FMDB *Transactions on Sustainable Intelligent Networks*

An Improved ARO Model for Task Offloading in Vehicular Cloud Computing in VANET

R. Mohan Das[1,*], M. Arunadevi Thirumalraj[2], T. Rajesh[3], S. Gopikha[4], Sureshkumar Somayajula[5]

[1]Department of Electrical and Electronic Engineering, New Horizon College of Engineering, Bengaluru, Karnataka, India.
[2]Department of Computer Science and Engineering, Karunya Institute of Technology and Sciences, Coimbatore, Tamil Nadu, India.
[2]Department of Computer Science and Business Management, Saranathan College of Engineering, Tiruchirappalli, Tamil Nadu, India.
[3]Department of Electrical and Electronic Engineering, Malla Reddy Engineering College, Hyderabad, Telangana, India.
[4]Department of Information Technology, St.Joseph's College of Engineering, Chennai, Tamil Nadu, India.
[5]Department of Computer Science and Technology, Sunlife Canada Financials, Toronto, Ontario, Canada.
mohandas@newhorizonindia.edu[1], aruna.devi96@gmail.com[2], rajeshpradha@gmail.com[3], gopikha.re@gmail.com[4], suresh.kumar.somayajula@sunlife.com[5]

Abstract: Vehicle-to-vehicle (V2V) communication enables a network of automobiles to perform collaborative computing, giving rise to the concept of a "vehicular cloud" (VC). However, without the need for edge nodes or cloud servers, vehicles can run applications that require massive processing cooperatively on their own by forming a Vehicular Ad-Hoc Network (VANET). Managing the recurrent topology alteration caused by vehicle mobility is a significant challenge for VANET cooperative computing. In this research, researchers present a V2V-based cooperative computing approach. The suggested method accounts for the distance between vehicles when selecting which vehicles to collaborate with, and it defers task offloading until the last possible moment to ensure a stable, energy-efficient cooperative computing environment. Despite its competitive performance compared to other MH algorithms, the artificial rabbits' optimisation (ARO) algorithm still suffers from poor accuracy and the risk of local optima. Using antagonism methods, this research develops a selective opposition version of the artificial rabbit procedure (LARO) that eliminates the negative consequences of these shortcomings. To begin, during the random concealment phase, a Lévy flight strategy is implemented to increase population diversity and dynamics. The algorithm's convergence accuracy is enhanced by the richness of its various population samples.

Keywords: Vehicular Cloud; Network Reliability; Lévy Flight; Artificial Rabbits Optimisation; Vehicular Ad-Hoc Network; Cooperative Computing Method; Vehicle-to-Vehicle.

Received on: 16/11/2024, **Revised on:** 24/01/2025, **Accepted on:** 01/03/2025, **Published on:** 05/09/2025

Journal Homepage: https://www.fmdbpub.com/user/journals/details/FTSIN

DOI: https://doi.org/10.69888/FTSIN.2025.000536

Cite as: R. M. Das, M. A. Thirumalraj, T. Rajesh, S. Gopikha, and S. Somayajula, "An Improved ARO Model for Task Offloading in Vehicular Cloud Computing in VANET," *FMDB Transactions on Sustainable Intelligent Networks*, vol. 2, no. 3, pp. 137–151, 2025.

1. Introduction

[*]Corresponding author.

The Internet of Vehicle Things (IoV) is a new approach to expanding the practical uses of vehicles [1]. Health care, package delivery, electronic transportation, advertising, route planning, autonomous vehicles, automobiles, virtual reality games, augmented reality, public surveillance, and 3D video games are just some of the potential uses [2]. However, due to resource constraints (such as battery life, processing power, and network throughput), smart devices cannot run compute-intensive IoT applications locally. For Internet of Things (IoT) applications that require a constant wireless connection, Vehicular Fog Cloud Network (VFCN) is a possible alternative [3]. Data security and ubiquitous app services are also handled by VFCN [4]. Future "smart cities" will rely heavily on their vehicle networks. Intelligent networking in road transportation systems improves traffic management, security, and in-car entertainment [5]. Smart cities of the future will have better air quality thanks to improved traffic management, which also reduces carbon emissions and the number of accidents [6]. Further, vehicular networks will improve route navigation apps, predictive car maintenance, and information and entertainment services [7]. On-board units (OBU) located in each vehicle, roadside units (RSU) strategically placed across a city, and traffic command centres (TCC) make up a vehicular network [8]. OBUs, or on-board units, are the wireless transceivers in vehicles that send and receive data of all kinds.

For cars to communicate with one another, they need on-board units (OBUs) [9]. Roadside units (RSUs) are strategically placed along roadways so that moving vehicles are always within range of at least one RSU. For traffic data collection and analysis, as well as for offering entertainment services to automobiles, these RSUs are retroactively connected to the TCC via the Internet cloud [10]. Intelligent transportation systems (ITS) are enabled by wireless technologies such as cellular communications, which are integrated into vehicular communication. ITS provides a wide variety of cutting-edge services, including driverless cars, video games, and augmented and virtual realities. The majority of these services require advanced computing capacity with no tolerance for delay [11]. In addition, it is anticipated that automobiles will increasingly support both high-traffic applications and simultaneous multitasking in the not-too-distant future. Vehicles need powerful computational and battery resources to address these issues. Although these problems can be overcome by shifting computationally intensive operations and data traffic to the Internet cloud, doing so may result in unacceptable delays for ITS systems that are particularly sensitive to latency [12]. Computing servers are placed at each RSU site and linked together so that each RSU may act as a fog node, mitigating the impact of latency on time-critical operations [22]. These fog nodes provide substantial storage for cached data and processing power for computationally intensive activities [13].

However, due to the high speed and restricted coverage area, cars equipped with RSUs have shorter connectivity times, which is a significant limitation for job computation. Furthermore, a fog node cannot perform all tasks concurrently due to the high number of task types. This slows the execution of some jobs that aren't necessary across diverse applications, potentially significantly impacting how well vehicular networks function. Task scheduling, which determines the optimal execution sequence, is another essential component for improving the effectiveness of task offloading [14]. The amount of data being sent, the cost of that transfer, and the total completion time for all jobs are all determined by the task scheduling. The majority of prior research has focused on reducing this execution time. However, researchers construct our job scheduling approach with both execution time and energy costs in mind. For electric cars in particular, energy costs are crucial because they affect range [15]. The primary goal is to plan task offloading using optimisation models. However, the experiments show that optimisation based on artificial rabbits has low convergence accuracy and tends to get stuck in local solutions when dealing with complex or high-latitude problems. In light of the foregoing, this work proposes an upgraded ARO algorithm (LARO) approach. The ARO procedure has a variation called LARO. First, the Lévy flight approach is widely used to enhance ARO's global discovery capabilities. LARO can better plan to avoid local solutions and explore foreign options thanks to the Lévy flight method. Second, an enhanced convergence-accuracy selective opposition technique allows for local LARO exploitation. Below, researchers list the paper's novel and important contributions:

- The ARO is further enhanced by the use of the Lévy flying strategy during the random concealment phase to increase population variety and dynamics.
- The algorithm's ability to escape local optima is enhanced by introducing a selection-based opposition strategy that builds on the original opposition strategy and re-adaptively updates it.

