmission rate for a UV-FV connection can be measured using the Shannon-Hartley theorem, which calculates the achievable data rate for UV s connected to FV k [14]. The data rate $R_{s,k}$ is calculated as follows:

$$R_{s,k} = B\log_2\left(1+\frac{G_{s,k}^n P_s^n}{\sigma^2}\right). \tag{1}$$

Here, P_s^n is the transmission power of UV s on PRB n, $G_{s,k}^n$ is the link gain between UV s and FV k on PRB n, σ^2 is the noise power, B is the bandwidth corresponding to the PRB.

Optimizing the power allocation for UVs can enhance the data rate. In the VFC scenario, the latency for task offloading, which includes both communication and computation delays, must meet the latency constraints. For simplicity, we consider single-hop transmission for the tasks. Consequently, the communication delay $\tau_{s,k}^{\mathrm{com}}$ can be expressed as:

$$\tau_{s,k}^{\mathrm{com}} = \frac{\Psi_s}{R_{s,k}}, \tag{2}$$

where Ψ_s is the data size of the task sent by UV s.

From an energy-saving perspective, we account for both the communication energy consumption from UV to FV and the computational energy consumption at the FV. The communication energy consumption from UV s to FV k is calculated as follows:

$$E_{s,k}^{\mathrm{com}} = \tau_{s,k}^{\mathrm{com}} X_s^k P_s^n. \tag{3}$$

C. Fog Computing Model

In VFC, FVs are responsible for receiving and processing tasks sent by UVs. However, due to inherent differences in computational capabilities, FVs may vary in processing speed. Therefore, optimizing the joint transmission power and computational resource allocation is essential when selecting suitable FVs for task offloading. As mentioned earlier, we assume that an FV can only process one task at a time. To measure the processing speed of an FV, we can define the computational delay as follows:

$$\tau_{s,k}^{\mathrm{comp}} = \frac{\Psi_s \gamma_s}{f_k}. \tag{4}$$

Here, γ_s represents the computational density of the task offloaded by UV s, and f_k is the CPU clock speed of FV k. The CPU clock speed is bounded, so we have $f_{\min} \le f_k \le f_{\max}$, $\forall k \in K$.

When tasks are processed by FVs, we assume that the primary energy consumption is due to the CPU, and we ignore other types of energy consumption at the FV. The CPU energy consumption for processing tasks offloaded by UVs is typically proportional to the CPU clock speed of FV k [15], and can be modeled as follows:

$$E_{s,k}^{\mathrm{comp}} = X_s^k f_k^2 \Psi_s \zeta. \tag{5}$$

Here, ζ is an effective energy coefficient related to the chip architecture. The more computational resources allocated by the FV, the lower the computational delay, but the higher the energy consumption. Therefore, it is meaningful to minimize energy consumption by reasonably allocating resources based on the communication and computation requirements of tasks.

Typically, the data size of computational results is very small and can be neglected. Therefore, the total latency for task offloading between UV s and FV k can be expressed as follows:

$$\tau_{s,k}^{\mathrm{total}} = \tau_{s,k}^{\mathrm{com}} + \tau_{s,k}^{\mathrm{comp}}. \tag{6}$$

Similarly, the energy consumption for task offloading between UV s and FV k can be calculated as follows:

$$E_{s,k}^{\mathrm{total}} = E_{s,k}^{\mathrm{com}} + E_{s,k}^{\mathrm{comp}}. \tag{7}$$

TABLE I
TASK CLASSES OF VEHICULAR APPLICATIONS

Service class	Example	Computing requirement	Latency requirement
Class 1	Multimedia and passenger entertainment activities	High	1-1.5 s
Class 2	Image-assisted navigation, parking navigation, and optional security applications	Medium	2-2.5 s
Class 3	Autonomous driving and road safety applications	Low	3 s

