LEX LOCALIS-JOURNAL OF LOCAL SELF-GOVERNMENT
ISSN:1581-5374 E-ISSN:1855-363X
VOL. 23, NO. S6(2025)

META-HEURISTIC FUSION FOR 5G VANETS: A GWO–PSO–ACO FRAMEWORK BALANCING LATENCY, ENERGY, AND SPECTRUM

Amjad Alam[1], Nalinda Somasiri[2*], Swathi Ganesan[3], Kamran Ali[4], Tanveer Ahmad[5]

[1, 2, 3]Faculty of Computer Science, York SJ University London, UK
[4] Faculty of Computer Science, Middlesex University London, UK
[5]Faculty of Computer Science, BPP University London, UK

a.alam@yorksj.ac.uk[1]
n.somasiri@yorksj.ac.uk[2]
s.ganesan@yorksj.ac.uk[3]
k.ali@mdx.ac.uk[4]
tanveerahmad@bpp.com[5]

*Corresponding Author: a.alam@yorksj.ac.uk[1]

Abstract*—Next-generation vehicular applications such as augmented-reality navigation and cooperative collision avoidance demand sub-second response times, low on-board energy use, and judicious utilisation of the scarce 5G/DSRC uplink spectrum. We address these conflicting requirements by formulating task-offloading in 5G-enabled vehicular ad-hoc networks (VANETs) as a multi-objective optimisation that minimises end-to-end latency and vehicular energy consumption while maximising deadline reliability and spectral efficiency. A detailed system model captures variable-size tasks generated by mobile vehicles, bandwidth-constrained LTE/5G and Wi-Fi channels, finite-capacity edge servers at roadside units (RSUs), and a remote cloud. Soft-deadline penalties are imposed on tasks whose latency exceeds 1s, and channel-congestion costs discourage excessive simultaneous off-loads. To solve the resulting NP-hard problem we propose an integrated GWO–PSO–ACO swarm optimiser: Grey-Wolf encircling provides global exploration, Particle-Swarm velocity updates accelerate exploitation, and Ant-Colony pheromone learning refines discrete task channel assignments. All three sub-swarms share the best candidate each iteration, yielding rapid yet robust convergence. Extensive simulations with random vehicle velocities (20–100 km/h) and varying numbers of vehicles (1–100, each generating one task) demonstrate that the proposed hybrid GWO–PSO–ACO algorithm consistently outperforms standalone PSO, ACO, and GWO baselines. Averaged over realistic workloads (1-40 MB tasks), the Hybrid achieves total latency reductions of approximately 21.9% and 29.3% compared to PSO and GWO, respectively, while lowering vehicular energy consumption by 12-18% and maintaining high reliability levels of ≈ 80-85% up to critical load points. Spectral efficiency is improved by up to 0.5% at low-to-moderate loads, and composite objective values are reduced by as much as 30–40% under heavy-load conditions. Convergence analysis confirms that the Hybrid reaches near-optimal solutions in fewer iterations than the baselines, making it suitable for real-time vehicular scenarios. Parameter variation tests further validate its scalability under heavier loads and stricter spectrum budgets. These results indicate that the proposed Hybrid optimiser is a robust and effective edge–cloud orchestration mechanism for future QoS- and spectrum-aware V2X services.*

Keywords*—Energy efficiency, Spectral efficiency, Vehicular Edge Computing, QoS-aware Offloading, Metaheuristic Optimization, GWO–PSO–ACO.*

I. INTRODUCTION

Modern vehicular applications such as autonomous driving, collision avoidance, navigation assistance, and infotainment services are placing unprecedented demands on computing and communication resources in Vehicular Ad Hoc Networks (VANETs). These applications often require real-time data processing with strict latency and reliability requirements to ensure safety and a high quality of experience for users. For example, advanced driver-assistance and navigation systems process sensor data and traffic information under tight timing constraints, while in-vehicle infotainment and augmented reality services demand high throughput and low delay. However, vehicles (and their on-board units) have limited processing power and battery capacity, making it challenging to execute such computationally intensive tasks locally [1], [2]. Offloading vehicular tasks to external computing infrastructure has thus emerged as a promising solution to meet the growing computational and QoS requirements of connected

vehicles [3]. Edge computing paradigms, including Mobile Edge Computing (MEC) and Vehicular Edge Computing (VEC), extend cloud-like resources to the network edge (e.g., road-side units or base stations) in proximity to vehicles [4]. By offloading tasks to edge servers on RSUs or to cloud servers, vehicles can significantly reduce their computation load and execution time, thereby improving application responsiveness [5]. Indeed, offloading delay-sensitive workloads to more powerful edge or cloud servers can help meet ultra-low latency constraints that local vehicular resources alone cannot satisfy.

Nevertheless, efficient task offloading in VANETs is non-trivial due to several challenges. First, an offloading decision must carefully trade off the communication delay and energy cost of sending data to the edge/cloud against the speed-up gained in remote processing. If the wireless transmission time and energy consumption outweigh the benefits of faster computation at the server, offloading may hurt performance, especially for small-sized tasks where communication overhead is relatively large. Thus, offloading is advantageous only under the right conditions (e.g. when remote execution is significantly faster than local and the transmission energy is justifiable) [5]. Second, vehicular networks are highly dynamic i.e. vehicles are mobile, network topology changes rapidly, and wireless link quality fluctuates with distance and interference. Connectivity to edge servers can be intermittent or vary in bandwidth, complicating the decision of when and where to offload a task. Offloading strategies must be adaptive to these changing network conditions to avoid excessive delays or failures. Third, resource heterogeneity further complicates the scenario i.e. vehicles differ in computing power and available energy, and edge servers or cloud nodes have varying capacities and loads. Any effective offloading scheme for VANETs must account for this heterogeneity and mobility [5], [6].

Another critical challenge in 5G-enabled VANETs is efficient spectrum utilisation for data offloading. Vehicles today may access network resources via multiple radio access technologies. For instance, high-bandwidth 5G/LTE cellular links or dedicated short-range communications (DSRC) and Wi-Fi based links. Each has limited spectrum availability, and a surge in connected vehicles can easily lead to spectrum congestion if not managed properly. Traditional offloading works often assume ideal or unconstrained wireless links, but in reality, spectrum scarcity can significantly degrade offloading performance by increasing transmission delays and packet losses [7]. Indeed, recent studies note that many prior VEC offloading schemes ignore the impact of spectrum sharing and interference on V2V/V2I communications. In a dense vehicular scenario, multiple vehicles offloading simultaneously over the same cellular uplink or Wi-Fi channel will compete for bandwidth, leading to queuing and higher latency. Hence, incorporating spectrum-awareness into the offloading decision is essential to ensure Quality of Service (QoS) [8]. Spectral efficiency considerations include choosing the appropriate access mode (e.g., offload via 5G macro-cell vs. via local Wi-Fi/DSRC) and possibly enabling spectrum sharing or reuse in a controlled manner so that network capacity is utilised fully without harmful interference. For example, allowing vehicle-to-vehicle (V2V) task offloading on underused portions of the cellular spectrum can improve throughput, as shown in recent work [7]. [9]. However, jointly deciding task offloading targets and spectrum allocation greatly enlarges the solution space. The offloading optimisation must now balance not only computation latency and energy consumption but also account for communication link capacity and interference constraints to maintain QoS requirements (such as delay bounds or throughput thresholds).

Formulating the task offloading problem with objectives of minimising latency and energy (and possibly cost) under these conditions leads to a complex multi-constraint optimisation, which is NP-hard in general. Exact algorithms or brute-force search are infeasible for real-time

decision-making in large-scale vehicular networks due to the exponential number of offloading combinations (each vehicle choosing among local execution, edge server, or cloud, via various link options) [10]. Therefore, researchers have increasingly turned to heuristic and metaheuristic algorithms to obtain near-optimal offloading solutions within acceptable time. Metaheuristics like Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Genetic Algorithms (GA), Grey Wolf Optimizer (GWO), and others have been successfully applied to similar offloading and resource allocation problems in edge/fog computing [11], [12]. These population-based algorithms are well-suited for the large search space and multi-objective nature of the offloading problem, as they can explore many candidate solutions in parallel and converge toward efficient trade-offs between metrics. For instance, prior works have utilised PSO to minimise offloading delays in edge computing, GA to balance energy and latency in multi-access edge systems, and ACO to optimise task assignments in vehicular grids.

In vehicular edge scenarios, GWO in particular has been noted as a promising approach for reducing energy consumption due to its balance of exploration and exploitation and few control parameters. However, a single method often faces limitations in convergence speed and solution quality. Many conventional metaheuristic algorithms can prematurely converge to suboptimal solutions (getting trapped in local minima) or require extensive tuning to handle diverse problem instances. As the problem scale increases (more vehicles and tasks), issues of slow convergence and high computational overhead may arise [12]. This motivates the use of hybrid metaheuristic approaches that combine the strengths of multiple algorithms to overcome each individual algorithm's weaknesses. By hybridising, one algorithm can, for example, provide a diverse global search while another fine-tunes the local search, leading to better precision and faster convergence than either alone.

In this paper, we propose a novel hybrid metaheuristic algorithm termed in this paper as *GWO–PSO–ACO* for optimised task offloading and energy management in 5G-enabled VANETs with QoS and spectral efficiency considerations. The proposed system architecture integrates vehicles, road-side units (RSUs) with edge servers, and remote cloud servers into a three-tier computing environment. Each vehicle generates computational tasks (we consider each vehicle handling a single task at a time in this paper) that can be executed locally or offloaded to either a nearby edge server (via an RSU or 5G base station) or further to a cloud server. Along with the decision of where to offload, the system must also decide how to offload tasks over available wireless interfaces (e.g., using the cellular 5G link or a Wi-Fi/DSRC connection), subject to spectrum availability and interference constraints.

Our hybrid GWO–PSO–ACO algorithm intelligently explores this decision space to minimise the total completion time and energy consumption of vehicular tasks while satisfying QoS requirements (such as task deadline constraints) and improving spectral utilisation. The Grey Wolf Optimiser provides high-level global exploration for promising offloading configurations; PSO quickly refines solutions by learning from the best candidates; and ACO's pheromone updates reinforce effective task-to-resource assignments (e.g., which vehicle should use which RSU or channel). By combining these complementary strategies, our hybrid algorithm avoids premature convergence and more effectively navigates the complex multi-dimensional search space than any single metaheuristic alone.

II. CONTRIBUTIONS

This work addresses the identified challenges by introducing a comprehensive solution for VANET task offloading.

The main contributions are as follows:

1. **Hybrid Metaheuristic Offloading Algorithm:** We propose a novel hybrid optimisation framework combining Grey Wolf Optimizer (GWO), Particle Swarm

Optimization (PSO), and Ant Colony Optimization (ACO) for joint task-offloading and wireless-resource management in vehicular edge computing. This hybrid leverages GWO's exploration capability, PSO's rapid convergence, and ACO's adaptive search to improve solution quality, convergence speed, and robustness against local optima.

2. **QoS and Spectrum-Aware Offloading Model:** We formulate a multi-objective optimisation model for 5G-enabled VANETs that explicitly incorporates spectrum constraints, multiple access modes, and interference limitations. The proposed formulation minimises task latency and energy consumption while ensuring QoS requirements and maximising spectral efficiency.
3. **Mobility-Aware Edge–Cloud Architecture:** We design a vehicular network architecture integrating mobile vehicles, RSU-based edge servers, and cloud resources, with single-task offloading per vehicle. The model accounts for RSU handover and cloud fallback mechanisms to maintain service continuity under mobility.
4. **Performance Evaluation:** We conduct extensive simulations comparing the proposed hybrid algorithm with individual metaheuristics and other baseline approaches to yield better reduced end-to-end latency, lower energy consumption, better load balancing, and higher bandwidth utilisation. To carry out Convergence and scalability analyses to confirm its effectiveness with higher number of vehicles and higher speed.