2. Related Works

A cooperative computing approach using V2V communication is proposed by Gong et al. [16]. The suggested technique accounts for the distance between vehicles when selecting which vehicles to collaborate with, and it defers task offloading until the last feasible moment to maintain a stable environment. In comparison to traditional static scheduling techniques, the results are in terms of both energy savings and network reliability. However, SCOCC's speedup increase is 9% lower than the others. SCOCC does not outperform static algorithms like HEFT and GBTSA in execution time, since it does not attempt to optimise the execution speed of entire jobs at once. However, SCOCC provides crucial stability in a rapidly moving vehicle and minimizes the energy required for wireless communication. Despite these difficulties, Taha et al. [17] have developed a strategy for achieving data secrecy on (CP-ABE). To perform CP-ABE tasks, VANET nodes form a vehicle cluster. To manage these

dispersed microtasks, researchers utilise a container orchestration framework to construct a vehicle cluster(s). In this approach, researchers use a set of variables that affect the computational processes in the OBU of a cluster vehicle. There is a corresponding weight for each component that affects computing processes and the cluster. Each factor's weight is determined using the Euclidean approach. Our method divides the work amongst vehicles based on their final combined weight. Researchers gauge the efficacy of our method by comparing it with Kubernetes's workload-allocation mechanism, which considers only the available resources in each car. In addition to utilising simulations to assess our method's effectiveness in terms of tran, researchers also consider a range of scenarios with varied parameters to analyse on OBUs. To intelligently meet the processing needs of vehicle applications, Haris et al. [18] have suggested a VANET architecture based on Intelligent Volunteer Computing. Researchers provide criteria for choosing volunteers whose computers can complete the computationally demanding assignment. In this research, researchers apply a procedure to accurately select the volunteer vehicle for job execution by predicting the competence of various cars. The computing power of the best volunteer cars is estimated based on extensive experimentation. Researchers compared the effectiveness of nine distinct regression methods using free, downloadable datasets.

The outcomes demonstrate the effectiveness of these methods in predicting volunteers' performance. Results from a comparison of regression methods show that ridge regression and support vector regression are superior at reducing MSE, RAE, and RMSE. The suggested approach is compared to the current one through simulations. Rashid et al. [19] have presented a machine learning-based approach for detecting rogue nodes in real time. A set of GBT, LR, MLPC, and SVM replicas was used to test the performance of our projected classifier in OMNET++ and SUMO. The suggested model is applied to a dataset comprising both typical automobiles and those used in attacks. The simulation findings improve the assault categorisation to 99% accuracy, a significant improvement. The system achieved 94% and 97% accuracy with LR and SVM, respectively. With 98% and 97% accuracy, respectively, the RF and GBT performed admirably. Because the training and testing periods do not scale proportionally with the number of network nodes, the network's performance has improved since researchers began using Amazon Web Services. Cars in (VANETs) can make use of a technique proposed by Gong et al. [20] termed vehicular adaptive offloading (VAO) to shift the burden of computationally demanding activities to neighbouring cars. This is accomplished by accounting for changes in VANETs. Overall performance in CO is significantly affected by task scheduling. To reduce the likelihood of task reallocations due to connection interruptions between cars in VANETs, it uses a directed acyclic graph (DAG) to express the priority relationships between jobs before assigning them to neighbours. The simulation results showed that compared to Max-Min, the suggested technique reduces the number of reallocations by 45.4%. This results in an average 14.4 percentage-point reduction in the duration of all processes within a single application. Avoiding potentially dangerous interactions with hostile vehicle users is made easier using the batch authentication and key exchange approach proposed by Poongodi et al. [21].

A Public Key Infrastructure (PKI), an ID-based system, and a MAC-based system are also proposed. VANET security ratings were predicted using the neuro-fuzzy inference method. For safe data transmission, the Chain model is employed. This study also analysed the paper from a blockchain and MEC standpoint. The tiers: services. The first layer of the blockchain ensures the safety of VANET data during transmission. The perception layer leverages edge computing and distributed cloud services. Data at the service layer is secured by both conventional cloud storage and blockchain systems. The throughput and quality-of-service needs of MEC users are addressed at the foundational level of the system design. The fundamental problem is reaching agreement among blockchain nodes without negatively impacting the efficiency of the MEC system or the blockchain. A Markov process with a reward function is used to model the joint optimisation issue. The simulation outcomes are presented to prove the study's claims. To ensure stability and reliability in (VANET) installations, offer a strategy based on Differential Evolution (MOTD-DE). To pick cars that meet the cluster algorithm's requirements, researchers use Kubernetes containers as the cluster. As a result, researchers can perform intricate operations on the data owner's automobiles. When a vehicle in our cluster joins the group, its data will be made available to the vehicle, and a deep learning model will determine the fit complexity of individual jobs. To cut down on execution time and resources (vehicles), the proposed MOTD-DE divides up jobs into smaller chunks and assigns them to groups of vehicles. Researchers also assume the jobs are decoupled. Researchers present scenarios in which the number of jobs, cars, CPUs, and memory, as well as the number of vehicles, are varied to assess the efficacy of our work. MOTD-DE is superior to four other popular methods, including the Particle Swarm algorithm, according to a summary of evaluation results.

2.1. Problem Description

With its reliable wireless connectivity and pervasive cloud services, Vehicular Cloud Network promotes mobility-aware apps. Users of apps will often incur expenses for both data transfer and processing time. Therefore, the VCN needs to provide low-cost services to user applications while they are in motion with as little latency as possible. Current VCNs provide cloud services on resource-intensive machines; these are prohibitively expensive and do not support mobile devices. In the section that follows, researchers detail our proposal for a new virtual private network (VCN) that leverages containerised microservices.

3. Proposed System

All of the cars discussed in this paper are electric and have GPS and transmission devices capable of direct V2V communication as well as cellular connectivity, allowing them to perform the functions of other vehicles as requested by VM technology. A CV initiates computation offloading by first making a direct V2V communication request to other cars within the transmission radius, then selecting a WV based on information from those vehicles. Researchers get into the specifics of how to find and prioritise WVs among the vehicles that meet your requirements later on. The tasks that make up an application appc of CV v_c are the smallest logical units that may still be further broken down, and these tasks are shown as nodes in a directed acyclic graph. The NP-completeness of the problem of assigning each job to a vehicle significantly contributes to the overall execution time. By default, a direct V2V connection is used to transmit data and to determine the outcome of task offloading. However, there are situations where the distance between a CV and a WV exceeds the V2V communication range, and the outcome of the offloaded job must be delivered over cellular networks. In this work, appc represents a time-sensitive application with strict requirements. Taking into account the processing speed and data transfer time of the nearby vehicles, CV v_c then sends a request to the relevant WV v_j to carry out task t_i of appc. The time and energy required to finish an application depend heavily on the task scheduling that decides which WV will be allocated to it.

3.1. System Description

A CV v_c will attempt to delegate application appc duties to other CVs. v_c looks for vehicles in the area, selects WVs, and then asks them to carry out the DAG-expressed duties of app_c. It takes, on average, as many instructions from ipbi to complete task t_i, where i is the number of bytes of output data. Therefore, D_o(t_i) ipb_i instructions are required if the data produced by task t_i is D_o(t_i) bytes. The computer capability of each vehicle is measured in instructions per second (ips), while the rate of information exchange between cars is represented in bytes per second (bps). To learn more about the cars in its vicinity, v_c first broadcasts a message requesting details about them. The vehicle that is willing to share its computing resources with v_c responds to this message with its current position, its movement status, and the computing resources it can share with v_c. All vehicles prioritise their own tasks; the resources left over are what's left once those jobs are completed. The next step is for v_c to use the gathered data to determine which cars should do which jobs. Since the amount of energy used and the time required to complete the application both depend on how jobs are scheduled, it's important to find an allocation strategy that minimizes energy use while still meeting the application's deadline T. Due to the channel's wireless nature, data broadcast overhead is crucial in a vehicle network. It takes energy to relay data across wireless channels to other cars. There is no point in offloading jobs if the cost of doing so, including data transmission, exceeds the benefit.