D. Problem Formulation

The objective of this work is to minimize the energy consumption in the VFC network by allocating appropriate computational resources and transmission power to UVs. Efficient resource allocation can also achieve lower latency while meeting specific constraints, as outlined below:

$$\begin{aligned} \text{P:} \min_{X_s^k, P_s^n} \quad & \sum_{s\in S}\sum_{k\in K} E_{s,k}^{\text{total}}, \\ \text{s.t.} \quad & C1: 0 \le P_s^n \le P_{\max}, \quad \forall s \in S, \\ & C2: X_s^k \in \{0,1\}, \quad \forall s \in S, \forall k \in K, \\ & C3: \sum_{s\in S} X_s^k \le 1, \quad \forall k \in K, \\ & C4: \sum_{k\in K} X_s^k \le 1, \quad \forall s \in S, \\ & C5: \tau_{s,k}^{\text{total}} < \tau_{\max}^{\text{total}}, \quad \forall s \in S, \forall k \in K. \end{aligned} \tag{8}$$

Here, C1 represents the power constraint of UVs on PRB, where $P_{\max}$ is the maximum transmission power of UVs. C2 ensures the binary limitation of computational resource allocation. C3 and C4 ensure that each UV can only select one FV for offloading, and each FV can only process one task. C5 imposes a latency constraint on UVs, where $\tau_{\max}$ is the maximum tolerable latency of UV s.

Problem P is a MINLP problem and solving it is NP-hard due to the presence of binary variables X_s^k. However, we can decompose problem P into two sub-problems, namely UV-FV matching optimization (for computational resource allocation) and UV transmission power allocation (for communication resource allocation). This decomposition allows us to solve each sub-problem individually and simplifies the entire optimization process.

In the first phase, computational resources are allocated by matching UVs with suitable FVs. This matching sub-problem, denoted as P1, can be formulated as follows:

$$\begin{aligned} \text{P1:} \min_{X_s^k} \quad & \sum_{s\in S}\sum_{k\in K} E_{s,k}^{\text{comp}}, \\ \text{s.t.} \quad & C2, C3, C4, C5. \end{aligned} \tag{9}$$

Solving this sub-problem, which involves MINLP and non-convexity, is challenging. Therefore, we employ the Hungarian algorithm to find a satisfactory matching solution.

In the second phase, UVs need to allocate their transmission power on PRBs to minimize energy consumption in the network. This leads to the transmission power allocation sub-problem, denoted as P2, which can be formulated as follows:

$$\begin{aligned} \text{P2:} \min_{P_s^n} \quad & \sum_{s\in S}\sum_{k\in K} E_{s,k}^{\text{com}}, \\ \text{s.t.} \quad & C1, C5. \end{aligned} \tag{10}$$

In sub-problem P2, the solution involves the transmission power allocation for all UVs. To address this problem, we utilize the PSO algorithm, which effectively reduces energy consumption.

Generally, problem P is a non-convex MINLP and solving it is NP-hard. Heuristic algorithms or reinforcement learning methods are typically used to solve such optimization problems. As mentioned earlier, problem P can be viewed as the joint optimization of sub-problems P1 and P2.

III. ALGORITHMS

In the task offloading scenario, UVs compete for computational and communication resources through UV-FV matching and transmission power allocation. This necessitates the optimization of joint resource allocation, which involves multiple interdependent decisions. When a UV occupies a PRB, the UV-FV matching problem, denoted as sub-problem P1, needs to be resolved before optimizing the UV's transmission power. Subsequently, we focus on optimizing the transmission power allocation for UVs to minimize energy consumption while maintaining task latency constraints.

A. Computational Resource Allocation

Computational resource allocation involves assigning suitable FVs to UVs. It is assumed that computational resources are abundant, with the number of FVs exceeding the number of UVs to ensure accommodation for all UVs. To ensure fairness, each FV can only process one task during a single allocation, and UVs' tasks can only be offloaded to a single FV. Consequently, sub-problem P1 naturally becomes a one-to-one matching problem.

The average link gain between UV s and FV k is calculated as $G_{s,k}^{\text{avg}} = \frac{\sum_{n\in N} G_{s,k}^n}{|N|}$. It is assumed that the preference of UV s for FV k is jointly determined by the average link gain and the CPU clock speed of FV k. The preferences for UV s and FV k are defined as follows:

$$\theta_s^k = \omega_1 G_{s,k}^{\text{avg}} + \omega_2 f_k. \tag{11}$$

Here, ω_1 and ω_2 are bias factors that can be freely set based on the computational and communication requirements of tasks. The preference distribution of UVs can be represented by the following matrix:

$$A = \begin{pmatrix} \theta_1^1 & \theta_1^2 & \cdots & \theta_1^k \\ \theta_2^1 & \theta_2^2 & \cdots & \theta_2^k \\ \vdots & \vdots & \ddots & \vdots \\ \theta_s^1 & \theta_s^2 & \cdots & \theta_s^k \end{pmatrix}. \tag{12}$$

To address this matching sub-problem, we employ the Hungarian algorithm.