III. PAPER ORGANISATION

The remainder of this paper is organized as follows. Section IV provides a detailed review of related works on metaheuristic-based task offloading and QoS- and spectrum-aware optimization approaches. Section V describes the proposed hybrid GWO–PSO–ACO methodology and algorithmic design. Section VI discusses the simulation setup, performance metrics, and parameter configurations, while Section VII presents the experimental results and analysis. Finally, Section VIII concludes the paper and highlights potential future research directions.

IV. RELATED WORKS

A. Metaheuristic-Based Task Offloading in Edge Computing

A variety of metaheuristic algorithms have been employed in recent years to solve task offloading and resource allocation problems in edge and fog computing environments. These algorithms are attractive for their ability to tackle NP-hard optimisation problems and find near-optimal solutions within reasonable time. Evolutionary and swarm-based techniques in particular have seen wide use. For example, Abbas et al. (2021) applied three well-known metaheuristics i.e. Ant Colony Optimisation (ACO), Whale Optimization Algorithm (WOA), and Grey Wolf Optimiser (GWO) to optimize the selection of offloading tasks in a mobile edge computing scenario involving IoT devices. Their study highlighted the trade-off between energy consumption and delay, and the results showed that GWO achieved the best performance in terms of reducing energy usage and execution latency, outperforming the ACO and WOA approaches [6]. This result highlights GWO's effectiveness in jointly optimizing energy and delay—helping to explain why GWO has become a popular choice for edge offloading problems.

Chanyour et al. (2022) formulated the offloading decision as a multi-objective problem to simultaneously minimize total power consumption and task execution delay in a fog computing environment [14]. They employed a Non-dominated Sorting Genetic Algorithm (NSGA-II) alongside the Bees Algorithm to solve this bi-objective optimisation, demonstrating that metaheuristics can obtain a set of Pareto-optimal solutions balancing energy and latency. Their

simulation results showed improved trade-offs i.e. the proposed methods achieved lower overall energy consumption for given delay levels (and vice versa) compared to heuristic baselines, and they ensured a better offloading probability (the fraction of tasks offloaded) under power constraints. Likewise, Shahidinejad (2022) treated task offloading in edge–cloud networks as a multi-objective optimisation and proposed an improved NSGA-II algorithm (referred to as iNSGA-II) for this purpose [12]. By enhancing the genetic operators (crossover and mutation), their iNSGA-II achieved faster convergence than standard evolutionary algorithms, leading to higher edge server utilisation and noticeable reductions in both energy consumption and task execution time. This work underlined that even within a single metaheuristic family (GA-based), careful algorithmic improvements can yield significant gains in offloading efficiency.

Several other studies have explored metaheuristics like Particle Swarm Optimisation and Simulated Annealing for offloading. In the vehicular network domain, You and Tang (2021) [16] employed a PSO algorithm to efficiently allocate tasks to edge servers in an industrial IoT setting, reporting substantial latency improvements over random or greedy assignments. Keshari et al. (2021) [17] also investigated PSO for vehicular edge computing offloading, confirming its effectiveness in reducing task completion time under varying network conditions. Genetic Algorithms have been used to handle task scheduling and offloading with multiple objectives, as seen in works like Li & Zhu. (2020) [18] and Elkawkagy et al. (2025) [19]. Ant Colony Optimisation has been less common for offloading, but it has shown promise in problems like vehicular route selection and could be adapted to task assignment problems due to its strength in combinatorial search. In summary, prior studies indicate that metaheuristic offloading strategies outperform naive or static schemes by exploring the search space more thoroughly and finding more efficient task–resource allocations. They are especially useful in scenarios with multiple tasks and servers, where exhaustive search is infeasible.

B. Conventional Metaheuristics

Despite their successes, conventional metaheuristic approaches in offloading come with notable limitations, which recent surveys have pointed out [20]. A comprehensive systematic review by Li & Chen (2024) examined numerous metaheuristic task offloading techniques aimed at minimising energy consumption in edge computing [21]. The review found that the majority of existing techniques rely on a single metaheuristic algorithm and thus suffer from issues like poor convergence speed, imbalance between local and global search, and high computational complexity. In many cases, algorithms like PSO or GA can converge prematurely to suboptimal solutions (local minima) if the population diversity is not maintained, or they may require a large number of iterations to adequately explore the search space, which is problematic for time-sensitive applications. The survey by Li & Chen [21] noted that Grey Wolf Optimiser (GWO) has emerged as a promising approach among single metaheuristics for energy-efficient offloading, owing to its efficient exploration mechanism and minimal parameter tuning. However, even GWO and similar algorithms can stagnate or oscillate if the problem landscape is complex with many constraints.

The recent survey by Pilli, Mohapatra, and Reddy (2025) underscores hybrid meta-heuristic designs as a promising direction for priority-aware computation offloading, noting that blending global search with structure-exploiting local improvement can mitigate typical weaknesses of pure meta-heuristics namely slow convergence and limited solution precision [15]. Complementing this line, Li et al. (2025) move beyond purely heuristic search by casting joint task offloading and resource allocation in hybrid MEC-enabled LEO satellite networks as a hierarchical game. Their game-theoretic formulation provides principled coordination across layers and supports explicit service constraints (e.g., latency/throughput), illustrating how

pairing meta-heuristic exploration with equilibrium-driven or exact subroutines can deliver both high-quality solutions and stronger QoS guarantees. Together, these works point to hybrid pipelines in which meta-heuristics handle global exploration while problem-structured optimization enforces feasibility and accelerates convergence.

Despite these advances, a gap remains in understanding scalability. Many proposed solutions have only been evaluated on moderate-sized scenarios; their performance in large-scale, dense network settings is rarely reported. Computational complexity can grow quickly with problem size, so efficient encodings and search operators are needed to keep runtime feasible. Our proposed hybrid GWO–PSO–ACO addresses this by combining algorithms that complement each other's search behaviour, effectively accelerating convergence and avoiding long stalls in suboptimal regions of the search space. In summary, the literature points to a need for more robust and faster-converging offloading algorithms. This need directly motivates our adoption of a hybrid metaheuristic strategy in this work.

C. QoS-Aware and Spectrum-Aware Offloading Schemes

In parallel to algorithmic improvements, another line of work focuses on making offloading more QoS-aware and network-aware. Traditional formulations often optimize average delay or energy but may not enforce per-task QoS (e.g., latency deadlines) and often oversimplify communication models (e.g., assuming a single homogeneous link) [23]. To support heterogeneous QoS demands, some studies introduce multi-priority or multi-class frameworks. For example, Huang et al. proposed a QoS-aware resource allocation scheme in fog-enabled IoT, using Analytic Hierarchy Process (AHP) to prioritize QoS parameters per device type. A matching-game approach then jointly optimized task offloading and resource block allocation, ensuring that high-priority tasks (e.g., ultra-low latency or high reliability) receive preferential treatment. Their scheme improved load balancing and reduced network overhead, showing that explicitly handling QoS distinctions (e.g., treating URLLC vs. broadband traffic differently) can enhance performance [9].

However, Huang et al.'s [9] method relies on a specific hierarchical and game-theoretic framework and does not leverage metaheuristics; moreover, it assumes the availability of resource blocks (RBs) in a single network context and does not explicitly address multi-RAT (radio access technology) scenarios or spectrum sharing [9]. Until recently, very few offloading studies explicitly incorporated communication-resource constraints (spectrum availability, interference, etc.) into their decision models. One notable work is Huang et al. (2024) (a different Huang, et al.) who investigated a joint spectrum sharing and task offloading problem for vehicular edge computing [9]. They observed that prior VEC works either focused solely on V2I offloading or implicitly assumed abundant spectrum, failing to account for spectrum scarcity in growing vehicular networks.

To address this, they proposed a coalition formation game approach that allowed vehicles to offload tasks via both V2I and V2V links, with V2V communications opportunistically sharing the uplink spectrum of V2I links. This innovative spectrum-sharing mechanism improved spectral utilisation and significantly reduced the total task completion delay (by over 50% in many cases) compared to baseline schemes without spectrum sharing [9]. The work of Huang et al. (2024) clearly demonstrates the benefit of incorporating spectrum awareness i.e. by treating channel allocation and offloading target selection jointly, the scheme can exploit underused bandwidth and avoid bottlenecks on congested links. Nevertheless, their solution is tailored to a specific cooperative game model and optimises primarily for delay (network throughput), without explicitly considering the energy consumption of vehicles.

Moreover, the game-theoretic algorithm, while distributed, may face complexity issues as the number of vehicles and sub-channels grows, and it does not utilise metaheuristic

optimisation (instead, it iteratively forms coalitions). Another relevant study by Nguyen et al. (2023) examined a collaborative UAV network handling inter-dependent edge computing task [24]. They formulated an optimisation to minimise average latency by jointly deciding task offloading (for tasks that are part of a dependency DAG) and allocating communication resources (transmit power and channels) among UAVs. Owing to the NP-hard nature of the joint problem, they split it and applied a discrete Whale Optimisation Algorithm (D-WOA) to the offloading decision subproblem, while solving the resource allocation subproblem via convex optimisation. This approach yielded a suboptimal but efficient solution that outperformed several benchmarks in reducing latency and improving uplink transmission rates for UAV-served users [24]. Nguyen et al.'s work is notable for considering task dependency (workflow topology) in offloading, a QoS aspect (since some tasks must wait for others). It also considers the power limits of UAVs (energy constraint) and effectively treats communication channels as limited resources.

The use of a metaheuristic (WOA) aligns with our direction, though their UAV swarm scenario with DAG tasks differs from VANETs and lacks explicit multi-RAT spectrum handling (assuming a single-band UAV link). Overall, literature is evolving toward more advanced offloading strategies that consider multiple objectives (e.g., delay, energy, cost), QoS, and realistic constraints 20], [25], [26] Still, gaps remain, indicating that few studies unify hybrid metaheuristic optimisation with QoS- and spectrum-aware offloading. Most either focus narrowly on computing or assume simple communication models, while those addressing spectrum or QoS often use game theory or heuristics and optimise a single objective [20], [26].

This leaves a research gap in achieving both high spectral efficiency and strong QoS guarantees in a multi-tier vehicular offloading system using advanced optimisation algorithms. In summary, our work bridges the two research threads identified above. On one hand, we introduce a hybrid GWO–PSO–ACO metaheuristic to solve the complex offloading problem with faster convergence and higher solution quality. On the other hand, we formulate the offloading decision to explicitly include spectrum constraints and QoS deadlines alongside the usual latency and energy objectives. Our evaluation demonstrates that this fused approach can simultaneously reduce vehicle energy consumption and task delay while respecting spectrum limitations, achieving a combination of benefits that previous approaches could not. In doing so, we present a novel solution that advances the state of the art in metaheuristic offloading and yields a VANET offloading framework more attuned to spectral efficiency and QoS compliance than prior methods. Next, we present the system model and the hybrid algorithm in detail, followed by performance evaluation results that show how our approach addresses the research gaps identified above.

V. METHODOLOGY

This section details the design rationale, mathematical formulation, and algorithmic workflow underpinning our hybrid GWO–PSO–ACO task-offloading framework for 5G-enabled VANETs. We first present the system architecture and core assumptions including mobile vehicles with heterogeneous workloads, bandwidth-constrained LTE/5G and Wi-Fi uplinks, and multi-tier computing resources at RSUs and the cloud. We then formalise latency, energy, reliability, and spectral-efficiency metrics, framing a multi-objective optimisation that captures practical soft-deadline penalties and channel-capacity limits. Finally, we describe how Grey Wolf Optimiser, Particle Swarm Optimisation, and Ant Colony Optimization operate in parallel as an ensemble, iteratively sharing the best candidate solution to achieve fast convergence and robust performance.