3.2. Problem Definition

A DAG $G = (T, E)$ is accepted to perfect errands of an application. app_c. T is a usual of nodes sense tasks ti's of app_c And E is called a source task. This work focuses on directed acyclic graphs (DAGs) with a single source job. However, the suggested technique is not limited in its applicability; DAGs with many source tasks can be readily transformed into DAGs with a single task by simply adding a void job with tasks. If there is a dividing line from t_i to t_l, t_i is called an instant predecessor of t_l. Conversel. After CV v_c brands a set V^w Of WVs with some 1-hop that reply to its broadcast communication, each job in T is allocated to a WV in V^w. The binary mutable x_{ij} designates that task t_i is allocated to WV v_j:

$$x_{ij} = \begin{cases} 1 & \text{if } t_i \text{ is allocated to } v_j \text{ by } v_c \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

X is a set of x_{ij}, which signifies a job map transmission errand in T to WVs in V^w:

$$\sum_j x_{ij} = 1 \quad (2)$$

v_j receives the allocated task via a V2V message and executes it. v_j can perform it after receiving the consequences of the t_i's i-predecessors. When the implementation of t_i Is completed, the subsequent data of size $Do(t_i)$ must be conveyed to the vehicle's execution t_i's i-successors. The execution t_i in v_j, signified by w_{ij}, is to be treated for t_i, $Do(t_i) \times ipb_i$, divided by the sum of orders that can be treated by v_j each second, ips_j:

$$w_{ij} = \frac{D_o(t_i) \times ipb_i}{ipb_j} \quad (3)$$

And the predictable surface period of t_i can be intended as the entirety of the preliminary time and processing period in v_j:

$$FT_i = ST_i + w_{ij} \quad (4)$$

Before t_i may begin t_i -Predecessors must have finished and reported their findings to v_j. Additionally, all tasks assigned to V_i With a higher priority than t_i, it must be finished before t_i may be started. In this context, "latest finish time." FT_l Refers to the latest moment when all of its predecessors have finished their operation and transferred the result to v_j:

$$LFT_i = \max_{t_l \in pred(t_i)} \left\{ FT_l + \frac{D_o(t_l)}{b} \right\}, \quad \forall i \in \{1,2,\dots.,n_w\} \quad (5)$$

where $pred(t_i)$ is a set of t_is, and b is the data broadcast speed among vehicles. A DAG already represents tasks scheduled for completion by a WV. A forerunner task must be finished before its successor may begin. When all of v_j's tasks with a greater priority than t_i have been finished, v_j is free to work on t_i. AT_{ij}:

$$AT_{ij} = \max_{l \in \{1,2,\dots.,n_t\}} \{x_{lj} FT_l\} \quad (6)$$

The initial start period when t_i can run on v_j, EST_{ij}, is the latter of LFT_i and AT_{ij}:

$$EST_{ij} = \max\{LFT_i, AT_{ij}\} \quad (7)$$

The initial finish time of t_i in v_j, EFT_{ij}, is as shadows:

$$EFT_{ij} = EST_{ij} + w_{ij} \quad (8)$$

tr_i is the period it takes to transmit the consequence of t_i from v_j to WVs having t_i's i-successors, which can be obtained as shadows:

$$tr_i = \frac{D_o(t_i)}{b} \quad (9)$$

For t_i to be executed in v_jTo ensure that all higher-priority jobs are finished, the starting time, ST_i, for t_i must satisfy the inequality below:

$$AT_{ij} \leq ST_i \quad (10)$$

In app_c To be finished before T, all errands must be finished before T:

$$\max_{i \in \{1,2,..,n_t\}} \{FT_i\} \leq T \quad (11)$$

The energy E_{total} spent by v_j for t_i is the quantity of the energy E_{exe} to achieve the task t_i and the energy E_{tr} to convey with t_i' i-successors:

$$E_{toatal} = E_{exe} + E_{tr} \quad (12)$$

The majority of a wireless network's power usage is often devoted to data transmission. Similarly, assuming that processors in all cars require the same amount of power to execute an instruction, mapping produces a greater change in data transmission energy than in task execution energy in our vehicular clouds. This makes sense, as state-of-the-art CPUs are likely to be installed in electric cars, and, as with desktop and laptop computers, only a select few companies develop these chips. Therefore, it makes little difference where tasks are executed, and this research centres on the energy expended during data transmission. Researchers use the energy perfect from Liu's research to determine the transmission energy E_tr. The energy required to send M bytes is detailed in the publication, among v_j and v_k is given as:

$$E_{amp\_jk} = M \times \varepsilon fs \times d_{jk}^2 \quad (13)$$

Here, εfs is the power required to send a single byte over space, where d_jk is the separation between two points in the universe, denoted by v_j and v_k. Therefore, the total energy required to transfer data from cars performing predecessor activities to successor duties may be calculated as follows across all tasks in an application:

$$E_{tr} = \sum E_{amp} = \varepsilon fs \sum_i \sum_{t_l \in succ(t_i)} \{D_o(t_i) \sum_j \sum_k x_{ij} x_{lk} d_{jk}^2\} \quad (14)$$

where $succ(t_i)$ is a set of t_iThe results of collaborative computing are communicated directly between vehicles via V2V networks. The problem is that by the time a WV completes its job operation, it is frequently no longer in close enough proximity to the vehicles that should receive the outcome to do so via direct connection. In this scenario, cellular networks are used to transfer information. The predicted data broadcast cost may be calculated as follows, where pout is the likelihood that $v_{v\,j}$ With TI, it is out of the direct communiqué variety at the time. t_j is finished, and ns(t_i) is the sum of vehicles of t_j:

$$\overline{E}[C_{tr}] = \{p_{out} c_{ce} + (1 - p_{out}) c_{di}\} \sum_i D_0(t_i) n_s(t_i) \quad (15)$$

where c_{ce} Is the cost of cellular networks and CDI the cost of communication? Because using a cellular network is more expensive per transmission, the total cost rises as use increases. Researchers factor in the potential cost of using the cellular network by accounting for the fees paid to network operators. In contrast to cellular communication, DSRC saves time and energy while transmitting data at a lower cost. Since WiFi and LTE are the foundations for contemporary DSRC and C-V2X, respectively, WiFi has a 60% greater energy efficiency than LTE [23]. Additionally, DSRC and C-V2X have transmission periods of 0.4 ms and 1 ms, respectively. As a result, researchers contend that direct DSRC connection is preferable to cellular communication for compute offloading whenever available. Therefore, researchers need to locate a task mapping X that, given (10) and (11), minimises E[C_tr]. It is a form of optimisation with the following possible ends in mind:

$$\begin{aligned} &\text{minimize} \quad E_{tr} \\ &\text{minimize} \quad \overline{E}[C_{tr}] \\ &\text{s.t (10), (11)} \end{aligned} \quad (16)$$

To minimise the above equation, the research proposes an improved ARO model.