The Hungarian algorithm is a classical algorithm for solving assignment problems and performs exceptionally well in bipartite graph matching and minimum cost allocation problems. The optimal assignment using the Hungarian algorithm can meet the demands of various application scenarios, such as time-optimality and cost-optimality [16]. For example, in a company, there are n jobs and n employees. Each job can be done by different employees, but each employee can only be assigned one job, and each job can only be assigned to one employee. The efficiency (such as time or cost) of each employee completing each job varies. We need to find an assignment scheme to optimize the total efficiency (such as minimizing the total time or cost). To solve this problem, the Hungarian algorithm requires a cost matrix. The cost matrix C is an $n \times n$ matrix that c_{ij} represents the cost of assigning task i to worker j, as shown below:

$$C = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{pmatrix}. \tag{13}$$

The steps of the Hungarian algorithm are as follows:

Step 1: For each row in the matrix, find the smallest element and subtract it from every element in that row.

Step 2: Similarly, for each column, find the smallest element and subtract it from every element in that column.

Step 3: A minimum number of lines in the cost matrix are drawn to cover all 0.

Step 4: If exactly n lines are required, an optimal assignment has been found, and the algorithm terminates. If the number of lines is less than n, the solution is not yet optimal; proceed to Step 5.

Step 5: Identify the smallest element not covered by any line. Subtract this value from all uncovered elements and add it to all elements covered by two lines. Then, return to Step 3.

The core idea of the Hungarian algorithm is to find an optimal assignment that minimizes the total cost based on the cost matrix. Since the Hungarian algorithm calculates the minimum cost for matching based on the cost matrix, we take matrix M as the cost matrix.

$$M = \begin{pmatrix} A' \\ 0 \end{pmatrix}, \tag{14}$$

$$A' = \begin{pmatrix} U-\theta_1^1 & U-\theta_1^2 & \cdots & U-\theta_1^k \\ U-\theta_2^1 & U-\theta_2^2 & \cdots & U-\theta_2^k \\ \vdots & \vdots & \ddots & \vdots \\ U-\theta_s^1 & U-\theta_s^2 & \cdots & U-\theta_s^k \end{pmatrix}. \tag{15}$$

We find the maximum value U in matrix A. Replace each element θ_i^j in matrix A with $U-\theta_i^j$, and denote the resulting matrix as A'. The Hungarian algorithm mainly solves square matrix problems. Since matrix A' is an $s \times k$ matrix, we add virtual rows to convert it into a square matrix, and set all its values to 0. Using matrix M as the cost matrix, the minimum cost obtained corresponds to the maximum benefit of the original matrix A. Algorithm 1 shows the pseudocode of the Hungarian algorithm.

Algorithm 1 Hungarian Algorithm

Input: Matrix A
Output: Matching matrix X

$U \leftarrow \max(A)$; $A' \leftarrow U - A$
$M \leftarrow \begin{bmatrix} A' \\ 0_{(k-s)\times k} \end{bmatrix}$
repeat
  $\text{row}_{\min} \leftarrow \min(M, \text{dim} = 2)$; $M \leftarrow M - \text{row}_{\min}$
  $\text{col}_{\min} \leftarrow \min(M, \text{dim} = 1)$; $M \leftarrow M - \text{col}_{\min}$
  $\text{cov} \leftarrow \text{MinLineCover}(M)$
  if $|\text{cov}| = k$ **then**
    break
  end if
  $\delta \leftarrow \min(M_{\text{uncovered}})$
  $M \leftarrow M - \delta$ on uncovered rows
  $M \leftarrow M + \delta$ on twice-covered lines
until convergence
$X \leftarrow \text{HungarianMatching}(M[1:s, :])$
return X

B. Interference Coordination

To reduce interference, we assign each PRB to only one UV. For each PRB, we first identify all UVs occupying that PRB. Then, for each active UV, we calculate a priority based on channel quality and historical allocation. UVs with better channel quality and fewer historical allocations are given higher priority. On each PRB, the UV with the highest priority continues to occupy the PRB, while other UVs are required to release it. This ensures that each PRB is occupied by only one UV at a time, thereby avoiding interference. Finally, the historical allocation records are updated to consider the historical usage of UVs in subsequent resource allocations. Through this interference coordination mechanism, efficient utilization of PRBs is ensured, and the total system energy consumption is reduced while meeting task latency constraints.