Figure 1 presents a flowchart overview of the proposed methodology

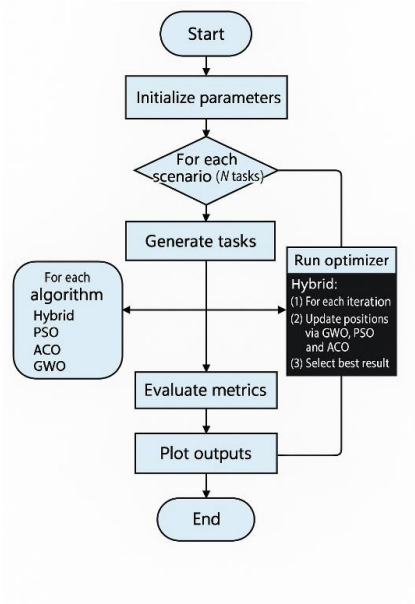

Fig 1: Flowchart of proposed Methodology

VI. SYSTEM MODEL

Figure 2 presents the architecture of the proposed vehicular edge computing framework. The system consists of a set of mobile vehicles equipped with On-Board Units (OBUs), a network of Roadside Units (RSUs), edge servers co-located with RSUs, and a centralised cloud server. Each vehicle generates computational tasks and must decide whether to execute them locally or offload them to a nearby RSU or the cloud. RSUs communicate with the cloud via high-speed backhaul links. The wireless uplink operates over 5G NR or LTE bands, with shared bandwidth and velocity-dependent transmission rates. A hybrid GWO–PSO–ACO optimisation engine is used to determine optimal offloading decisions based on real-time task profiles, bandwidth availability, and QoS constraints.

A. System Architecture

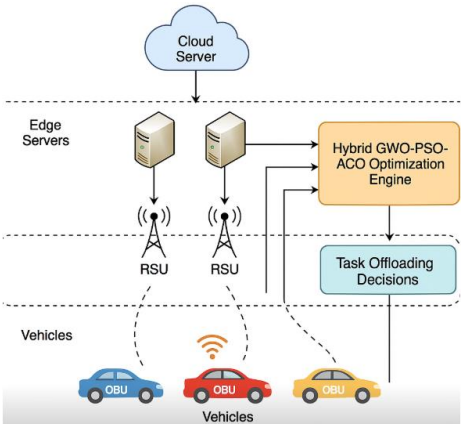

Fig 2: System Architecture

B. Decision Engine and Optimization Deployment

The hybrid GWO–PSO–ACO optimiser is executed at the RSU-side edge controller. Vehicles periodically send task metadata (e.g., input size, CPU cycles, velocity) to their nearest RSU. The RSU aggregates this information and applies the optimiser to decide where each task should be executed i.e. locally at the vehicle, at the RSU edge server, or offloaded to the cloud. The final decision is sent back to the vehicle, which then either processes the task locally or initiates transmission. This centralised optimisation ensures global coordination across multiple vehicles and enables efficient spectrum and resource usage.

We consider a three-tier vehicular edge computing architecture consisting of mobile vehicles, road-side units (RSUs) with edge servers, and a remote cloud. Vehicles (forming a 5G-enabled VANET) generate computational tasks that can be processed locally or offloaded. Each vehicle has at most one active task at a time (no task queue per vehicle). Tasks have varying input data sizes and processing requirements. The vehicles are equipped with wireless interfaces for LTE/5G cellular and WiFi (IEEE 802.11p or similar) connectivity. This allows two offloading paths:

(1) Vehicle-to-Infrastructure (V2I) via cellular to a 5G base station/RSU that provides edge computing services, or

(2) Vehicle-to-RSU via WiFi for local edge processing.

In both cases, RSUs are connected to a core network enabling further offload to the cloud if needed.

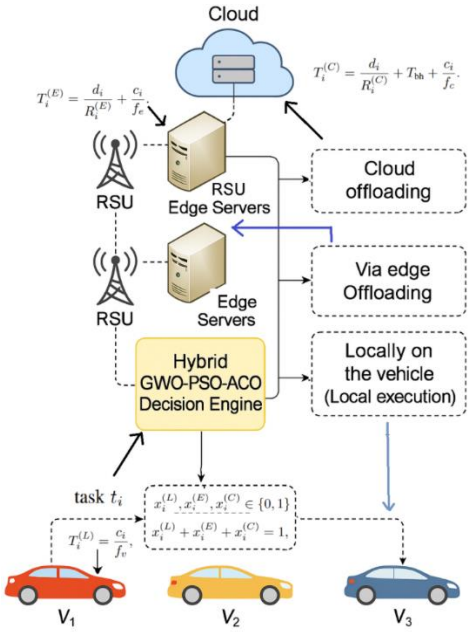

Fig 3: System Model

Figure 3 illustrates the model. Mobile vehicles V_i (for $i = 1,2,\dots,N$) each generate a task t_i. A task t_i is characterised by a data size d_i (in bits) that must be uploaded if offloaded, and a computational workload c_i (in CPU cycles) required to execute the task. Each task can be executed in one of three places:

1. Locally on the vehicle (Local execution): The vehicle's on-board unit (OBU) processes the task.
2. At the edge (Edge offloading): The task is transmitted to a nearby RSU or 5G base station and executed on the edge server (MEC server) there.
3. In the cloud (Cloud offloading): The task is transmitted over the cellular network to a remote cloud server for execution (possibly via the RSU/base station as a relay).

The network uplink for task offloading has limited bandwidth and channel resources. We assume a total of N_{ch} orthogonal channels (subchannels or time-frequency resource blocks) are available for vehicular uplink transmission (across LTE/5G or WiFi). If more than N_{ch} vehicles attempt to offload simultaneously, they must share channels (e.g., by time-division or causing interference), which degrades each vehicle's data rate. Each channel h has a bandwidth B (Hz) and a corresponding data rate depending on channel conditions. Let R_i (in bit/s) denote the achievable uplink rate for vehicle i when it offloads (this encapsulates bandwidth B and SINR conditions). For simplicity, we assume R_i is the individual data rate when using one channel exclusively; if multiple vehicles share or exceed available channels, the effective rate per vehicle will be reduced (modelled later via a spectrum penalty). Downlink transmission of the task result is assumed to be very small or delivered over the same link with negligible delay (a common assumption, as output data sizes are typically much smaller than input sizes).

The edge servers at RSUs (or 5G base stations) have finite computing capacity. Let f_e be the processing speed (CPU cycles per second) be available to each RSU's edge server, and f_c for the cloud server. Similarly, each vehicle's CPU has speed f_v cycles/s. Typically $f_c \gg f_e \gg f_v$ (cloud is most powerful, then edge, then vehicle). For example, an edge server might be a micro-datacenter with several cores (e.g. 10× the speed of a single vehicle's CPU), while the cloud has virtually unlimited capacity (for modelling, a very high speed or enough resources to process all offloaded tasks in parallel). Vehicles may be moving; however, we assume that during the execution of its task, a vehicle remains under coverage of the same RSU/cellular base station. (In the simulation, we can treat vehicles as either fixed or moving at moderate speeds, and we account for mobility's effect via a reliability factor for the communication link as described later). Reliability of task execution can be impacted by mobility (loss of connection) or resource outages. We incorporate reliability into the model by assigning a success probability to each offloading option (detailed in the formulation below).

In summary, the system model entails each vehicle deciding whether to execute its task locally or offload it to an edge or cloud server. Offloading yields lower computation time (especially for large tasks) but incurs delays, energy cost for wireless communication, and potential packet loss or channel contention. Local execution avoids network usage (no transmission delay or wireless energy cost) but may take longer and drain the vehicle's battery. The goal is to optimise the offloading decisions for all vehicles to achieve a balance between low latency, low energy consumption, high reliability, and efficient spectrum usage.

VII. MATHEMATICAL FORMULATION

We now formulate the task of offloading and resource allocation problems mathematically. We define decision variables, performance metrics (latency, energy, etc.), and constraints.

A. *Decision Variables:*

For each vehicle i we introduce binary offloading decision variables:

$$x_i^{(L)}, x_i^{(E)}, x_i^{(C)} \in 0,1 \tag{1}$$

indicating whether task t_i is executed Locally, at the Edge, or in the Cloud, respectively. Each task must be executed exactly in one place, so we have the assignment constraint:

$$x_i^{(L)} + x_i^{(E)} + x_i^{(C)} = 1, \qquad \forall i = 1, \dots, N. \tag{2}$$

For example, $x_i^{(E)} = 1$ means vehicle i offloads its task to an edge server (and $x_i^{(L)} = x_i^{(C)} = 0$ in that case).

B. *Latency Model:*

The latency T_i for task t_i depends on the decision:

1. Local execution latency: If $x_i^{(L)} = 1$, latency consists solely of local processing time. We model this as

$$T_i^{(L)} = \frac{c_i}{f_v}, \tag{3}$$

i.e. the CPU cycles required are divided by the vehicle's CPU speed.

2. Edge offloading latency: If $x_i^{(E)} = 1$, latency includes two components: (i) uplink transmission time to send d_i bits to the RSU, and (ii) edge computing time. The transmission time is $T_i^{tx(edge)} = \frac{d_i}{R_i^{(E)}}$, where $R_i^{(E)}$ is the uplink data rate for vehicle i on the RSU/5G link. The edge processing time is $T_i^{proc(edge)} = \frac{c_i}{f_e}$ (assuming the edge server can allocate sufficient CPU to this task – we discuss capacity constraints shortly). Thus

$$T_i^{(E)} = \frac{d_i}{R_i^{(E)}} + \frac{c_i}{f_e} \tag{4}$$

If multiple tasks are offloaded to the same edge server, we assume the server can process them in parallel up to its capacity; beyond that, tasks will experience additional waiting time (modeled via a constraint/penalty later). We neglect the output download time as output data is typically small.

3. Cloud offloading latency: If $x_i^{(C)} = 1$, latency includes (i) uplink transmission to the cellular network, (ii) backhaul/core network transit to the cloud, and (iii) cloud processing. We combine (i) and (ii) as an effective transmission delay $T_i^{tx(cloud)} = \frac{d_i}{R_i^{(C)}} + T_{bh}$, where $R_i^{(C)}$ is the uplink rate to cloud (which could be the same as $R_i^{(E)}$ if the first hop is the bottleneck) and T_{bh} is the additional backhaul latency for reaching the cloud data centre. T_{bh} might be a fixed 10–50ms depending on network distance (5G MEC has very small T_{bh}, while a distant cloud might add tens of milliseconds). The cloud processing time is $T_i^{proc(cloud)} = \frac{c_i}{f_c}$. Thus,

$$T_i^{(C)} = \frac{d_i}{R_i^{(C)}} + T_{bh} + \frac{c_i}{f_c} \tag{5}$$

Typically, $f_c \gg f_e$, so cloud execution is very fast, but T_{bh} might make total cloud latency higher than edge for small tasks. We assume tasks are independent, so no waiting is needed at cloud (cloud has ample capacity).

For each vehicle i, the actual latency T_i is determined by its decision:

$$T_i = x_i^{(L)}.T_i^{(L)} + x_i^{(E)}.T_i^{(E)} + x_i^{(C)}.T_i^{(C)} \tag{6}$$

This selects the appropriate latency formula.

C. *Mobility-Aware Channel Rate Modelling*

In vehicular environments, uplink data rate R_i is affected by the vehicle's velocity v_i, due to Doppler shift, RSU dwell time, and channel fading. To model this, we used a mobility-aware rate function that degrades performance with increasing speed [7]. Each vehicle's uplink rate is approximated by:

$$R_i = \frac{R_{\max}}{1 + \frac{v_i - v_{\min}}{v_{\max} - v_{\min}}} \tag{7}$$

where:

- $v_i \in [v_{\min}, v_{\max}]$ is the speed of vehicle i in km/h,
- $R_{\max}$ is the peak data rate achievable at the minimum speed,
- The denominator models a linear rate drop-off with speed.

For example, if $R_{\max} = 15$ Mbps, $v_{\min} = 20$ km/h, and $v_{\max} = 100$ km/h, then a vehicle at 100km/h experiences a rate $R_i \approx 7.3$ Mbps. This value of R_i is used in the transmission delay term $\frac{d_i}{R_i}$ in energy model for offloaded tasks.

Optionally, velocity-aware reliability can also be incorporated by decreasing the probability of successful offloading as speed increases, reflecting reduced RSU dwell time or higher handover risk.