3.3. Basic Steps of Artificial Rabbits Optimisation (ARO)

Two natural rules of rabbit survival—detour hiding—serve as the basis for the ARO algorithm's proposal [24]. One such tactic is called "detour foraging," in which rabbits eat grass around their nests to avoid being spotted by predators. As part of their technique of "random hiding," rabbits frequently hop to different burrows. The initialisation procedure is crucial to the launch of any search algorithm. The size of the artificial rabbit colony is a design variable with dimension d, where N is the value, and the limits are ub and lb. Then shadows:

$$\vec{z}_{i,k} = r.(ub_k - lb_k) + lb_k, k = 1,2,\dots,d \quad (17)$$

where $\vec{z}_{i,k}$ Signifies the location of the ith rabbit's jth dimension, and r is a chance sum that is also provided. The metaheuristic algorithm considers both exploration and exploitation, although detour foraging focuses on the latter. To find enough to eat, individual rabbits may often wander away from the group and forage in a different part of the territory. Below is the most recent iteration of the detour foraging formula:

$$\vec{v}_i(t+1) = \vec{z}_j(t) + R.\left(\vec{z}_i(t) - \vec{z}_j(t)\right) + \text{round}\left(0.5.(0.05 + r_1)\right).n_1 \quad (18)$$

$$R = l.C \quad (19)$$

$$l = \left(e - e^{\left(\frac{t-1}{T_{max}}\right)^2}\right).\sin(2\pi r_2) \quad (20)$$

$$C(k) = \begin{cases} 1 \text{ if } k == G(l) \\ 0 \text{ else} \end{cases} lk = 1,\dots,d \text{ and } l = 1,\dots,[r_3.d] \quad (21)$$

$$G = randp(d) \quad (22)$$

$$n_1 \sim N(0,1) \quad (23)$$

where $\vec{v}_{i,k}(t+1)$ denotes the rabbit, $i,j = 1,\dots,N.\vec{z}_i$ denotes the rabbit, and $\vec{z}_j$ Stands in for phony bunnies in several other locations. The maximum number of cycles is denoted by T_(max). Rounding to the closest integer is represented by the [] symbol, while the randp symbol denotes a random permutation of integers from 1 to d. r_1, r_2, and r_3 are stochastic numbers

distributed uniformly between 0 and 1. L signifies the running length; this is the rate of travel when on a foraging detour. The distribution of n1 is normal. The random number n_1 from a normal distribution best represents the disturbance. The final term in Equation (18) can be perturbed to help ARO avoid local extrema and conduct a global search. The algorithm's random hiding strategy is inspired by the exploring phase, when rabbits dig many burrows near their nests and pick one at random to hide in to avoid predators. First, researchers provide a working definition of how rabbits create their own burrows. When the ith rabbit digs the jth hole, it is called:

$$\vec{b}_{i,j}(t) = \vec{z}_i(t) + H.g.\vec{z}_i(t) \quad (24)$$

$$H = \frac{T_{max}-t+1}{T_{max}}.n_2 \quad (25)$$

$$n_2 \sim N(0,1) \quad (26)$$

$$g(k) = \begin{cases} 1 \text{ if } k == j \\ 0 \quad \text{else} \end{cases} \; l\, k = 1, \ldots, d \quad (27)$$

where $i = 1, \ldots, N$ and $j = 1, \ldots, d$, and n_2 shadows the distribution. H signifies the hidden limit that reduces linearly from 1 to $1/T_{max}$ Perturbations. Figure 1 demonstrates the change in the value over 1000 iterations. The lessening trend in H values shown in the image indicates a steady progression as the repetitions increase.

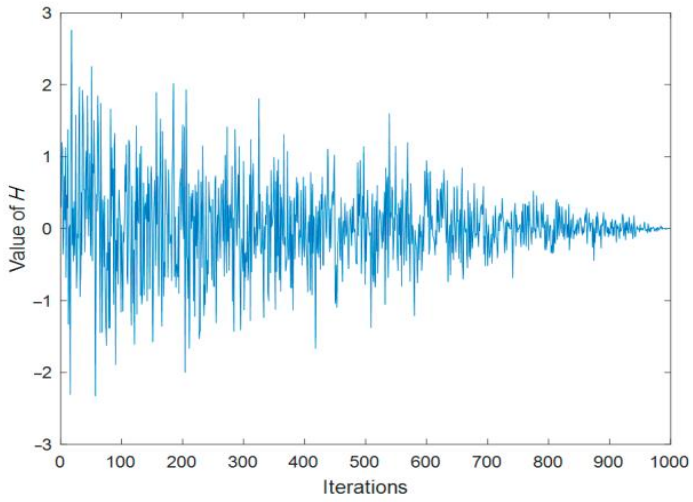

Figure 1: The alteration of H over the course of 1000 repetitions

Below is the formula for updating the random concealment technique:

$$\vec{v}_i(t+1) = \vec{z}_i(t) + R.(r_4.\vec{b}_{i,r}(t) - \vec{z}_i(t)) \quad (28)$$

$$g_r(k) = \begin{cases} 1 \text{ if } k == [r_5.d] \\ 0 \quad \text{else} \end{cases} \; lk = 1, \ldots., d \quad (29)$$

$$\vec{b}_{i,r}(t) = \vec{z}_i(t) + H.g_r.\vec{z}_i(t) \quad (30)$$

Equation (31) is used to update the ith artificial rabbit's site after the two update procedures have been applied:

$$\vec{z}_i(t+1) = \begin{cases} \vec{z}_i(t) \text{ if } f(\vec{z}_i(t)) \le f(\vec{v}_i(t+1)) \\ \vec{v}_i(t+1) \text{ else } f(\vec{z}_i(t)) > f(\vec{v}_i(t+1)) \end{cases} \quad (31)$$

An adaptive update is shown here as an equation. Based on the adaptation value, the rabbit will decide whether to remain in its current location or search for a new one. In an optimisation algorithm, populations often choose to explore early and optimise

later. ARO uses the rabbits' dwindling vitality to devise a search strategy that mimics the discovery-to-exploitation cycle. In the algorithm of the synthetic rabbit, researchers define the energy factor as:

$$A(t) = 4.\left(1 - \frac{t}{T_{max}}\right).\ln\frac{1}{r} \quad (32)$$

Where (r, r) are two random numbers between zero and one. By analysing the data shown in the image, researchers see that A decreases overall, ensuring a smooth transition as the iterations proceed.

3.4. Hybrid Artificial Rabbits Optimization

Since hybrid optimisation procedures result from targeted changes to the original method that improve the algorithm's performance, they are widely used in practical engineering. When applied to wavefront shaping, for instance, Liu's novel hybrid method combines particle swarm optimisation with a single-layer neural network to leverage the strengths of both approaches [25]. For the clustered vehicle routing problem, Islam uses particle swarm optimisation (PSO) and variable neighbourhood search (VNS), with PSO combining the solutions and VNS guiding them to local optima [26]. Devarapalli and Bhattacharyya [27] proposed a cosine-based approach to address the challenge of power system stabiliser parameter adjustment. To compensate for the ARO algorithm's subpar accuracy and tendency to quickly descend into local optima, researchers propose a hybrid, improved LARO algorithm that incorporates it into engineering optimisation problems. A Lévy flight is one such method that may be used to boost the procedure's correctness. Using the selected opposition method, the algorithm is freed from the confines of local minima.

3.4.1. Lévy Flight Method

Regular random numbers, generated by the Lévy flight operator, are small in most cases and large in a few, and are thus often used to add energy to the algorithms' updates. This law for generating random numbers can aid in introducing dynamism into update techniques and help them escape from stagnant local minima. The following equation describes the Lévy distribution:

$$\text{Levy}(t) \sim u = t^{-1-\gamma}, 0 < \gamma \leq 2 \quad (33)$$

Where t is the step size and is determined by (34), equations (34)-(37) provide the formulae for determining the step size of a Lévy flight:

$$t = \frac{u}{|v|^{1/\gamma}} \quad (34)$$

$$u \sim N(0, \sigma_u^2), v \sim N(0, \sigma_v^2) \quad (35)$$

$$\sigma_u = \left(\frac{\Gamma(1+\beta).\sin(\pi.{}^{\beta}/_{2})}{\Gamma((1+\beta).\beta.2^{(\beta-1)/2}}\right)^{1/\beta} \quad (36)$$

$$\sigma_v = 1 \quad (37)$$

where σ_u and σ_v They are distinct as assumed in Equations (36) and (37). Both u and v obey mean zero and variance σ_u^2 and σ_v^2, as exposed in Equation (35). Γ signifies a standard Gamma purpose, while b means a set to 1.5. In the random concealment phase, researchers use the random statistics generated by the Lévy flight method in place of the r4 random numbers. To prevent ARO from getting stuck on local candidate solutions during the exploitation phase, which includes the random concealment step, researchers incorporate Lévy flights into this technique. The algorithm's convergence accuracy is enhanced, and its random concealing step is made more adaptable. The Lévy flight's hidden phase is given by the following equation, where a = 0.1:

$$\vec{v}_i(t+1) = \vec{z}_i(t) + R.\left(a.\,\text{levy}(\beta).\vec{b}_{i,r}(t) - \vec{z}_i(t)\right), i = 1, \ldots, N \quad (38)$$