C. Transmission Power Allocation

PSO is a swarm-based stochastic algorithm proposed originally by Kennedy and Eberhart, which exploits the concepts of the social behavior of animals like bird flocking and fish schooling [17].

The steps of the PSO algorithm are as follows:

Step 1: Parameter setting

Set the number of particles N, the maximum number of iterations T, the inertia weight w, the individual learning factor c_1, the social learning factor c_2, the velocity bound v_{max}, and the position range $[x_{min}, x_{max}]$.

Step 2: Particle Swarm Initialization

- Randomly generate the initial position x_i of each particle i within the solution space.
- Randomly generate the initial velocity v_i of each particle i within the range $[-v_{max}, v_{max}]$.

- Evaluate the fitness of each particle based on the objective function $f(x_i)$.
- Individual Best Initialization: $p_i = x_i$.
- Global Best Initialization Set: $gbest$ as the position with the optimal fitness among all p_i.

Step 3: Iterative optimization process, repeat the following steps until the termination condition is met.

- For each particle i:
  1) Generate random numbers: $r_1, r_2 \in [0, 1]$.
  2) Update velocity: $v_i(t) = w \cdot v_i(t-1) + c_1 \cdot r_1 \cdot (p_i - x_i(t-1)) + c_2 \cdot r_2 \cdot (gbest_i - x_i(t-1))$.
  3) Limit velocity: If v_i exceeds v_{max}, clamp it to $[-v_{max}, v_{max}]$.
  4) Update position: $x_i(t) = x_i(t-1) + v_i(t)$.
  5) Boundary handling: If the position goes beyond the solution space, adjust it using truncation, reflection, or resetting methods.
- For each particle i, compute the fitness of the new position $x_i(t)$ as $f(x_i(t))$.
- If the new fitness is better than the fitness of p_i, then update p_i: $p_i = x_i(t)$.
- Select the position with the optimal fitness from all p_i as the new $gbest$.
- If the maximum number of iterations T is reached or the improvement of $gbest$ is less than a threshold, stop the iteration.

Step 4: Output the results.

In PSO, each potential solution to a given problem is regarded as a particle with a certain velocity, flying through the problem space like a flock of birds [18]. Each particle then combines aspects of its own historical best position and current position records with aspects of the records of one or more agents in the group, along with some random perturbations, to determine its next move in the search space. When all particles have been moved, the next iteration begins. As a whole, the swarm may gradually approach the optimal of the objective function. Algorithm 2 presents the pseudocode of the PSO algorithm.

Using the PSO algorithm to optimize power allocation ensures that the power allocation for each UV is within the allowed range, and the optimal solution is found through iterations. Initially, the inverse of channel quality is calculated and used for initial power allocation, allowing PRBs with good channel quality to be allocated less power and more power to be allocated in cases of poor channel quality to compensate for the adverse channel conditions, thereby optimizing the power allocation strategy. Then, the total energy consumption is set as the objective function for PSO. During transmission power allocation, PSO randomly generates new solutions for each UV on each available PRB. Therefore, PSO generates transmission power for each UV and minimizes energy consumption by optimizing transmission power.

The joint Hungarian algorithm and PSO algorithm are used to solve the energy minimization problem. Algorithm 3 presents the pseudocode for this joint algorithm.