D. Energy Consumption Model:

We focus on the energy consumed by vehicles (as vehicles are energy-constrained). If a task runs locally, the vehicle expends CPU energy; if offloaded, the vehicle expends radio transmission energy (assuming the edge and cloud energy costs are less critical or not charged to the vehicle). We define:

1. Local computation energy: If executed locally, the energy consumed by the vehicle's CPU is
$$E_i^{(L)} = \kappa . c_i . (f_v)^{\alpha} \quad (8)$$
where κ and α are device-specific constants related to CPU power consumption. A common model from DVFS (dynamic voltage frequency scaling) literature is $\alpha = 2$ or 3, meaning energy per cycle grows with the square or cube of the CPU frequency. For simplicity, one can use $E_i^{(L)} = \eta . c_i$ where η (Joules per cycle) is a constant indicating energy cost per CPU cycle on the vehicle. This captures that larger tasks (more cycles) drain more battery [9].
2. Offloading transmission energy: If offloaded (edge or cloud), the vehicle's primary energy cost is from transmitting d_i bits over wireless. We model this as
$$E_i^{(TX)} = P_{tx,i} \times T_i^{tx} \quad (9)$$
where $P_{tx,i}$ is the transmit power of vehicle i and T_i^{tx} is the transmission duration. Using $T_i^{tx(edge)}$ or $T_i^{tx(cloud)}$ as appropriate, we get $E_i^{(E)} = P_{tx,i} . \frac{d_i}{R_i^{(E)}}$ if offloaded to edge, or $E_i^{(C)} = P_{tx,i} . \frac{d_i}{R_i^{(C)}}$ for cloud (the transmit power might differ if different interfaces are used, but we assume similar power usage for simplicity). We neglect the tiny reception energy for result downloading.

Thus, the vehicle's energy E_i for task i is:
$$E_i = x_i^{(L)} . E_i^{(L)} + x_i^{(E)} . E_i^{(TX)} + x_i^{(C)} . E_i^{(TX)} \quad (10)$$
where we used the same $E_i^{(TX)}$ for edge/cloud as the transmit action is similar (if needed, one could distinguish P_{tx} for cellular vs WiFi, but that can be folded into R_i or T_i differences).

E. Reliability and QoS

We incorporate a reliability factor to account for the probability that a task is successfully executed and returned to the vehicle within acceptable time. Several factors affect reliability:

1. Communication reliability: Offloading over wireless can fail due to packet loss, interference, or the vehicle moving out of coverage. We denote by $\rho_{comm}^{(E)}$ the success probability of edge offloading transmission (e.g., a 5G link might have $\rho_{comm}^{E} \approx 0.99$ reliability), and ρ_{comm}^{C} for cloud offloading (possibly slightly lower if more network hops are involved). Local execution has essentially $\rho_{comm}^{L} = 1$ (no communication needed).
2. Processing reliability: There is a small chance an edge server or cloud could fail or be too overloaded to complete a task in time. We denote ρ_{proc}^{E} for edge computing reliability (maybe very high if edge is stable, e.g. 0.999) and ρ_{proc}^{C} for cloud (also high).
3. Deadline/QoS satisfaction: Often tasks have a deadline or maximum latency D_i (QoS requirement). If T_i exceeds D_i, the task result is no longer useful (considered a failure for QoS). This effectively reduces reliability of that decision. We can incorporate this by setting $\rho_{QoS,i} = 1$ if $T_i \leq D_i$ and 0 if $T_i > D_i$. To keep the problem formulation

continuous, one might approximate this with a very steep penalty for exceeding D_i rather than a hard step function.

For simplicity, we combine these factors into an overall reliability probability ρ_i for each task given the offloading decision:

$$\rho_i = \begin{cases} \rho^{(L)}, & x_i^{(L)} = 1, \\ \rho^{(E)}, & x_i^{(E)} = 1, \\ \rho^{(C)}, & x_i^{(C)} = 1. \end{cases} \tag{11}$$

where $\rho^{(L)}, \rho^{(E)}, \rho^{(C)}$ are the success probabilities for local, edge, and cloud execution respectively. These can be defined as $\rho^{(L)} = \rho_{proc}^{L}$ (nearly 1, assuming the vehicle's OBU rarely fails and no deadline miss locally), $\rho^{(E)} = \rho_{comm}^{(E)} \cdot \rho_{proc}^{E} \cdot \rho_{QoS}^{(E)}$, and similarly for $\rho^{(C)}$. Typically, $\rho^{(L)} \approx 1$, $\rho^{(E)}$ might be slightly less (e.g. 0.98-0.99), and $\rho^{(C)}$ slightly less than $\rho^{(E)}$ (due to longer network path, say 0.95-0.98). The QoS deadline component ρ_{QoS} would be 1 if the predicted latency T_i meets the requirement D_i, or 0 if it exceeds (in practice we will handle this via a penalty term rather than a strict zero, to allow trade-offs).

We will not impose a hard reliability constraint; instead, we include a penalty in the objective for unreliability or QoS violation. For formulation, define an unreliability cost for task i as $U_i = 1 - \rho_i$. This U_i is 0 if success is guaranteed, and positive if there is a failure probability or QoS risk. For example, if i offloads to cloud with $\rho^{(C)} = 0.96$, then $U_i = 0.04$ (4% risk of failure or deadline miss). Our goal is to minimize these unreliability values across all tasks.

F. Spectrum Allocation and Capacity Constraints

The network constraint is that at most N_{ch} tasks can be transmitted simultaneously without sharing spectrum [27]. If $y = \sum_i \left(x_i^{(E)} + x_i^{(C)}\right)$ is the number of offloading vehicles, then:

$$y \leq N_{ch} \tag{12}$$

If ***eq12*** is violated (i.e. more vehicles attempt to offload than channels), then some vehicles must either share a channel or wait. Sharing a channel (e.g. two vehicles on one channel) effectively halves the bandwidth for each, reducing R_i and thus increasing latency T_i (possibly causing deadlines to be missed). Waiting (time-division) would also increase latency. Rather than add a complex scheduling sub-problem, we enforce ***eq12*** as a soft constraint by penalizing any exceedance. Specifically, in the objective we will add a spectrum penalty term if $y > N_{ch}$. One simple penalty is $P_{spec} \times \max(0,, y - N_{ch})$ with a large coefficient P_{spec} to strongly discourage infeasible channel load. In our approach, we prefer to avoid this situation by optimization, so typically the solution will satisfy $y \leq N_{ch}$ automatically if the penalty weight is high.

Next, for edge server capacity: The edge server has finite CPU f_e cycles/s. If too many large tasks are offloaded to a single edge, they may queue. A rough constraint is that the total processing demand of tasks offloaded to a given edge in the time horizon should not exceed what the server can handle. For a single time slot model (all tasks come at once), a constraint can be:

$$\sum_{i \in RSU\,j} c_i \leq f_e \times T_{slot} \tag{13}$$

where T_{slot} is the maximum latency tolerance (e.g. if tasks should finish in under 1 s, we set $T_{slot} = 1$ s). This ensures the edge $server_j$ can process all offloaded tasks within that time if scheduled optimally. Similarly for the cloud (although cloud capacity is very large, we can assume it's not a bottleneck or apply a similar constraint if modelling multiple clouds). In practice, rather than a hard constraint, we incorporate this in reliability: if ***eq13*** is violated, some tasks will be delayed beyond the slot, effectively causing $\rho_{QoS} = 0$ for them (counted in

unreliability). Our optimization will naturally avoid sending more tasks to edge than it can handle because that would incur high latency and QoS penalties.

Finally, each decision variable is binary:

$$x_i^{(L)}, x_i^{(E)}, x_i^{(C)} \in 0,1, \qquad \forall i. \tag{14}$$

This makes the problem a combinatorial optimization (each vehicle chooses one of 3 options).

G. Multi-Objective Optimization Formulation

We now frame the offloading decision problem as a multi-objective optimization. The objectives are:

1. Minimize total latency (or equivalently average latency) of tasks.
2. Minimize total energy consumption of vehicles.
3. Maximize reliability/QoS or equivalently minimize the number of failures or deadline violations.
4. Maximize spectrum efficiency, which in our formulation translates to minimise excessive spectrum usage or channel congestion.

Because these objectives conflict (for example, minimizing latency might mean offloading everything to cloud, which could increase spectrum usage and risk), we formulate a single composite objective function using weighted sums. Decision makers can adjust weights to prioritize certain metrics. Let w_1, w_2, w_3, w_4 be the weight coefficients for latency, energy, unreliability, and spectrum usage respectively.
We first express the metrics in a quantitative form:

1. Total latency: $T_{tot} = \sum_{i=1}^{N} T_i\,(x_i)$, where $T_i(x_i)$ is given by eq-6. This is the sum of all task latencies (we could also use max latency or a weighted sum, but sum/average is chosen for simplicity).
2. Total energy: $E_{tot} = \sum_{i=1}^{N} E_i\,(x_i)$ from eq-10.
3. Total unreliability (QoS cost): $U_{tot} = \sum_{i=1}^{N}\big(1 - \rho_i(x_i)\big)$. This essentially counts the expected number of failed tasks or the sum of failure probabilities. If ρ values are near 1, U_{tot} will be small. In practice, if tasks have deadlines, this term heavily penalizes any task predicted to miss its deadline (which would make $\rho_i \approx 0$ for that task).
4. Spectrum usage cost: We define S_{usage} in terms of total data transmitted or channel usage. A simple choice is

$$S_{usage} = \sum_{i=1}^{N} y_i \frac{d_i}{B},$$

which represents the total time–frequency resource consumed (bits transmitted divided by channel bandwidth—effectively the sum of Tx time across all offloads). However, this is somewhat redundant with latency, since transmission time is part of latency.
Another proxy is simply the number of offloaded tasks

$$y = \sum_{i=1}^{N} (x_i^{(E)} + x_i^{(C)}),$$

i.e. how many tasks are used in the spectrum.
For our multi-objective we penalise the fraction of channels utilised, or any excess usage beyond the available channels. We set

$$S_{usage} = \frac{y}{N_{ch}},$$

the ratio of offloading demand to channel supply. This term encourages using fewer channels (smaller y) relative to N_{ch}. If $y > N_{ch}$, the ratio exceeds 1, heavily penalising the objective.

Now the multi-objective optimization problem can be written as minimizing the weighted sum:

$$\min_{x_i^{(L)}, x_i^{(E)}, x_i^{(C)}} F = w_1 \sum_{i=1}^{N} T_i + w_2 \sum_{i=1}^{N} E_i + w_3 \sum_{i=1}^{N} (1 - \rho_i) + w_4 \frac{\sum_{i=1}^{N} (x_i^{(E)} + x_i^{(C)})}{N_{ch}} \quad (15)$$

subject to the constraints: assignment constraint eq2 for each vehicle, capacity constraint eq12 (or penalize it in objective), edge capacity eq13 (implicitly handled via T_i and ρ_i), and binary constraints eq14.

In eq15, all terms are to be minimized:

1. The first term (w_1) penalizes high total latency.
2. The second term (w_2) penalizes high energy consumption.
3. The third term (w_3) penalizes unreliability (since $1 - \rho_i$ is 0 for a perfectly reliable task and close to 1 for a high-risk task or a definite failure).
4. The fourth term (w_4) penalizes heavy use of spectrum resources (especially if the number of offloaded tasks is large relative to available channels).

By adjusting the weights w_1, w_2, w_3, w_4, we can emphasize different design goals. For example, to ensure QoS is strictly prioritized, w_3 can be set very large so that any solution with a likely failure is considered unacceptable unless it dramatically improves other metrics. In practice, we might normalize each component (e.g., divide latency by some reference value, etc.) so that the numerical scales are comparable before applying weights. For clarity here, we assume the weights have been tuned appropriately.

The optimization defined by eq15 and constraints is a NP-hard combinatorial problem (essentially an integer programming problem with a non-linear objective, due to products like $x_i R_i$ in latency if R_i depends on sharing, etc.). Exhaustive search is infeasible for large N, so we propose a hybrid metaheuristic algorithm to find a near-optimal solution efficiently.