3.4.2. Selective Opposition (SO) Strategy

Based on the concept of opposition-based learning (OBL), SO was developed [28]. The goal of SO is to use novel opposition-based learning to shrink the scope of rabbits far from the ideal solution, thereby bringing them closer to the optimal rabbit size. Furthermore, a linearly declining threshold is typically associated with the selective opposition technique. By altering the closeness dimension of various bunnies, selected opposition helps the rabbits improve their condition throughout growth when they use SO [29]. What follows is an update. Researchers start by settling on a cutoff point. Until the limit case is achieved, the

threshold value will be lowered. The following equation shows how SO determines the optimal rabbit location given the current rabbit dimension and the set of candidate rabbit locations:

$$dd_i = \left| z_{ibest,j} - z_{i,j} \right| \quad (39)$$

where dd_j This is how far apart each rabbit is in every direction. The distant and near rabbit locations are computed when dd_j exceeds the researchers' threshold. Next, researchers see a complete catalogue of rabbit-positional distance differences:

$$src = 1 - \frac{6.\sum_{j=1}(dd_j)^2}{dd_j.(dd_j^2 - 1)} \quad (40)$$

The src is projected mostly to measure the association between the current best rabbit position. Presumptuous that src < 0 and the far dimension (d_f) is greater than the (d_c), the rabbit's site will be efficient by Equation (41):

$$z'_{df} = lb_{df} + ub_{df} - Z_{df} \quad (41)$$

3.4.3. Detailed Implementation of LARO

ARO incorporates two changes: opposition. These tweaks improve the ARO algorithm, leading to faster convergence, greater population diversity, and higher-quality solutions. Listed below are LARO's methods in detail:

- **Step-1:** Appropriate limits for LARO are supplied: the scope of rabbit N, the dimensionality of the variable star d, the bounds ub and lb of the problematic variables, and iterations T_{Max}.
- **Step-2:** Pick many locations at random for the rabbits and assign fitness levels to each. Identify the strategically placed bunny.
- **Step-3:** The value of the energy component A may be determined using Equation (32). If A > 1, pick any rabbit at random from each set.
- **Step-4:** Using Equations (19)–(22), determine R's value. Apply the detour-foraging approach by solving Equation (18). Then, using Equation (31), determine the new rabbit position's adaptation value and update it.
- **Step-5:** If A $\leq$ 1, make a bunch of burrows at random, and then pick one at random using Equation (30). With the help of a random concealment plan based on the enhanced Lévy flying strategy of Equation (38), the rabbit's current location is constantly being updated. Equation (31) is used to determine the consistent fitness and then update the rabbit's site.
- **Step-6:** The distance of each site from the rabbit dimension to the best rabbit position is determined by Equation (39).
- **Step-7:** If dd_j > Threshold, regulate the near size d_f and count the sum of df. If $dd_j \leq$ Threshold, regulate the far DC and sum the sum of d_c. Then compute the src from the calculated dd_j by Equation (40).
- **Step-8:** If src <= 0 and $d_f > d_c$, execute Equation (41) and the rabbit's location.
- **Step-9:** The ideal result is exported if the sum of iterations exceeds the worst-case scenario.

3.4.4. In-Depth Discussion of LARO Complexity

Adding the selective opposition component to the ARO base method is the major way that LARO complexity is estimated. However, the Lévy approach adds no complexity while simplifying ARO updates. Assessing the difficulty of addressing real-world situations is challenging, but complexity calculations can help. The difficulty is related to the size of false rabbits N, d, and T_{Max}. The total difficulty of the rabbit's procedure is as shadows:

$$O(ARO) = O(1 + N + T_{Max}N + 0.5T_{Max}Nd + 0.5T_{Max}Nd) = O(T_{Max}Nd + T_{Max}N + N) \quad (42)$$

With selective opposition, every facet of every possible rabbit habitat is considered. Consequently, the LARO algorithm is complex because:

$$O(LARO) = O(2T_{Max}Nd + T_{Max}N + N) \quad (43)$$

4. Results and Discussion

Simulations were used to assess the effectiveness of the suggested model. Researchers used Veins to think about how the road actually works and how cars travel over it [30]. This simulation follows a standard architecture for virtual autonomous networks (VANETs) in which cars travel at varying speeds over a straight section of Manhattan Street. Every two seconds, a brand-new

car is produced and driven at a random speed between 25 and 30 kilometres per hour. After 5 seconds, the speed of each car is raised by 5 km/h to account for the shift in the distance between them; after another 5 seconds, the speed is decreased to its previous value. In terms of processing power, all cars fall somewhere in the middle. The maximum data transfer rate between cars is 25 Mbps. Researchers use DSRC, IEEE 802.11p, and cellular networks for the physical and connection layers. They implement CSMA/CA for collision avoidance and channel assignment. Since the DSRC data rate is up to 27 Mbps, the maximum data transfer speed between cars has been set at 25 Mbps. Due to the simulation's narrow focus, researchers don't account for congestion by having trucks dump data in any particular order; instead, researchers monitor how well different algorithms divide the work. The assignment of work to a worker car is not directly responsible for the congestion. Meanwhile, some works examine data offloading and load balancing in scenarios where many vehicles simultaneously deliver packets. In addition, researchers do not analyse scenarios that are meaningless for the following presentation measures, as is the case with most prior research on compute offloading [31].

4.1. Average Packet Delivery Ratio (PDR)

The packet reception ratio is the measure of network performance, defined as the ratio of packets received at the surface sink to packets sent from the source node:

$$PDR = \frac{\sum_{u=1}^{K} \frac{P_{ru}}{P_{lu}}}{K} \quad (44)$$

where P_{ru}, P_{lu}, and K are the number of simulation runs, the number of packets generated by the source node in the uth simulation run, and the number of packets received by the surface sink in the uth simulation run, respectively.

Table 1: PDR analysis

Models	100	200	300	400	500	600	700
GWO	0.1	0.21	0.31	0.38	0.44	0.45	0.45
Whale	0.12	0.29	0.39	0.42	0.52	0.52	0.52
Butterfly	0.16	0.34	0.44	0.51	0.66	0.73	0.73
Proposed	0.22	0.53	0.72	0.85	0.91	0.92	0.92

In Table 1 and Figure 2, the PDR Analysis is represented. In this analysis, researchers used different models, including GWO, Whale, Butterfly, and the proposed model.