Algorithm 2 PSO Algorithm

Input: objective $f(x)$; swarm size N; iterations T; inertia weight ω; factors c_1, c_2; velocity bound $v_{\max}$; position range $[x_{\min}, x_{\max}]$

Output: global best g_{best} and its fitness $f(g_{\text{best}})$

/* Initialization */
for $i = 1 \,..\, N$ **do**
  $x_i \sim U(x_{\min}, x_{\max})$
  $v_i \sim U(-v_{\max}, v_{\max})$
  $p_i \leftarrow x_i$
end for
$g_{\text{best}} \leftarrow \arg\min_{p_i} f(p_i)$
/* Main loop */
for $t = 1 \,..\, T$ **do**
  for $i = 1 \,..\, N$ **do**
    $r_1, r_2 \sim U(0, 1)$
    $v_i \leftarrow \omega v_i + c_1 r_1 (p_i - x_i) + c_2 r_2 (g_{\text{best}} - x_i)$
    $v_i \leftarrow \text{clamp}(v_i, -v_{\max}, v_{\max})$
    $x_i \leftarrow x_i + v_i$
    $x_i \leftarrow \text{clamp}(x_i, x_{\min}, x_{\max})$
    if $f(x_i) < f(p_i)$ **then**
      $p_i \leftarrow x_i$
    end if
  end for
  $g_{\text{best}} \leftarrow \arg\min_{p_i} f(p_i)$
  if stopping rule satisfied **then**
    break
  end if
end for
return g_{best}, $f(g_{\text{best}})$

TABLE II
PARAMETER SETTINGS

Parameter	Value
PRB bandwidth B	180 kHz
Total PRBs N	60
Background noise σ^2	-80 dBm
CPU speed clock of FVs f	[2.2-3.6] GHz
Computation density γ	[50-150] cycle/bit
Data size of a task Ψ	[8,16] Mb
Latency bound	[1-3] s
Link gain G	[0.001-0.002]
Transmission power threshold P_{max}	37 dBm

IV. PERFORMANCE EVALUATION

In the transmission model, link gain is assumed to be a random value determined by the communication environment and remains constant during the experiment. Parameter settings are shown in Table II. The deployment probabilities for service classes 1, 2, and 3 service requirements are set to $p_{sc_1} = p_{sc_2} = p_{sc_3} = \frac{1}{3}$, and latency requirements are defined by specific service request types.

We compare the algorithm used in this paper with the randomized algorithm, the greedy algorithm, and existing approaches. HHECO algorithm is proposed by [2], which first employs the Gale-Shapley stable matching algorithm and

Algorithm 3 Joint Hungarian-PSO Algorithm for Energy Minimization

Input: S, K

Output: Matching matrix X, Total energy consumption E_{total}, Total latency τ_{total}

for each UV s and FV k **do**
  $\mathrm{A}[s,k] \leftarrow 0.6 \cdot \mathrm{G}_{s,k}^{\text{avg}} + 0.4 \cdot \mathrm{f}_k$
  $U \leftarrow \max(A)$; $A' \leftarrow U - A$
  $M \leftarrow \begin{bmatrix} A' \\ 0_{(k-s)\times k} \end{bmatrix}$
end for
Use Hungarian algorithm to find optimal matching: row_ind, col_ind $\leftarrow$ linear_sum_assignment(cost_matrix)
Construct optimal matching matrix: X $\leftarrow$ zeros(S, K)
for all (s, k) in zip(row_ind, col_ind) **do**
  $\mathrm{X}[s,k] \leftarrow 1$
end for
Initialize power allocation: P_init $\leftarrow$ zeros$(S \cdot N)$
for each UV s and FV k **do**
  if $\mathrm{X}[s,k] = 1$ **then**
    Calculate reciprocal of channel quality: inverse_channel_quality
    Allocate power: P_init$[s \cdot N : (s+1) \cdot N] \leftarrow P_{\max} \cdot$ inverse_channel_quality$/\sum$(inverse_channel_quality)
  end if
end for
Execute PSO optimization: P_optimal $\leftarrow$ pso(pso_objective, lb, ub, args $=$ (X, params))
Calculate: $E_{\text{total}}, \tau_{\text{total}}$