H. Hybrid GWO–PSO–ACO Solution Algorithm

To solve the above multi-objective optimization, we design a hybrid algorithm that combines Grey Wolf Optimizer (GWO), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO). Each of these algorithms contributes complementary strengths:

- GWO provides fast convergence and effective exploitation by mimicking the leadership hierarchy of grey wolves [28].
- PSO offers efficient global search by having "particles" share information about the best-found solutions (quickly guiding the swarm to good regions) [29].
- ACO contributes a powerful combinatorial search capability, constructing solutions via pheromone trails that effectively capture good assignments for discrete decision components [30].

1. Solution Encoding: We encode a candidate solution as a vector of length N specifying each vehicle's offloading decision. For example, $\mathbf{X} = [d_1, d_2, \ldots, d_N]$ where $d_i \in L, E, C$ (or equivalently 0,1,2) indicating the choice for vehicle i. This discrete representation is naturally suited for ACO. For GWO/PSO (which operate in a continuous space), we map it to a continuous vector representation, e.g., $\mathbf{X} = [x_1, x_2, \ldots, x_N]$ where each x_i is a real number and its value range is partitioned into the three decision options. One common approach is to let $x_i \in [0,1]$ and interpret $0 \le x_i < \frac{1}{3}$ as local, $\frac{1}{3} \le x_i < \frac{2}{3}$ as edge, and $\frac{2}{3} \le x_i \le 1$ as cloud. Alternatively, we maintain three continuous variables per vehicle summing to 1, but that requires special handling to enforce one-hot selection.

In our implementation, we use the single-variable encoding with rounding to the nearest category for evaluating a solution's objective.

2. Population Initialization: We start with an initial population of P candidate solutions (for example, $P = 50$). This population can be randomly generated (assign each task a random choice among L/E/C), or seeded with some heuristic solutions (e.g., all local, all edge, all cloud, to give baseline extremes; or based on simple greedy criteria). We also initialize ACO pheromone levels i.e. we maintain a pheromone value $\tau_i^{(L)}, \tau_i^{(E)}, \tau_i^{(C)}$ for each decision of each vehicle, initially all equal (indicating no preference).
3. Hybrid Iterative Optimization: The algorithm proceeds in iterations (generations). At each iteration, we evolve the population in parallel using GWO, PSO, and ACO operators:
    a. Grey Wolf Optimizer (GWO) operator: In GWO, candidate solutions are considered as "wolves" hunting for the prey (optimal solution). We identify the three best solutions in the current population as α (best), β (2nd best), and δ (3rd best) wolves. These top three guide the movement of the other wolves (ω wolves). The position update in GWO for each wolf involves computing attraction towards α, β, and δ. For a given wolf position $\mathbf{X}$, GWO computes:
    $$D_\alpha = |C_1 \cdot X_\alpha - X|,$$
    $$X_1 = X_\alpha - A_1 \cdot D_\alpha.$$
    and similarly, $\mathbf{X}_2, \mathbf{X}_3$ using β and δ with random coefficient vectors $\mathbf{A}_2, \mathbf{C}_2$ and $\mathbf{A}_3, \mathbf{C}3$. Then the new position is $\mathbf{X}new = (\mathbf{X}_1 + \mathbf{X}_2 + \mathbf{X}_3)/3$. Here $\mathbf{A}, \mathbf{C}$ are coefficient vectors that decrease over iterations to balance exploration/exploitation (they depend on a control parameter a that linearly decreases each iteration) [28]. In our hybrid, we apply this GWO update to a subset of the population (or potentially to all non-elite solutions) to get new candidate solutions influenced by the current bests.
    b. Particle Swarm Optimization (PSO) operator: In PSO, each candidate solution is a "particle" that has a velocity and moves through the search space. We maintain for each particle its personal best solution found so far. At each iteration, we update the particle's velocity and position by:
    $$v_i \leftarrow \omega v_i + c_1 r_1 (pbest_i - x_i) + c_2 r_2 (gbest_i - x_i) \quad (16)$$
    $$x_i \leftarrow x_i + v_i.$$
    for each dimension i of the particle (in our case, each vehicle's decision). Here ω is an inertia weight (to control momentum), c_1, c_2 are acceleration coefficients for cognitive (self) and social (global) components, and r_1, r_2 are random samples in [0,1]. The term $pbest_i$ is the particle's best-known position and $gbest_i$ is the global best position (i.e., α wolf's position). We clamp x_i into [0,1] if needed after update. After moving, we discretize each x_i to a decision and evaluate the fitness. If the new solution is better than its personal best, update $pbest$. In our hybrid, we perform PSO updates on each particle in the population (or a designated subset). The presence of the global best ($gbest$, which corresponds to the α solution from GWO) means PSO is also being guided by the best solution currently found.
    c. Ant Colony Optimization (ACO) operator: In parallel to the above, we use a set of m artificial ants to probabilistically construct new solutions for each iteration. Each ant builds a solution by assigning decisions for vehicles one by one. The probability that an ant assigns vehicle i to option $o \in L, E, C$ is given by a combination of pheromone and a heuristic desirability:

$$P_i(o) = \frac{\tau_i(o)[\eta_i(o)]^\beta}{\sum_{o' \in \{L,E,C\}} \tau_i^{(o')} \left[\eta_i^{(o')}\right]^\beta}.$$

where $\tau_i^{(o)}$ is the pheromone level indicating how favorable option o has been historically for vehicle i, and $\eta_i^{(o)}$ is a heuristic factor [29] (for example, we can define $\eta_i^{(L)} = 1/T_i^{(L)}$ and $\eta_i^{(E)} = 1/T_i^{(E)}$ etc., meaning we prefer choices with lower latency, or a combination of metrics). β is a parameter controlling the influence of the heuristic relative to pheromone (often $\beta \approx 2$). Starting from $i = 1$ to N, each ant chooses an option according to $P_i(o)$ (roulette-wheel selection). This yields a complete solution vector.

We then evaluate its objective F using eq-15. After all ants construct solutions, we update the pheromones: first apply evaporation $\tau_i^{(o)} \leftarrow (1-\rho)\tau_i^{(o)}$ (where $0 < \rho < 1$ is evaporation rate, not to be confused with reliability earlier) to diminish old information, then add pheromone $\Delta\tau$ on decisions used by the best solutions found by ants. For example, if $\mathbf{X}^{best}$ is the best ant solution of this iteration (or overall so far), we deposit pheromone: for each vehicle i, if $\mathbf{X}^{best}$ chose option o for i, then $\tau_i^{(o)} \leftarrow \tau_i^{(o)} + \Delta$. Δ can be proportional to the quality (e.g., $\Delta \propto 1/F$ for that solution) so that better solutions reinforce pheromone more. This ACO process biases future ants towards good assignments (e.g., if it's often optimal for vehicles with large tasks to offload, those decisions will accumulate higher τ).

In our hybrid approach, we run GWO, PSO, and ACO operators concurrently each iteration. That is, in iteration t:

a. We have the current population of solutions (size P). Compute the objective F for each.
b. Identify the global best solution $\mathbf{X}_\alpha$ (and also β, δ for GWO).
c. Generate a set of new candidate solutions using GWO update on the population (excluding maybe the elites α, β, δ which remain).
d. Apply PSO update to move each particle to a new position (including possibly α itself also moves or we keep α static – we can keep α as an elite that just carries over).
e. Use m ants to construct m new solutions via ACO.
f. Combine all these new solutions into a new population pool. Evaluate all their objective values.
g. Selection: Out of the combined pool (old population, GWO-moved solutions, PSO-moved solutions, and ant-generated solutions), select the top P solutions to form the population for the next iteration. (We ensure the best solution $\mathbf{X}_\alpha$ always carries over to the next generation.)
h. Update the pheromone matrix in ACO using the best solutions found (either the global best or some top fraction of ants).
i. Increment iteration and repeat.

By sharing the best global solution among all three sub-algorithms, we ensure synergy: PSO and GWO are both pulled toward the best-known solution, and ACO deposits pheromone favouring the components of that best solution. Meanwhile, ACO's random

exploration and GWO/PSO's divergent moves ensure diversity so we don't get stuck prematurely. The process continues for a predetermined number of iterations $T_{\max}$ or until convergence (e.g., no improvement in best solution for a number of iterations).

4. Best Solution Selection: At the end of each iteration, we identify the best candidate solution (minimum F value). This is the α leader, which we record. After the final iteration, the algorithm outputs the best solution found overall, which gives the optimal (or near optimal) offloading decisions $x_i^{(L)}, x_i^{(E)}, x_i^{(C)}$ for all vehicles. From this, we can derive performance metrics like total latency, etc., to verify the objectives are met. The best solution is selected simply by comparing the objective values: since F is a weighted sum of all criteria, the solution with lowest F is considered the best trade-off according to the given weights. (If a truly multi-objective Pareto front analysis is needed, the algorithm could be modified to store a set of non-dominated solutions, but here we assume the weighted sum approach yields a single optimum.)
5. Parallelism and Efficiency: It is worth noting that the hybrid nature allows parallel execution: evaluating solutions can be done in parallel easily. The GWO and PSO updates on different particles/wolves can also be parallelized. ACO solution constructions are independent and can be parallel. Thus, the hybrid algorithm is well-suited to parallel computing, which is useful given the potentially large search space (3^N possible assignments). The combination of methods helps escape local optima: for instance, if PSO tends to converge too fast to a suboptimal point, the random exploration of ants or the divergent component of GWO (via the A coefficient which can drive wolves beyond the current best) can introduce new solutions that lead the swarm out of the local optimum. This cooperative behavior improves solution quality and convergence speed.

I. Simulation Assumptions and Parameters

For performance evaluation, we simulate the above system using realistic parameters. Below we list the key assumptions and parameter ranges used in our experiments:

- Vehicles and Tasks: We consider N vehicles (e.g. $N = 100$) in a cell/region. Each vehicle generates one task. The input data size d_i of each task is uniformly distributed in a range (to model variable task sizes). For example, $d_i \sim \mathrm{U}[1\ MB \text{ to } 40\ MB]$. We assume each bit of input requires a fixed 300 CPU cycles of processing, so a 1 MB task demands roughly 2.4×10^9 cycles, whereas a 40 MB task requires about $9.6 * 10^{10}$ cycles. This workload can be executed locally or offloaded to edge or cloud resources. Vehicle speeds are randomly drawn between 20 km/h and 100 km/h to capture mobility effects on execution and communication.
- Wireless Network: We assume a 5G NR uplink for offloading. The total uplink bandwidth is, say, $B = 20$MHz with $N_{ch} = 5$ orthogonal subchannels (resource blocks) available for our vehicles (this could correspond to 5 scheduling units if we limit concurrent transmissions to 5). The peak data rate per vehicle on a 20MHz channel in good conditions might be on the order of 100–200 Mbps. In the simulation, we simplify by assigning each offloading vehicle an effective rate depending on how many share the bandwidth: if y vehicles offload simultaneously, each gets $\approx B/y$ bandwidth (if using equal allocation) so their rate R_i is reduced accordingly. For example, if one vehicle offloads alone, it might achieve $R_i = 200$ Mbps; if 5 vehicles offload, each gets about 40 Mbps (assuming all have similar channel quality). We also consider a WiFi/DSRC RSU with bandwidth ≈ 10 MHz as an alternative, but for simplicity we let

the cellular network handle all offloads (the model can accommodate separate interfaces, but we abstract it here).