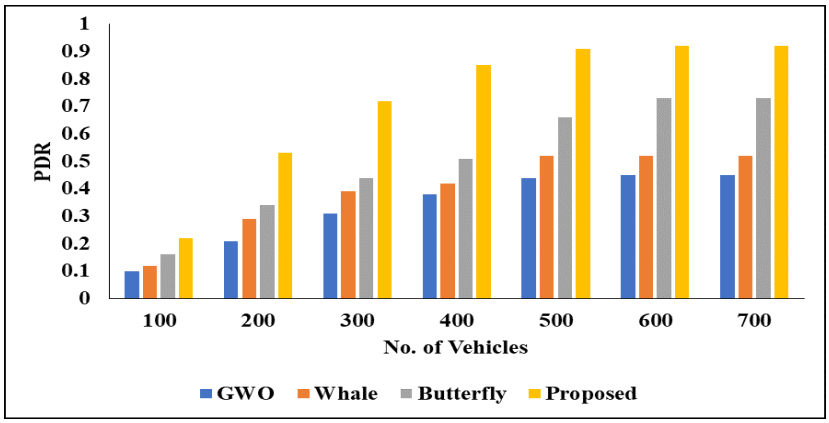

Figure 2: Graphical analysis for PDR

In the model of GWO reached the PDR value of 100 runs at 0.1 and also, the 200 runs the model reached the PDR value of 200 runs at 0.21 and also, the 300 runs the model reached the PDR value of 200 runs at 0.31 and also, the 400 runs the model reached the PDR value of 200 runs at 0.38 and also, the 500 runs the model reached the PDR value of 200 runs at 0.44 and also, the 600 runs the model reached the PDR value of 200 runs at 0.45 and also, the 700 runs the model reached the PDR value of 200 runs at 0.45 respectively. And another model of Whale reached the PDR value of 100 runs at 0.12, 200 runs at 0.29, 300 runs at 0.39, 400 runs at 0.42, and 500 runs at 0.52. Another reached the PDR value of 600 runs at 0.52, and another reached the PDR value of 700 runs at 0.52. And also, the Butterfly model in 100 runs the model reached the PDR value of 0.16 and the model in 200 runs the model reached the PDR value of 0.34 and the model in 300 runs the model reached the PDR value of

0.44 and the model in 400 runs the model reached the PDR value of 0.51 and the model in 500 runs the model reached the PDR value of 0.66 and the model in 600 runs the model reached the PDR value of 0.73 and the model in 700 runs the model reached the PDR value of 0.73 respectively. And finally, the proposed model reaches the PDR value in the runs of 100 in the PDR value as 0.22 and in the runs of 200 in the PDR value as 0.53 and in the runs of 300 in the PDR value as 0.72 and in the runs of 400 in the PDR value as 0.85 and in the runs of 500 in the PDR value as 0.91 and in the runs of 600 in the PDR value as 0.92 and in the runs of 700 in the PDR value as 0.92 respectively.

4.2. Average End-to-End Delay

The time it takes for a packet to travel from its origin node to its destination sink on the network's outermost layer is known as its end-to-end latency. Average end-to-end latency may be calculated using the formula:

$$EED = \frac{\sum_{u=1}^{K}\sum_{m=1}^{Pr}\left\{(TP_{um}-RP_{um})+T_{H_{r_j}}\right\}}{P_r K} \quad (45)$$

where P_r, RP_{um}, and TP_{um} Is the sum of the timestamps of the mth packet transmitted and received in the uth imitation run and the mth packet received in the uth imitation run, correspondingly.

Table 2: End-to-end delay (s)

Models	100	200	300	400	500	600	700
GWO	93	81	76	75	72	72	72
Whale	84	74	69	64	57	57	56
Butterfly	68	52	46	46	44	42	41
Proposed	58	43	38	36	34	34	34

Table 2 and Figure 3 present the validation of delay analysis for various models. In this analysis, researchers used different models, including GWO, Whale, Butterfly, and the proposed model. In the model of GWO reached the End-to-End delay value of 100 runs at 93 and the End-to-End delay value of 200 runs at 81 and the End-to-End delay value of 300 runs at 76 and the End-to-End delay value of 400 runs at 75 and the delay value of 500 runs at 72 and the End-to-End delay value of 600 runs at 72 and the value of 700 runs at 72 respectively.

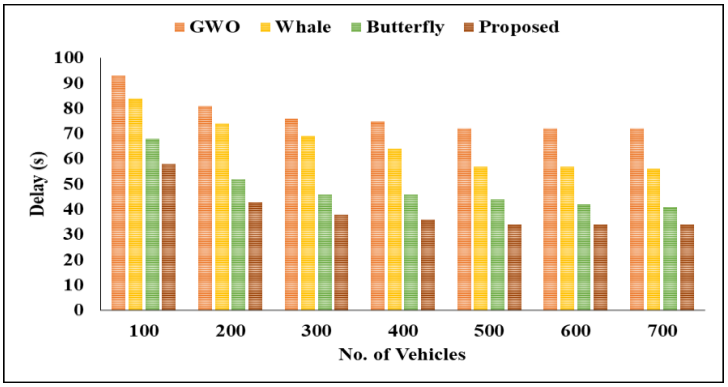

Figure 3: Validation of various models

In the model of Whale reached the End-to-End delay of 100 runs at value as 84, and the End-to-End delay of 200 runs at value as 74, and the delay value of 300 runs at value as 69, and the End-to-End delay value of 400 runs at value as 64, and the End-to-End delay value of 500 runs at value as 57 and the End-to-End delay value of 600 runs at value as 57 and the value of 700 runs at value as 56 respectively. And another Butterfly model reaches the End-to-End delay value of 100 runs at value as 68 and the End-to-End delay value of 200 runs at value as 52 and the End-to-End delay value of 300 runs at value as 46 and the End-to-End delay value of 400 runs at value as 46 and the End-to-End delay value of 500 runs at value as 44 and the End-to-End delay value of 600 runs at value as 42 and the End-to-End delay value of 700 runs at value as 41, respectively. And finally, the proposed model reaches the End-to-End delay value of 100 runs at value as 58 and reaches the End-to-End delay value of 200 runs at value as 43 and model reaches the End-to-End delay value of 300 runs at value as 38 and the End-to-End delay value of 400 runs at value as 36 and the value of 500 runs at value as 34 and the delay value of 600 runs at value as 34 and the delay value of 700 runs at value as 34 respectively.

4.3. Average Energy Consumption

The average energy used for successful transmission includes the transmission and reception of packets, as well as overhearing by nodes in the forwarding relay set. The regular network power ingesting may be determined if:

$$E_{Avg} = \frac{\sum_{u=1}^{K}\{F(i), r_j\}}{K} \quad (46)$$

Table 3: Validation of proposed model in terms of energy consumption (J)

Models	100	200	300	400	500	600	700
Proposed	32	86	120	153	190	230	260
Whale	45	110	125	160	210	240	290
Butterfly	55	125	135	185	245	265	330
GWO	86	140	156	210	260	290	340

In Table 3 and Figure 4, the Validation of the proposed model is shown in terms of Energy Consumption (J). In the Projected model reaches the Energy at the 100 runs in value of 32 and also reaches the Energy Consumption at the 200 runs in value of 86 and also reaches the Energy Consumption at the 300 runs in value of 120 and also reaches the Energy Consumption at the 400 runs in value of 153 and also reaches the Energy Consumption at the 500 runs in value of 190 and also reaches the Energy Consumption at the 600 runs in value of 230 and also reaches the Energy Consumption at the 700 runs in value of 260 respectively.