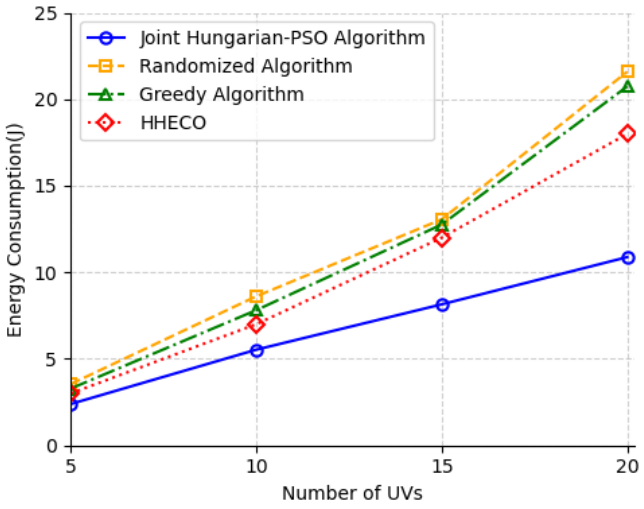

Fig. 2. Energy consumption comparison of algorithms with different UV numbers when the task data size is 8 Mb.

then introduces a decision tree-guided simulated annealing algorithm.

When the task data size is 8 Mb, the total energy consumption of the proposed algorithm remains the lowest as the number of UVs increases from 5 to 20. As shown in Fig. 2, as the number of UVs increases, the slope of energy consumption

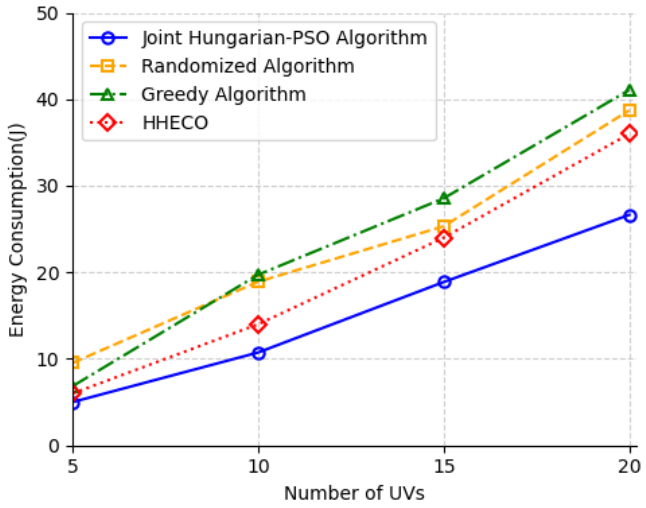

Fig. 3. Energy consumption comparison of algorithms with different UV numbers when the task data size is 16 Mb.

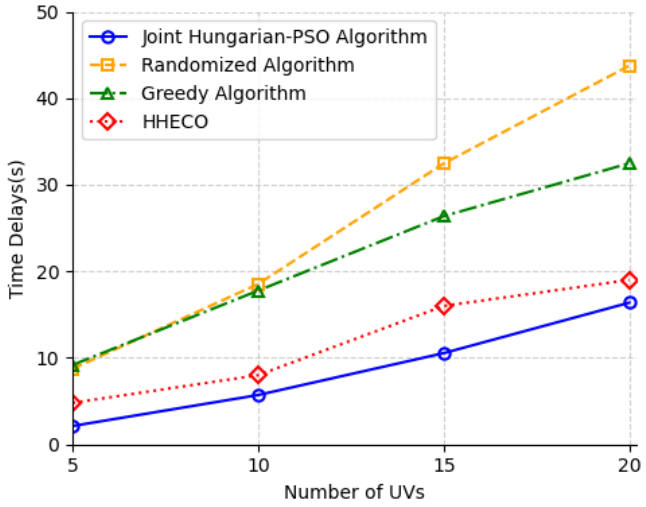

Fig. 4. Total latency comparison of algorithms with different UV numbers when the task data size is 8 Mb.

growth for the algorithm used in this paper is more gradual, indicating stronger robustness in large-scale task scheduling. For example, when the number of UVs is 5, the difference in energy consumption between the proposed algorithm and the HHECO algorithm is 25.52%. When the task data size increases to 16 Mb, as shown in Fig. 3, the energy consumption of all algorithms rises, but the advantage of the proposed algorithm becomes more pronounced. In the scenario with 20 UVs, its energy consumption is approximately 9.36 J lower than that of the HHECO algorithm, demonstrating its superior ability to handle computation-intensive tasks. According to Fig. 4, the total task latency of the proposed algorithm is the lowest across all tested scenarios. Especially when N $>$ 10, its capability to meet strict latency constraints significantly