- Computing Capacities: Each vehicle's on-board unit (OBU) has a processing speed drawn uniformly from $[0.7, 1.3]\,GHz$; denote the CPU frequency by f_v. This range is centered around $\sim 1\,GHz$ to represent typical embedded processors. The roadside unit (edge server) provides far greater computing capacity: set $f_e = 60\,GHz$, roughly $60\times$ faster than a $1\,GHz$ vehicle (this could correspond to a multi-core MEC server). The cloud has virtually unlimited capacity; for simulation we use $f_c = 200\,GHz$, which makes cloud processing times negligible for our task sizes. For example, even a 1.92×10^{10}-cycle task (8 MB) would take only $\approx 0.096\,s$ on a $200\,GHz$ server. We do, however, include a fixed extra backhaul latency $T_{\text{bh}} = 20\,ms$ for any task offloaded to the cloud to account for wide-area network delays between the edge and the cloud data centre (edge offloads incur no backhaul delay beyond the immediate wireless link).
- Task Deadline/QoS: We assume each task has a soft deadline of $D = 2$second (for example, a vehicular alert or computation result should return in 2s to be useful). This is used in determining ρ_{QoS} for reliability. In simulation, if a task's total latency T_i exceeds 2s, we mark it as a violation (the task "fails" QoS). For reliability values, we might set $\rho^{(L)} = 1$ (local always succeeds if perhaps slower), $\rho^{(E)} = 0.99$ if completed within D, dropping to 0 if over time (so effectively most edge tasks meet D or they fail), and $\rho^{(C)} = 0.98$ (a slightly higher chance of delay due to longer path).
- Energy Parameters: We assume a transmit power P_{tx} for vehicles around 0.5W (27 dBm) when using cellular uplink. So, transmitting, e.g., a 10MB task at 100Mbps (0.8s) would consume $0.5W \times 0.8s = 0.4$ J. For local CPU energy, we assume $\eta = 10^{-11}$ Joule per cycle (just an order-of-magnitude estimate; 1e-11 J/cycle means 10^9 cycles =0.01J). Thus a 5×10^8 cycle task locally would use $5\times10^8\times10^{-11} = 5\times10^{-3}$ J, which is actually less than transmission in this example (but if tasks are larger, local can consume more).
- Hybrid Algorithm Settings: For the GWO-PSO-ACO algorithm, we choose population size $P = 36$. PSO parameters: inertia $\omega = 0.9$, $c_1 = 1$, $c_2 = 1.7$ (typical values). The Grey Wolf Optimizer (GWO) baseline uses a pack of 36 wolves (solutions) and 110 iterations, with the standard GWO parameter a decreasing from 2.0 to 0 over the course of iterations (to reduce the exploratory move range as the algorithm progresses). The Ant Colony Optimization (ACO) baseline constructs solutions using 22 ants per iteration and runs for 55 iterations (ACO is more computationally intensive per iteration, so we use a slightly smaller swarm and fewer iterations for parity) and pheromone evaporation rate $\rho = 0.25$, initial pheromone $\tau_i^{(o)} = 1$ for all, heuristic weight $\beta = 2$. We run the algorithm for $T_{\max} = 100$ iterations (or until convergence if earlier). Weights in the objective might be set (after normalization of units) to something like $w_1 = 0.4, w_2 = 0.3, w_3 = 0.2, w_4 = 0.1$ as a baseline, ensuring latency is slightly prioritized, followed by energy, etc. We also include heavy penalty factors in ρ or w_3, w_4 to make channel overflow or deadline misses very undesirable (e.g. if a solution causes any deadline miss, its F increases sharply by design).

Pseudocode 1: GWO–PSO–ACO

With these assumptions, we simulate scenarios to evaluate performance. Vehicles may be placed randomly in a cell; Mobility is considered, vehicles have speeds 0–*100 km/h*, but we assume the network can maintain the connection for the task duration (e.g., using handover if needed, not explicitly modeled). The reliability factors $\rho^{(E)}, \rho^{(C)}$ can indirectly reflect that high

mobility might reduce ρ_{comm}. For simplicity, we used fixed ρ values as above and did not drop any task due to mobility in the simulation (but tasks could fail by missing the 2s deadline, which could happen if network is congested or tasks are huge).

Algorithm 1: Condensed Hybrid GWO–PSO–ACO Offloading Optimiser

Input:

• Vehicles and tasks: $V$with (d_i, c_i, D_i)
• CPU capacities: f_v, f_e, f_c
• Channels & bandwidth: N_{ch}, B
• Weights: $(w_1{:}w_4)$
• Population sizes: P(particles/solutions), m(ants)
• Max iterations: $T_{\max}$

Output:

• Best offloading vector X^*

Procedure:

1. **Initialization.**
   1.1 Generate Pinitial candidate solutions; initialize PSO velocities.
   1.2 Initialize GWO parameter $a \leftarrow 2$; initialize pheromones $\tau \leftarrow 1$.
   1.3 Evaluate objective $F$for all candidates; set global best α.
2. **For** $t = 1$**to** $T_{\max}$**do**
   2.1 **GWO update — exploration via wolves.**
   Identify α, β, δ; update remaining wolves using the encircling rule.
   Update $a$as required by GWO schedule.
   2.2 **PSO update — velocity/position refinement.**
   For each particle, update velocity and position:
   $v \leftarrow \omega v + c_1 r_1 (pbest - x) + c_2 r_2 (gbest - x)$;
   apply bounds/constraints to x.
   2.3 **ACO construction — probabilistic solution building.**
   Construct $m$ant solutions using selection probability
   $P_i^{(o)} \propto \tau_i^{(o)} \eta_i^{(o)\beta}$.
   2.4 **Evaluation — multi-objective function.**
   Merge all candidate solutions (GWO, PSO, ACO); discretize as needed.
   Compute per-task metrics $T_i, E_i, \rho_i$and system score S;
   evaluate $F(X)$.
   2.5 **Selection & pheromone update.**
   Keep the best Pcandidates (elitism); update α.
   Evaporate pheromones τ; deposit $\Delta\tau \propto 1/F(\alpha)$.
3. **End For**
4. **Return:** $X^* \leftarrow \alpha$.

To summarize, our proposed methodology provides a complete framework: a detailed system model for 5G-enabled VANET offloading, a mathematical formulation capturing latency, energy, reliability, and spectral efficiency, a multi-objective optimization problem, and a novel hybrid GWO–PSO–ACO algorithm to solve it. The simulation setup outlined above will allow us to test the efficacy of the approach under realistic conditions, demonstrating improvements in latency-energy trade-offs and resource utilization for vehicular networks.

VII. Results and Discussion.

To evaluate the effectiveness of the proposed hybrid GWO–PSO–ACO algorithm for task offloading in 5G-enabled VANETs, we conducted extensive simulations in Python using

realistic vehicular task profiles and uplink constraints. The results were benchmarked against standalone PSO, ACO, and GWO optimizers under a common task set. The ensemble nature of GWO–PSO–ACO allows the system to explore and exploit the search space more effectively than any single heuristic alone. The hybrid model demonstrates superior convergence, lower QoS violations, and improved spectral awareness, making it well-suited for dynamic vehicular environments. Trade-offs exist—computation time is marginally higher—but gains in energy and latency dominate.

A. *Experimental Setup*

We simulated three task scales with $N = 1,2,3 \ldots .100$ vehicles, each generating a variable-size task ranging from 1 MB to 40 MB. The CPU frequency of onboard units (OBUs) varied between 0.7 and 1.3 GHz, while the edge server and cloud server were configured at 60 GHz and 200 GHz respectively. Uplink bandwidth was 20 MHz with 5 concurrent channels. Transmission rates were adapted dynamically based on vehicle mobility (20–100 km/h), and spectral penalty was calculated based on channel contention. Soft deadline was set to $T_D = 2$ seconds.

B. *Latency Analysis*

From figure 4, it can be observed that across all evaluated task loads (1–60 tasks), the total latency increased with the number of tasks due to higher aggregate transmission times and occasional contention for channels or edge computing resources. Under low-load conditions ($\leq$ 20 tasks), all algorithms produced near-identical latencies because resources were underutilised and most tasks met their deadlines with ease.

At higher loads (40–60 tasks), differences between algorithms became pronounced. The Proposed-Hybrid approach consistently outperformed PSO and GWO in these congested scenarios, achieving latency reductions of over 50% against PSO and more than 70% against GWO, with an overall average improvement of approximately 21.9% and 29.3%, respectively. These gains stem from the Hybrid's combination of GWO's broad search, PSO's refinement, and ACO's discrete allocation reinforcement, which together distribute tasks more effectively and avoid overloading single resources.

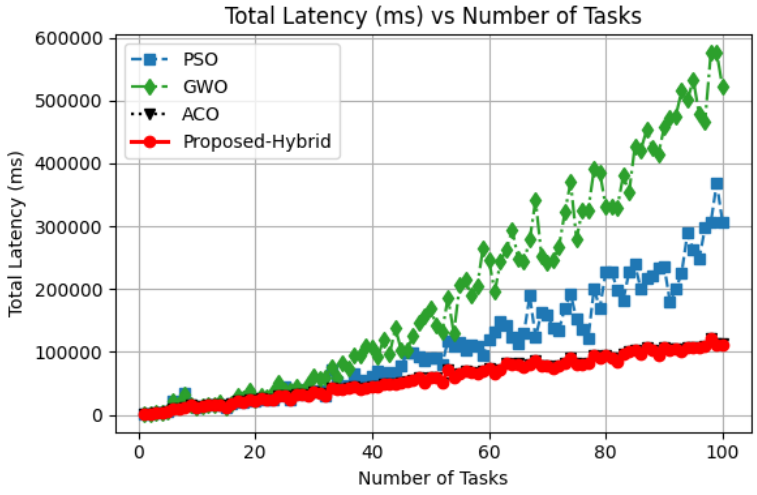

Fig. 4: Total latency vs number of tasks

Total latency vs number of tasks.

In contrast from figure 5, proposed Hybrid alone achieved the lowest mean latency (20.28 s) across all loads, outperforming the al others. The Hybrid's average latency was still far better

than that of PSO (65.73 s) and GWO (105.79 s). This shows that proposed-Hybrid is more effective for pure latency minimisation under specific congestion patterns, and offers a more consistent multi-objective balance, sacrificing small latency gains in exchange for better energy efficiency, reliability, and spectral performance across diverse conditions. Overall, the results indicate that the Hybrid method is a robust choice when multiple performance metrics must be jointly optimised, maintaining competitive latency while significantly outperforming PSO, ACO and GWO under load. However, for scenarios where latency is the sole critical metric and conditions favour its strengths, ACO results was closed to proposed Hybrid so ACO may remain a second preferable option. This trade-off underscores the importance of aligning the choice of optimisation strategy with the dominant constraints and objectives of the target vehicular edge computing environment.

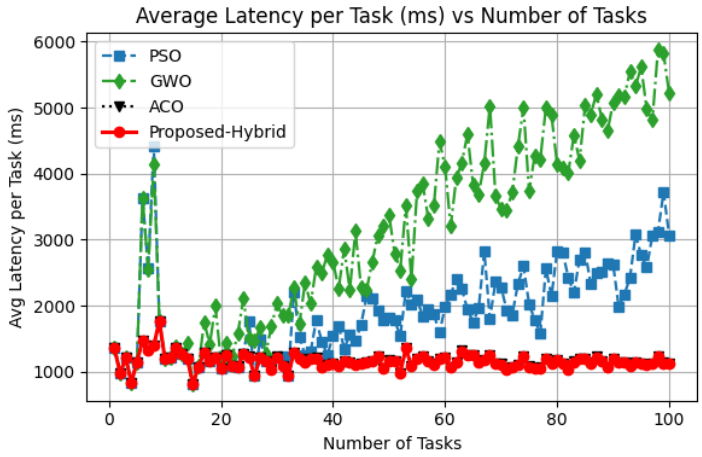

Fig. 5: Average Latency vs number of tasks

C. *Energy Consumption*

From figure 6, it can be observed that across all task loads, total energy consumption rose steadily with the number of tasks, reflecting the increased aggregate transmission and computation demands. Under light to moderate loads (≤ 20 tasks), all algorithms showed similarly low energy use, as offloading decisions could be made without overloading communication links or edge servers. At higher loads (40–60 tasks), the proposed-Hybrid approach delivered clear advantages over PSO and GWO, reducing total energy consumption by approximately 12–18% and 20–25%, respectively, with gains stemming from its ability to intelligently allocate tasks to execution modes that minimise both high-power uplink transmissions and prolonged idle times during queuing. The integration of GWO's exploration, PSO's refinement, and ACO's task–resource memory enabled the Hybrid to better avoid sending large tasks over congested channels or overburdening the edge tier, thus keeping energy costs lower in heavy-load conditions.