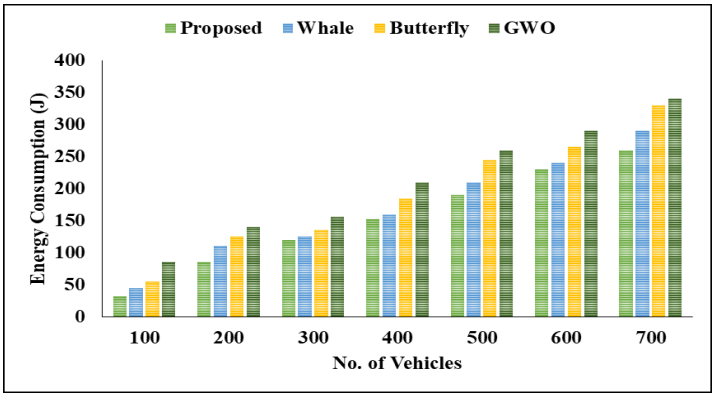

Figure 4: Graphical representation of various models

Another model of Whale reaches the Energy Consumption at the 700 runs in value of 45 and also reaches the Energy Consumption at the 700 runs in value of 110 and also reaches the Energy Consumption at the 700 runs in value of 125 and also reaches the Energy Consumption at the 700 runs in value of 160 and also reaches the Energy Consumption at the 700 runs in value of 210 and also reaches the Energy Consumption at the 700 runs in value of 240 and also reaches the Energy Consumption at the 700 runs in value of 290 respectively. Another model of Butterfly reaches the Energy Consumption at the 100 runs in value of 55 and also reaches the Energy Consumption at the 200 runs in value of 125 and also reaches the Energy Consumption at the 300 runs in value of 135 and also reaches the Energy Consumption at the 400 runs in value of 185 and also reaches the Energy Consumption at the 500 runs in value of 245 and also reaches the Energy Consumption at the 600 runs in value of 265 and also reaches the Energy Consumption at the 700 runs in value of 330 respectively. Another model of GWO reaches the Energy Consumption at the 100 runs in value of 86 and also reaches the Energy Consumption at the 200 runs in value of 140 and also reaches the Energy Consumption at the 300 runs in value of 156 and also reaches the Energy Consumption at the 400 runs in value of 210 and also reaches the Energy Consumption at the 500 runs in value of 260 and also reaches the Energy Consumption at the 600 runs in value of 290 and also reaches the Energy Consumption at the 700 runs in value of 340 respectively.

4.4. Average Network Lifetime

For at least this long, the network would function normally. In this study, researchers define network lifespan as the period until the first participating sensor node in the network loses power. Therefore, researchers determined how long a network

would last by timing how long it took for the first node to run out of juice throughout the simulation. To analyse a network's expected lifespan, one may use the formula:

$$L_{Avg} = \frac{\sum_{u=1}^{K}(ST_u - FT_u)}{K} \quad (47)$$

where ST_u and FT_u Are the times at which the first node in the uth imitation run began using energy, and the moment at which the simulation began?

Table 4: Analysis based on network lifetime (s)

Models	100	200	300	400	500	600	700
GWO	1000	800	750	700	700	700	700
Whale	1200	940	820	750	750	750	750
Butterfly	1600	1400	1200	1000	900	900	900
Proposed	1800	1500	1300	1150	1050	1050	1050

In Table 4 and Figure 5, the analysis is based on network lifetime across various models. In the model of GWO reaches the network lifetime value in 100 runs as 1000 and reaches the network lifetime value in 200 runs as 800 reaches the network lifetime value in 300 runs as 750 reaches the network lifetime value in 400 runs as 700 reaches the network lifetime value in 100 runs as 700 and reaches the network lifetime value in 100 runs as 700 and reaches the network lifetime value in 100 runs as 700 respectively.

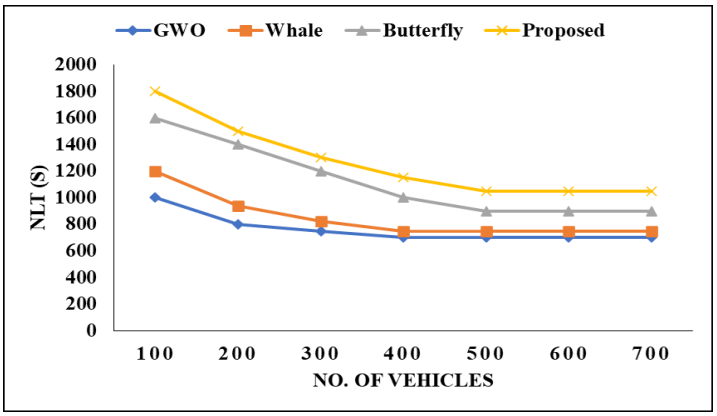

Figure 5: NLT analysis

In the model of Whale reaches the network lifetime value in 100 runs as 1200 and reaches the network lifetime value in 100 runs as 940 and reaches the network lifetime value in 100 runs as 820 and reaches the network lifetime value in 100 runs as 750 and reaches the network lifetime value in 100 runs as 750 and reaches the network lifetime value in 100 runs as 750 and reaches the network lifetime value in 100 runs as 750 respectively. In the model of Butterfly reaches the network lifetime value in 100 runs as 1600 and reaches the network lifetime value in 200 runs as 1400 and reaches the network lifetime value in 300 runs as 1200 and reaches the network lifetime value in 400 runs as 1000 and reaches the network lifetime value in 500 runs as 900 and reaches the network lifetime value in 600 runs as 900 and reaches the network lifetime value in 700 runs as 900 respectively. In the model of Proposed reaches, the network lifetime value in 100 runs as 1800 and reaches the network lifetime value in 200 runs as 1500 and reaches the network lifetime value in 300 runs as 1300 and reaches the network lifetime value in 400 runs as 1150 and reaches the network lifetime value in 500 runs as 1050 and reaches the network lifetime value in 600 runs as 1050 and reaches the network lifetime value in 700 runs as 1050 respectively.

5. Conclusion

The LARO (Location-Aware Reinforcement Offloading) model was developed to establish a reliable Vehicular Ad hoc Network (VANET) cloud that enables vehicles to offload energy and computational power efficiently. The proposed concept aims to improve system stability and reduce energy use in highly dynamic vehicular settings, where topological changes occur frequently, and mobility-related problems are common. LARO meets its goals through two main methods: (i) strategically putting off job responsibilities for as long as possible, and (ii) carefully choosing worker vehicles based on how far apart the client and worker nodes are from one another. The system ensures better offloading opportunities are available by delaying

task execution when necessary. Distance-aware worker selection also reduces communication overhead and power consumption. The reduced average distance between vehicles makes VANETs more stable, as link failures are less likely while tasks are being performed. Also, shortening the time between assigning a duty and doing it helps save energy, which makes the whole system work better.

These design choices make LARO especially well-suited for real-world driving situations where things move quickly, and connections are not always stable. The proposed LARO model does not seek to globally optimise the execution speed of all tasks simultaneously; therefore, it does not surpass static optimisation algorithms such as the Grey Wolf Optimiser (GWO), Whale Optimisation Algorithm, or Butterfly Optimisation Algorithm in terms of execution latency. Even though this is a problem, LARO has a significant advantage because it maintains stable operations even during high levels of mobility, which is very important for real-world VANET deployments. The concept also effectively reduces power consumption for wireless transmission, making it energy-efficient and scalable. As a future extension, a new serverless computing method will be investigated to further reduce the costs of storage and processing for applications. This serverless model is likely to make the VANET cloud more flexible and better utilize its resources. This study also suggests using a reinforcement learning-based approach to offloading and scheduling to address all dynamic, mobility-aware faults. This will enable adaptive, intelligent decision-making in highly dynamic vehicle environments.

Acknowledgment: The authors collectively express their sincere appreciation to New Horizon College of Engineering, Karunya Institute of Technology and Sciences, Saranathan College of Engineering, Malla Reddy Engineering College, St. Joseph's College of Engineering, and Sun Life Canada Financials for their continuous academic and professional support.

Data Availability Statement: The datasets generated and analyzed during this study are available from the corresponding authors upon reasonable request, subject to applicable ethical and confidentiality considerations.

Funding Statement: The authors confirm that this research and the preparation of the manuscript were carried out without the receipt of any external funding or financial assistance.

Conflicts of Interest Statement: The authors declare that there are no known conflicts of interest that could have influenced the outcomes or interpretations presented in this work. All sources referenced have been properly acknowledged.

Ethics and Consent Statement: This study was conducted in accordance with established ethical standards. Informed consent was obtained from all participants, and appropriate measures were taken to ensure anonymity and data confidentiality.