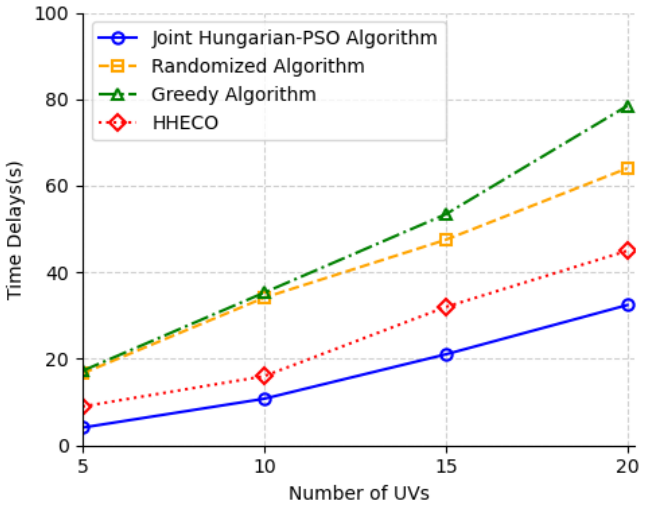

Fig. 5. Total latency comparison of algorithms with different UV numbers when the task data size is 16 Mb.

outperforms that of the comparative algorithms. As shown in Fig. 5, for large tasks, the latency of the random and greedy algorithms deteriorates sharply and may even exceed the maximum tolerable latency. In contrast, the latency curve of the proposed algorithm remains within a controllable range, proving its practicality in handling large tasks.

A comprehensive analysis of Fig. 2 and 3 or Fig. 4 and 5 reveals that the proposed algorithm simultaneously minimizes both energy consumption and time delay, thereby overcoming the traditional optimization dilemma of trading energy for delay or vice versa. This demonstrates the effectiveness of the stepwise optimization framework, which first matches UVs and FVs using the Hungarian algorithm and then regulates transmission power through the PSO algorithm.

V. Conclusion

This paper proposes an optimized method based on the Hungarian algorithm and PSO algorithm for task offloading in VFC to minimize energy consumption. VFC, as a distributed computing architecture, effectively alleviates the problem of insufficient computing power in vehicle terminals. However, the efficiency of task offloading and resource optimization are still of great importance. This paper first employs the Hungarian algorithm to solve the optimal allocation problem between tasks and fog nodes, ensuring the global optimality of task allocation. Then, it employs the PSO algorithm to optimize transmission power and minimize energy consumption. Experimental results indicate that the joint algorithm can ensure the optimality of task allocation while significantly improving the system's real-time performance and resource utilization. Future research will further extend this method to more complex network environments and explore dynamic scenario-based task offloading optimization strategies.

References

[1] X. Wu, S. Zhao, and H. Deng, "Joint task assignment and resource allocation in vfc based on mobility prediction information," *Computer Communications*, vol. 205, pp. 24–34, 2023.

[2] C. Liang, Y. Zhao, Z. Gao, K. Cheng, B. Wang, and L. Huang, "Ealso: joint energy-aware and latency-sensitive task offloading for artificial intelligence of things in vehicular fog computing," *Wireless Networks*, vol. 31, no. 1, pp. 583–599, 2025.

[3] G. Zhang, F. Shen, Z. Liu, Y. Yang, K. Wang, and M.-T. Zhou, "Femto: Fair and energy-minimized task offloading for fog-enabled iot networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4388–4400, 2018.

[4] J. Kim, T. Ha, W. Yoo, and J.-M. Chung, "Task popularity-based energy minimized computation offloading for fog computing wireless networks," *IEEE Wireless Communications Letters*, vol. 8, no. 4, pp. 1200–1203, 2019.

[5] Z. Lin, X. Chen, X. He, D. Tian, Q. Zhang, and P. Chen, "Energy-efficient cooperative task offloading in noma-enabled vehicular fog computing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 7, pp. 7223–7236, 2024.

[6] A. Yadav, P. K. Jana, S. Tiwari, and A. Gaur, "Clustering-based energy efficient task offloading for sustainable fog computing," *IEEE Transactions on Sustainable Computing*, vol. 8, no. 1, pp. 56–67, 2022.