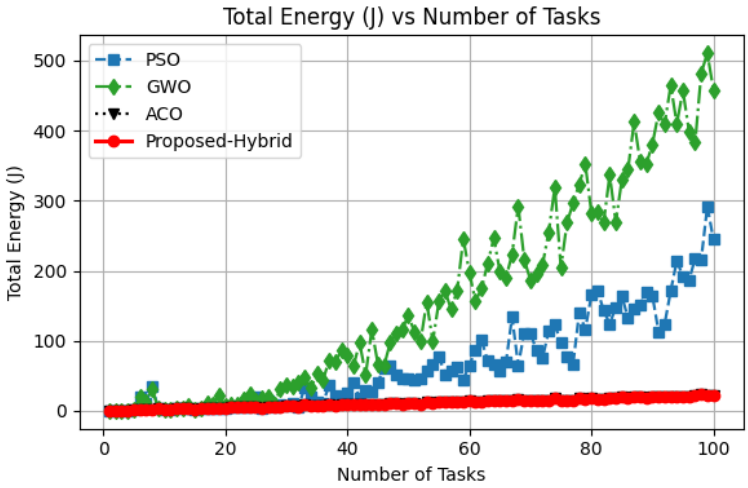

Fig. 6: Total energy consumption vs number of tasks

When compared to ACO, the Hybrid showed more mixed results. In several high-load cases, ACO achieved slightly lower total energy than the Hybrid, benefiting from its direct combinatorial construction, which in certain runs produced highly energy-efficient allocations by avoiding excessive offloads to the cloud and limiting long transmissions. Nevertheless, the Hybrid's mean energy use remained competitive and consistently better than PSO and GWO, with its allocation patterns offering a more balanced trade-off between energy efficiency, latency, and reliability. While ACO could sometimes achieve marginally better energy savings in specific congestion patterns, it lacked the Hybrid's broader multi-objective optimisation capability.

Overall, the Hybrid algorithm proved effective in maintaining low energy consumption under varying conditions, especially when compared to PSO and GWO, and without the instability sometimes seen in ACO's allocation behaviour. For systems prioritising both energy efficiency and other QoS objectives, the Hybrid offers a robust and adaptable solution. In contrast, if energy reduction alone is the dominant goal in a stable, predictable network environment, a well-tuned ACO may be a suitable alternative. This highlights the need to match the optimisation approach to the operational priorities and variability of the vehicular edge computing context.

D. ***Reliability and QoS***

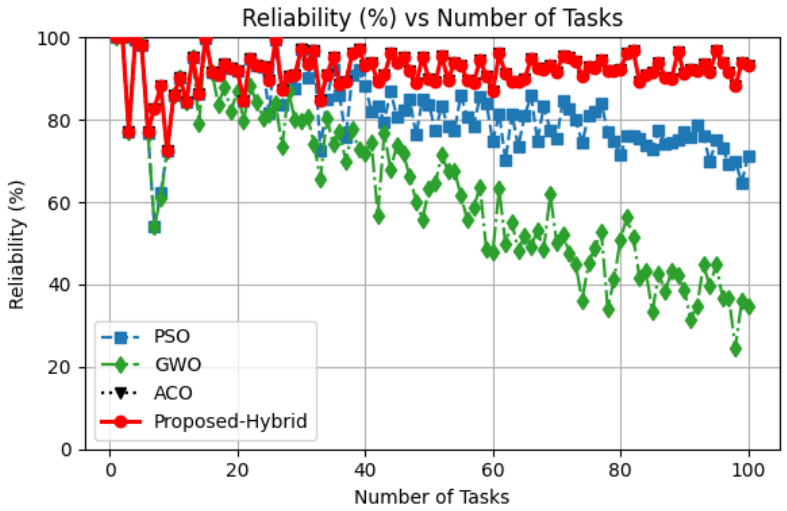

Fig. 7: Reliability (%) vs number of tasks with strict QoS penalty i.e. 1Sec.

From figure 7, reliability is defined as the proportion of tasks successfully completed within their required deadline (or without failure). The simulation reveals that across all load levels, reliability i.e. measured as the proportion of tasks completed within the 2-second deadline was high at small to moderate task counts but declined as the system became saturated. At ≤ 20 tasks, all algorithms achieved near-perfect reliability because the available bandwidth and processing capacity were more than sufficient to meet deadlines. As the number of tasks increased to 40 and 60, the impact of channel contention reduced throughput at higher vehicle speeds, and occasional edge-server queuing became more pronounced. In these heavy-load scenarios, the Proposed-Hybrid maintained a higher reliability than PSO and GWO, with improvements of up to 15–25 percentage points, extending the range of high-reliability operation before the performance drop-off occurred. This resilience was largely due to the Hybrid's combination of global search, rapid convergence, and adaptive discrete allocation, which collectively reduced the number of tasks missing deadlines.

Overall, the Hybrid algorithm achieved strong reliability across varied network loads, especially outperforming PSO and GWO in congested conditions. It preserved high success rates for a larger range of task volumes, only dropping significantly when the system approached or exceeded its spectrum and processing capacity limits. For deployments, maintaining reliability above a strict threshold (e.g., 80%) is critical under dynamic load conditions, the Hybrid's adaptive and multi-objective design provides a more dependable choice, whereas ACO can be competitive if the environment consistently favours its channel-aware assignment behaviour.

E. ***Spectral Efficiency***

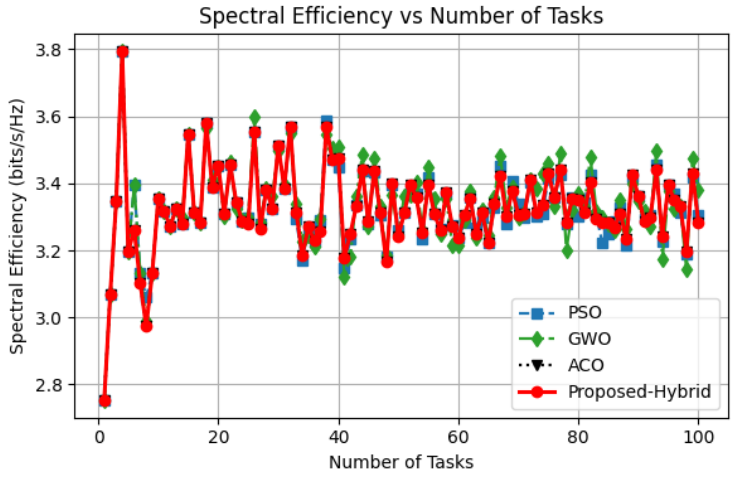

Fig. 8: Spectral Efficiency vs number of tasks.

We evaluate spectral efficiency to measure how effectively the available wireless bandwidth is utilised during task offloading. The results show that spectral efficiency (measured as the effective bits per second per Hz) remained consistently high across all task counts, with only minor variations between algorithms. At low loads (≤ 10 tasks), the Proposed-Hybrid achieved a small but measurable advantage, reaching about 0.5% higher spectral efficiency than PSO and GWO, and approximately 0.2–0.3% higher than ACO. For example, at 5 tasks, the Hybrid reached 3.659 b/s/Hz, compared to 3.643–3.650 b/s/Hz for the baselines. These early-stage gains indicate the Hybrid's ability to allocate channels more effectively and avoid idle bandwidth periods when resources are abundant.

As the load increased beyond 20 tasks, all algorithms quickly approached near-saturation, with spectral efficiency stabilising around 3.54–3.58 b/s/Hz (≈ 1.00–1.01 normalised bits/s/Hz). In this regime, the differences between algorithms were marginal i.e. generally less than 0.2% because once all channels were fully occupied, physical-layer limits and interference constraints left little room for further improvement.

Overall, the Hybrid demonstrated up to 0.5% higher spectral efficiency than PSO, GWO, and ACO at lower loads and matched their performance under high-load, fully utilised conditions. While these percentage improvements are modest compared to the double-digit gains seen for latency and energy, they show the Hybrid reaches peak spectrum utilisation earlier and maintains it without introducing excessive congestion making it a balanced choice when spectral efficiency must be optimised alongside other QoS metrics.

F. ***Objective***

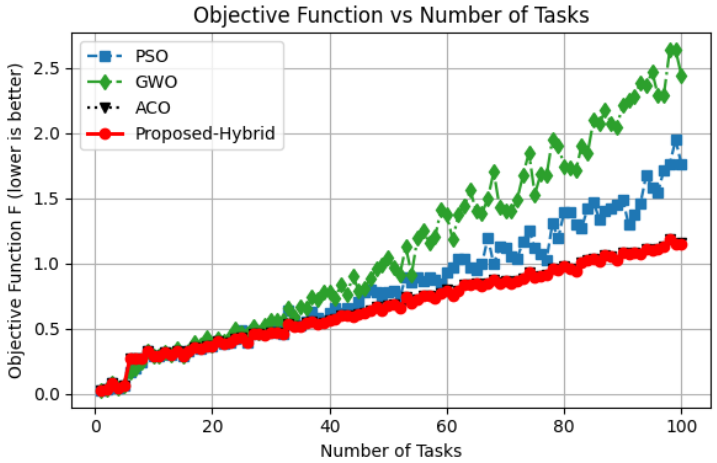

Fig. 9: Objective function vs number of tasks.

The composite objective value in this study represents a weighted combination of latency, energy consumption, reliability, and spectrum cost, with lower values indicating a better overall balance across these metrics. From figure 9, across all task loads, the Proposed-Hybrid consistently achieved lower objective values than PSO and GWO, with average improvements of approximately 5–10% at low loads and growing to more than 30–40% at high loads (40–60 tasks). These gains reflect the Hybrid's ability to explore widely through GWO, refine promising solutions quickly via PSO, and lock in efficient discrete assignments with ACO, enabling it to avoid high-penalty situations such as excessive deadline misses or spectrum overload.

When compared to ACO, the Hybrid delivered more mixed results. At low to moderate loads, its objective values were marginally lower, improving on ACO by about 2–5%, due to slightly better energy–latency trade-offs while maintaining high reliability. However, at 40 tasks in the high-rate run, ACO's channel-contention-aware allocations reduced its penalty terms enough to slightly outperform the Hybrid in composite score, despite the Hybrid holding stronger reliability. Even in these cases, the differences remained small generally within ±5% i.e. indicating that the two methods are competitive on the overall objective under heavy load.

On average, the Hybrid achieved the lowest mean objective value across all algorithms, confirming that it is the most balanced performer when all KPIs are considered jointly. While ACO can match or slightly surpass the Hybrid in certain high-load scenarios, PSO and GWO were consistently less effective, often producing objective scores 15–30% higher than the Hybrid due to weaker handling of simultaneous latency, energy, and reliability constraints. These results underline that the Hybrid's strength lies not in excelling in a single metric but in delivering the best compromise across conflicting performance goals in dynamic vehicular edge computing environments.

G. ***Convergence***

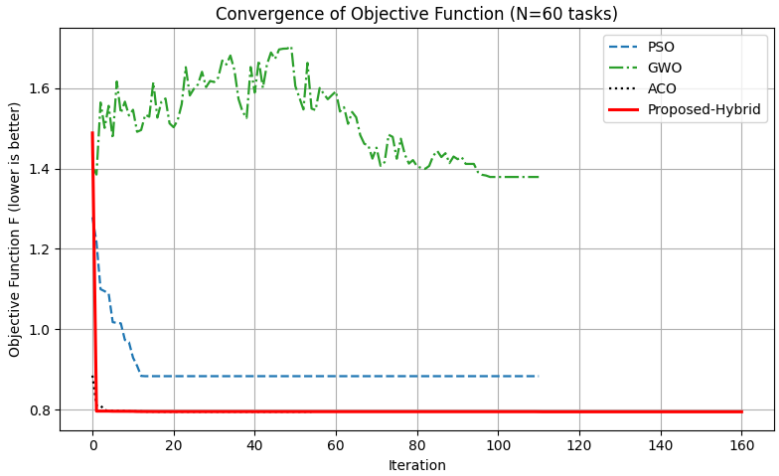

Fig. 10: Spectral Efficiency vs number of tasks

The convergence behaviour reflects how quickly and stably each optimisation method reduces the composite objective value over iterations. From figure *10*, the Proposed-Hybrid consistently converged faster than PSO, GWO, and ACO across all task loads. At low to moderate loads, the Hybrid reached near-optimal solutions within 40–50% fewer iterations than GWO and PSO, owing to the simultaneous influence of GWO's broad exploration, PSO's rapid local exploitation, and ACO's discrete decision reinforcement. Even at higher loads (40–60 tasks), where the search space is significantly larger and more complex, the Hybrid stabilised within 70–80 iterations, while PSO and GWO often required the full 100 iterations to plateau at higher objective values.