References

1. M. Khayyat, I. A. Elgendy, A. Muthanna, A. S. Alshahrani, S. Alharbi, and A. Koucheryavy, "Advanced deep learning-based computational offloading for multilevel vehicular edge-cloud computing networks," *IEEE Access*, vol. 8, no. 7, pp. 137052–137062, 2020.
2. S. Xu and C. Guo, "Computation offloading in a cognitive vehicular network with vehicular cloud computing and remote cloud computing," *Sensors*, vol. 20, no. 23, p. 6820, 2020.
3. S. Xu, C. Guo, R. Q. Hu, and Y. Qian, "Blockchain-inspired secure computation offloading in a vehicular cloud network," *IEEE Internet of Things Journal*, vol. 9, no. 16, pp. 14723–14740, 2021.
4. X. Li, Y. Dang, M. Aazam, X. Peng, T. Chen, and C. Chen, "Energy-efficient computation offloading in vehicular edge cloud computing," *IEEE Access*, vol. 8, no. 2, pp. 37632–37644, 2020.
5. M. LiWang, Z. Gao, S. Hosseinalipour, and H. Dai, "Multi-Task Offloading over Vehicular Clouds under Graph-based Representation," *ICC IEEE International Conference on Communications (ICC)*, Dublin, Ireland, 2020.
6. A. B. De Souza, P. A. L. Rego, P. H. G. Rocha, T. Carneiro, and J. N. De Souza, "A Task Offloading Scheme for WAVE Vehicular Clouds and 5G Mobile Edge Computing," *GLOBECOM IEEE Global Communications Conference*, Taipei, Taiwan, 2020.
7. Z. Liu, P. Dai, H. Xing, Z. Yu, and W. Zhang, "A distributed algorithm for task offloading in vehicular networks with hybrid fog/cloud computing," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 7, pp. 4388–4401, 2021.
8. A. Lakhan, M. Ahmad, M. Bilal, A. Jolfaei, and R. M. Mehmood, "Mobility aware blockchain enabled offloading and scheduling in vehicular fog cloud computing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4212–4223, 2021.
9. S. Raza, W. Liu, M. Ahmed, M. R. Anwar, M. A. Mirza, Q. Sun, and S. Wang, "An efficient task offloading scheme in vehicular edge computing," *Journal of Cloud Computing*, vol. 9, no. 1, pp. 1–14, 2020.

10. A. Shakarami, M. Ghobaei-Arani, M. Masdari, and M. Hosseinzadeh, "A survey on the computation offloading approaches in mobile edge/cloud computing environment: A stochastic-based perspective," *Journal of Grid Computing*, vol. 18, no. 4, pp. 639–671, 2020.
11. Y. Kang, Z. Liu, Q. Chen, and Y. Dai, "Joint task offloading and resource allocation strategy for DiffServ in vehicular cloud system," *Wireless Communications and Mobile Computing*, vol. 2020, no. 1, pp. 8823173, 2020.
12. J. Sun, Q. Gu, T. Zheng, P. Dong, A. Valera, and Y. Qin, "Joint optimization of computation offloading and task scheduling in vehicular edge computing networks," *IEEE Access*, vol. 8, no. 1, pp. 10466–10477, 2020.
13. L. Tang, B. Tang, L. Zhang, F. Guo, and H. He, "Joint optimization of network selection and task offloading for vehicular edge computing," *Journal of Cloud Computing*, vol. 10, no. 1, pp. 1–13, 2021.
14. H. Guo, J. Liu, J. Ren, and Y. Zhang, "Intelligent task offloading in vehicular edge computing networks," *IEEE Wireless Communications*, vol. 27, no. 4, pp. 126–132, 2020.
15. H. Li, H. Huang, and Z. Qian, "Latency-aware Batch Task Offloading for Vehicular Cloud: Maximizing Submodular Bandit," *IEEE 14th International Conference on Cloud Computing (CLOUD)*, Chicago, Illinois, United States of America, 2021.
16. M. Gong, Y. Yoo, and S. Ahn, "Vehicular cloud forming and task scheduling for energy-efficient cooperative computing," *IEEE Access*, vol. 11, no. 1, pp. 3858-3871, 2023.
17. M. B. Taha, S. Alrabaee, and K. K. R. Choo, "Efficient resource management of micro-services in VANETs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 7, pp. 6820-6835, 2023.
18. M. Haris, M. A. Shah, and C. Maple, "Internet of Intelligent Vehicles (IoIV): An intelligent VANET-based computing via predictive modelling," *IEEE Access*, vol. 11, no. 2, pp. 49665-49674, 2023.
19. K. Rashid, Y. Saeed, A. Ali, F. Jamil, R. Alkanhel, and A. Muthanna, "An adaptive real-time malicious node detection framework using machine learning in vehicular ad-hoc networks (VANETs)," *Sensors*, vol. 23, no. 5, p. 2594, 2023.
20. M. Gong, Y. Yoo, and S. Ahn, "Adaptive computation offloading with task scheduling minimizing reallocation in VANETs," *Electronics*, vol. 11, no. 7, p. 1106, 2022.
21. M. Poongodi, S. Bourouis, A. N. Ahmed, M. Vijayaragavan, K. G. S. Venkatesan, W. Alhakami, and M. Hamdi, "A novel secured multi-access edge computing based VANET with neuro fuzzy systems based blockchain framework," *Computer Communications*, vol. 192, no. 8, pp. 48–56, 2022.
22. M. B. Taha, C. Talhi, H. Ould-Slimane, S. Alrabaee, and K. K. R. Choo, "A multi-objective approach based on differential evolution and deep learning algorithms for VANETs," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 3, pp. 3035-3050, 2022.
23. L. Zou, A. Javed, and M. Gabriel-Miro, "Smart mobile device power consumption measurement for video streaming in wireless environments: WiFi vs. LTE," *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, Cagliari, Italy, 2017.
24. L. Wang, Q. Cao, Z. Zhang, S. Mirjalili, and W. Zhao, "Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems," *Engineering Applications of Artificial Intelligence*, vol. 114, no. 9, p. 105082, 2022.
25. K. Liu, H. Zhang, B. Zhang, and Q. Liu, "Hybrid optimization algorithm based on neural networks and its application in wavefront shaping," *Optics Express*, vol. 29, no. 10, pp. 15517–15527, 2021.
26. M. A. Islam, Y. Gajpal, and T. Y. ElMekkawy, "Hybrid particle swarm optimization algorithm for solving the clustered vehicle routing problem," *Applied Soft Computing*, vol. 110, no. 10, p. 107655, 2021.
27. R. Devarapalli and B. Bhattacharyya, "A hybrid modified grey wolf optimization–sine cosine algorithm-based power system stabilizer parameter tuning in a multimachine power system," *Optimal Control Applications and Methods*, vol. 41, no. 4, pp. 1143–1159, 2020.
28. T. C. Service, "A No Free Lunch theorem for multi-objective optimization," *Information Processing Letters*, vol. 110, no. 21, pp. 917–923, 2010.
29. F. Y. Arini, S. Chiewchanwattana, C. Soomlek, and K. Sunat, "Joint Opposite Selection (JOS): A premiere joint of selective leading opposition and dynamic opposite enhanced Harris' hawks optimization for solving single-objective problems," *Expert Systems with Applications*, vol. 188, no. 2, p. 116001, 2022.
30. C. Sommer, D. Eckhoff, A. Brummer, D. S. Buse, F. Hagenauer, S. Joerer, and M. Segate, "Veins: The open source vehicular network simulation framework," in Recent Advances in Network Simulation, *Springer*, Cham, Switzerland, 2019.
31. Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4377–4387, 2018.