[7] M. Hussain, M. Saad Alam, M. Sufyan Beg, and N. Akhtar, "Towards minimizing delay and energy consumption in vehicular fog computing (vfc)," *Journal of Intelligent & Fuzzy Systems*, vol. 38, no. 5, pp. 6549–6560, 2020.

[8] M. M. Hussain, A. T. Azar, R. Ahmed, S. Umar Amin, B. Qureshi, V. Dinesh Reddy, I. Alam, and Z. I. Khan, "Song: A multi-objective evolutionary algorithm for delay and energy aware facility location in vehicular fog networks," *Sensors*, vol. 23, no. 2, p. 667, 2023.

[9] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3860–3873, 2016.

[10] R. Yadav, W. Zhang, O. Kaiwartya, H. Song, and S. Yu, "Energy-latency tradeoff for dynamic computation offloading in vehicular fog computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14 198–14 211, 2020.

[11] A. Pratap, R. Misra, and S. K. Das, "Maximizing fairness for resource allocation in heterogeneous 5g networks," *IEEE transactions on mobile computing*, vol. 20, no. 2, pp. 603–619, 2019.

[12] X. Huang, L. He, X. Chen, L. Wang, and F. Li, "Revenue and energy efficiency-driven delay-constrained computing task offloading and resource allocation in a vehicular edge computing network: A deep reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 8852–8868, 2021.

[13] C. Liang, Z. Gao, B. Wang, K. Cheng, and Y. Zhao, "Qeclo: A novel qos-aware joint optimization of energy and latency for vfc task offloading," in *2024 27th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE, 2024, pp. 1814–1819.

[14] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[15] S. Xia, Z. Yao, Y. Li, and S. Mao, "Online distributed offloading and computing resource management with energy harvesting for heterogeneous mec-enabled iot," *IEEE Transactions on Wireless Communications*, vol. 20, no. 10, pp. 6743–6757, 2021.

[16] S. Zhang, Y. Xue, H. Zhang, X. Zhou, K. Li, and R. Liu, "Improved hungarian algorithm–based task scheduling optimization strategy for remote sensing big data processing," *Geo-spatial information science*, vol. 27, no. 4, pp. 1141–1154, 2024.

[17] A. G. Gad, "Particle swarm optimization algorithm and its applications: a systematic review," *Archives of computational methods in engineering*, vol. 29, no. 5, pp. 2531–2561, 2022.

[18] M. Jain, V. Saihjpal, N. Singh, and S. B. Singh, "An overview of variants and advancements of pso algorithm," *Applied Sciences*, vol. 12, no. 17, p. 8392, 2022.

Lin Chai graduated with a Bachelor of Science degree in Mathematics and Applied Mathematics from Lyuliang University in 2022. She is currently pursuing a Master of Science degree in Probability Theory and Mathematical Statistics at Yunnan Minzu University. Her research interests focus on Vehicular Ad-hoc Networks.

Jun Wang graduated with a Bachelor of Science degree in Mathematics and Applied Mathematics from Boda College of Jilin Normal University in 2023. She is currently pursuing a Master of Science degree in Probability Theory and Mathematical Statistics at Yunnan Minzu University. Her research interests focus on Vehicular Ad-hoc Networks and Fog Networks.

Shuhui Fang graduated with a Bachelor of Science degree in Applied Statistics from Chuxiong Normal University in 2022. She is currently pursuing a Master of Science degree in Probability Theory and Mathematical Statistics at Yunnan Minzu University. Her research interests focus on Privacy protection and recommendation system.

Yumei Yang graduated with a Bachelor of Science degree in Mathematics and Applied Mathematics from Yunnan Minzu University in 2024. She is currently pursuing a Master of Science degree in Probability Theory and Mathematical Statistics at Yunnan Minzu University. Her research interests focus on Vehicular Ad-hoc Networks.

Wu Wang received his B.S. and M.S. degrees from Yunnan University, China, in 2003 and 2007, respectively. Received his Ph.D. degree from Future University Hakodate, Japan, in 2018. He is currently an associate professor at the School of Mathematics and Computer Science at Yunnan Minzu University. His research interest includes ad hoc networks, neural network, and network security.