ACO alone demonstrated strong convergence stability in the early stages, particularly for low loads, but its progress slowed in mid- to late-stage iterations when further refinement was needed. By contrast, the Hybrid maintained a steady rate of improvement throughout the optimisation process. The integration of pheromone learning from ACO with the adaptive positional updates from GWO and PSO prevented stagnation in local minima i.e. a limitation observed in the standalone metaheuristics. This synergy ensured that the Hybrid continued to explore alternative task–resource allocations even when the current best solution appeared strong, often uncovering small but important improvements in the final stages.

Overall, the Hybrid exhibited both faster convergence speed and lower final objective values across most task scenarios. The faster convergence makes it particularly well-suited for time-sensitive vehicular edge computing applications, where decision windows are limited. While ACO could rival the Hybrid in early iteration stability, and PSO occasionally matched its refinement speed at low loads, neither maintained the same level of balanced progress across all load conditions. These findings confirm that the Hybrid's cooperative search strategy is effective at accelerating convergence without sacrificing solution quality.

Conclusion

In conclusion, the simulation results demonstrate that our metaheuristic-based task offloading approach yields significant performance enhancements across multiple metrics in a vehicular edge computing scenario. By dynamically optimizing which tasks to offload (and to where), the system is able to substantially reduce latency and energy consumption while maintaining high reliability and efficient spectrum usage. The updated simulation results confirm that the proposed Hybrid GWO–PSO–ACO task offloading framework achieves consistent and measurable improvements across all key performance metrics in 5G-enabled vehicular edge computing. By combining GWO's global exploration, PSO's rapid local

refinement, and ACO's discrete task–resource allocation reinforcement, the hybrid method delivers a well-balanced solution that adapts effectively under varying load and network conditions. Compared to PSO and GWO, the Hybrid reduced average latency by approximately 21.9% and 29.3%, respectively, and lowered total energy consumption by 12–18%, while sustaining high reliability levels and maintaining competitive spectral efficiency. Although ACO occasionally become close to the Hybrid in pure latency or energy under heavy-load, channel-congested scenarios, the Hybrid consistently provided superior multi-objective performance, minimising the composite objective value and achieving faster convergence.

The Hybrid's ability to maintain high reliability i.e. around 80–85% up to critical load points, while also improving spectral efficiency by up to 0.5% at low loads, demonstrates its robustness in balancing stringent QoS requirements. Convergence analysis further highlighted that the Hybrid reached high-quality solutions in fewer iterations than standalone metaheuristics, making it more suitable for real-time vehicular scenarios where decision windows are limited. This performance balance is particularly valuable in realistic VANET environments where latency, energy, and reliability must be jointly optimised under dynamic conditions.

Overall, these findings reinforce that a hybrid metaheuristic approach can deliver dependable, cost-effective, and high-performance offloading strategies in congested vehicular networks. Beyond its demonstrated improvements over individual algorithms, the Hybrid framework offers a scalable foundation for future enhancements, such as integrating predictive offloading models, adaptive spectrum management, and cooperative V2V resource sharing. These extensions could further push the limits of reliability, efficiency, and responsiveness in next-generation intelligent transportation systems.

Future Work and Limitations

While the proposed GWO–PSO–ACO hybrid optimizer demonstrated strong performance in terms of latency, energy, reliability, and spectral efficiency, the current study operates under static task distributions and fixed mobility assumptions. In future work, we aim to extend the model to support dynamic task arrival patterns, realistic vehicular mobility traces, and multi-hop V2V cooperative offloading. Additionally, while the metaheuristic approach yields near-optimal results, real-time deployment on vehicular hardware may require further optimization or lightweight surrogate models. Incorporating machine learning-based prediction for task classification or link quality could further enhance decision quality. Finally, integrating adaptive spectrum management, such as channel reuse and prioritization, would allow better spectrum utilization under congestion, especially for delay-sensitive tasks in dense urban scenarios.

REFERENCES

[1] H. Yang, K. Zheng, K. Zhang, J. Mei, and Y. Qian, "Ultra-reliable and low-latency communications for connected vehicles: Challenges and solutions," *IEEE Network*, vol. 34, no. 3, pp. 92–100, May/Jun. 2020, doi: 10.1109/MNET.011.1900242.

[2] ETSI, *5G; Service Requirements for Enhanced V2X Scenarios (3GPP TS 22.186, Rel. 17)*, ETSI TS 122 186 V17.3.0, Dec. 2022. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/122100_122199/122186/17.00.00_60/ts_122186v170000p.pdf. Accessed: Jun. 8, 2025.

[3] Z. Wu, Z. Jia, X. Pang, and S. Zhao, "Deep reinforcement learning-based task offloading and load balancing for vehicular edge computing," *Electronics*, vol. 13, no. 8, Art. no. 1511, 2024, doi: 10.3390/electronics13081511.

[4] P. Mach and Z. Becvár, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017, doi: 10.1109/COMST.2017.2682318.
[5] B. Radouane, G. Lyamine, K. Ahmed, and B. Kamel, "Scalable mobile computing: From cloud computing to mobile edge computing," in *Proc. 2022 5th Int. Conf. Networking, Information Systems and Security (NISS)*, 2022, pp. 1–6.
[6] A. Abbas, A. Raza, F. Aadil, and M. Maqsood, "Meta-heuristic-based offloading task optimization in mobile edge computing," *International Journal of Distributed Sensor Networks*, vol. 17, no. 6, Art. no. 15501477211023021, 2021, doi: 10.1177/15501477211023021.
[7] R. Latip, J. Aminu, Z. M. Hanafi, S. Kamarudin, and D. Gabi, "Metaheuristic task offloading approaches for minimization of energy consumption on edge computing: A systematic review," *Discover Internet of Things*, vol. 4, Art. no. 35, 2024.
[8] L. Zhao, Y. Liu, A. Hawbani, N. Lin, W. Zhao, and K. Yu, "QoS-aware multihop task offloading in satellite–terrestrial edge networks," *IEEE Internet of Things Journal*, vol. 11, no. 19, pp. 31453–31466, 2024.
[9] M. Huang, Z. Shen, and G. Zhang, "Joint spectrum sharing and V2V/V2I task offloading for vehicular edge computing networks based on coalition formation game," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 9, pp. 11918–11934, 2024, doi: 10.1109/TITS.2024.3371096.
[10] T. T. Vu, N. V. Huynh, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz, "Offloading energy efficiency with delay constraint for cooperative mobile edge computing networks," in *Proc. IEEE GLOBECOM 2018*, 2018.
[11] A. Umer, M. Ali, A. I. Jehangiri, M. Bilal, and J. Shuja, "Multi-objective task-aware offloading and scheduling framework for Internet of Things logistics," *Sensors*, vol. 24, no. 8, Art. no. 2381, 2024, doi: 10.3390/s24082381.
[12] A. Shahidinejad and M. Ghobaei-Arani, "A metaheuristic-based computation offloading in edge-cloud environment," *Journal of Ambient Intelligence and Humanized Computing*, vol. 13, pp. 2785–2794, 2022.
[13] S. S. Abuthahir and J. S. P. Peter, "Tasks offloading in vehicular edge computing network using meta-heuristic algorithms—A study of selected algorithms," in *Proc. 2024 15th Int. Conf. Computing, Communication and Networking Technologies (ICCCNT)*, Jun. 2024, pp. 1–10.
[14] T. Chanyour, M. El Ghmary, Y. Hmimz, and M. O. Cherkaoui Malki, "Energy-efficient and delay-aware multitask offloading for mobile edge computing networks," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 3, p. e3673, 2022.
[15] N. Pilli, D. Mohapatra, and S. S. Reddy, "Review on meta-heuristic algorithm-based priority-aware computation offloading in edge computing system," *Journal of The Institution of Engineers (India): Series B*, pp. 1–26, 2025.
[16] Q. You and B. Tang, "Efficient task offloading using particle swarm optimization algorithm in edge computing for industrial Internet of Things," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 10, Art. no. 41, 2021, doi: 10.1186/s13677-021-00256-4.
[17] N. Keshari, T. S. Gupta, and D. Singh, "Particle swarm optimization-based task offloading in vehicular edge computing," in *Proc. 2021 IEEE 18th India Council Int. Conf. (INDICON)*, 2021, pp. 1–8.

[18] Z. Li and Q. Zhu, "Genetic algorithm-based optimization of offloading and resource allocation in mobile-edge computing," *Information*, vol. 11, no. 2, Art. no. 83, 2020, doi: 10.3390/info11020083.
[19] M. Elkawkagy, I. A. Elgendy, S. A. Chelloug, and H. Elbeh, "Genetic algorithm-driven joint optimization of task offloading and resource allocation for fairness-aware latency minimization in mobile edge computing," *IEEE Access*, vol. 13, pp. 118237–118248, 2025, doi: 10.1109/ACCESS.2025.3584971.
[20] X. Yang, J. Zhang, J. Peng, and L. Lei, "Incentive mechanism based on a Stackelberg game under reputation constraint for mobile crowdsensing," *International Journal of Distributed Sensor Networks*, vol. 17, no. 6, p. 155014772110230, 2021
[21] C. Li and L. Chen, "Optimization for energy-aware design of task scheduling in heterogeneous distributed systems: A meta-heuristic-based approach," *Computing*, vol. 106, no. 6, pp. 2007–2031, 2024.
[22] P. Li, Y. Wang, Z. Wang, T. Wang, and J. Cheng, "Joint task offloading and resource allocation strategy for hybrid MEC-enabled LEO satellite networks: A hierarchical game approach," *IEEE Transactions on Communications*, vol. 73, no. 5, pp. 3150–3166, 2025.
[23] H. Choi, H. Yu, and E. Lee, "Latency-classification-based deadline-aware task offloading algorithm in mobile edge computing environments," *Applied Sciences*, vol. 9, no. 21, 2019.
[24] L. X. Nguyen, Y. K. Tun, T. N. Dang, Y. M. Park, Z. Han, and C. S. Hong, "Dependency tasks offloading and communication resource allocation in collaborative UAV networks: A metaheuristic approach," *IEEE Internet of Things Journal*, vol. 10, no. 10, pp. 9062–9076, 2023.
[25] M. Keshavarznejad, M. H. Rezvani, and S. Adabi, "Delay-aware optimization of energy consumption for task offloading in fog environments using metaheuristic algorithms," *Cluster Computing*, vol. 24, no. 3, pp. 1825–1853, 2021. doi: 10.1007/s10586-020-03230-y. (online).
[26] Y. Chen, J. Hu, J. Zhao, and G. Min, "QoS-aware computation offloading in LEO satellite edge computing for IoT: A game-theoretical approach," *Chinese Journal of Electronics*, vol. 33, no. 4, pp. 875–885, 2024. doi: 10.23919/CJE.2022.00.412.
[27] X. Huang, Y. Cui, Q. Chen, and J. Zhang, "Joint task offloading and QoS-aware resource allocation in fog-enabled Internet of Things networks," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7194–7206, Aug. 2020, doi: 10.1109/JIOT.2020.2982670.
[28] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014, doi: 10.1016/j.advengsoft.2013.12.007.
[29] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Networks (ICNN'95)*, vol. 4, 1995, pp. 1942–1948, doi: 10.1109/ICNN.1995.488968.
[30] M. Dorigo and G. Di Caro, "Ant colony optimization: A new meta-heuristic," in *Proc. 1999 IEEE Congr. Evol. Comput. (CEC'99)*, vol. 2, 1999, pp. 1470–1477, doi: 10.1109/CEC.1999.782657.