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

Mohan Das R
New Horizon College of Engineering
Arunadevi Thirumalraj (aruna.devi96@gmail.com)
K. Ramakrishnan College of Technology
Rajesh T
Malla Reddy Engineering College

Short Report

Keywords: Vehicular Cloud, Artificial rabbits&rsquo; optimization, Vehicular Ad-Hoc Network, Cooperative computing method, L&eacute;vy flight

Posted Date: August 31st, 2023

DOI: https://doi.org/10.21203/rs.3.rs-3291507/v1

Additional Declarations: No competing interests reported.

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

[1]Dr.R.Mohan Das [2]Arunadevi Thirumalraj and T.Rajesh,

[1]Associate Professor, Department of Electrical and Electronic Engineering, New Horizon College of Engineering Bengaluru, Karnataka

[2]Department of Computer Science Engineering, K.Ramakrishnan College of Technology, Trichy, Tamil Nādu.

[3]Professor, Department of Electrical and Electronic Engineering, Malla Reddy Engineering College, Hyderabad

[1]mohandas@newhorizonindia.edu [2]aruna.devi96@gmail.com and [3]rajeshpradha@gmail.com

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 carry out applications needing the massive amount of processing cooperatively on their own by creating 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, we present a V2V-based cooperative computing approach. The suggested method takes into account the distance between vehicles while choosing which ones to collaborate with, and it waits task offloading until the last possible moment to ensure a stable and energy-efficient cooperative computing environment. Despite its competitive performance when compared to other MH algorithms, the artificial rabbits optimisation (ARO) algorithm still suffers from poor accuracy and the issue of becoming trapped in solutions. By antagonism methods, this research creates selective opposition version of the artificial rabbit procedure (LARO), which 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. The tracking efficiency is improved, and ARO is kept from getting stuck in its existing local solutions by adopting the selective opposition technique. In comparison to traditional static scheduling techniques, the suggested strategy improves upon both energy efficiency and network reliability.

Keywords: Vehicular Cloud; Artificial rabbits' optimization; Vehicular Ad-Hoc Network; Cooperative computing method; Lévy flight.

Introduction

The Internet of Vehicle Things (IoV) is a new method of increasing practical uses of vehicles. 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 limitations (such as battery life, processing power, and network throughput), smart devices are unable to 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 services for apps are also taken care of 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 better traffic management, which also decreases carbon emissions and the number of accidents that occur there [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 are responsible for sending and receiving data of any kind. In order 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 always have a connection to at least one RSU. For the purposes of traffic data collection and analysis, as well as offering entertainment services to automobiles, [10] these RSUs are retroactively connected with the TCC through the Internet cloud. Intelligent transportation systems (ITS) are made possible by wireless technologies like 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 need advanced computing capacity without any 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 sources to deal with 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. These fog nodes contain a large amount of storage space for cached data and processing power for computationally intensive activities [13]. However, because to the high speed and restricted coverage area, cars equipped with RSUs have a shorter connectivity time, which is a significant limitation in job computation. Furthermore, a fog node cannot perform all jobs concurrently due to the high number of various sorts of tasks. This slows down the execution of some jobs that aren't necessary for diverse applications, which might have a significant impact on how well the vehicular networks function. Task scheduling, in which the ideal sequence in which to conduct tasks is determined, is another essential component in enhancing 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 primarily concerned itself with reducing this executing time. However, we construct our job scheduling approach with both execution time and energy costs in mind. For electric cars in particular, the energy cost is crucial because it affects their range [15].

The primary goal is to plan the offloading of tasks using optimisation models. However, the findings of the experiments show that the optimisation based on artificial rabbits has a low convergence accuracy and becomes stuck in 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 used extensively to improve ARO's global discovery capabilities. LARO is able to better plan for avoiding local solutions and exploring foreign options thanks to the Lévy flight method. Second, an enhanced convergence-accuracy selective opposition technique allows for local LARO exploitation. Below, we list the paper's most novel and important contributions:

(i) The ARO is further enhanced by the use of the Lévy flying strategy during the random concealment phase to increase population variety and dynamics.

(ii) The algorithm's capability to escape the local optimum is enhanced by the introduction of a selection opposition strategy that builds on the original opposition strategy and re-adaptively updates it.

The remaining sections of the paper are prearranged as shadows: The literature is presented in Section 2, while Section 3 particulars the suggested perfect. In Section 4, the consequences of the experiments and their verification are provided, and in Section 5, the research is summed up and suggestions for further investigation are offered.

.

2. Related works

A cooperative computing approach using V2V communication is proposed by Gong et al. [16]. The suggested technique takes into account the distance between vehicles while choosing which ones to collaborate with, and it waits task offloading until the last feasible moment to provide a stable environment. In comparison to traditional static scheduling techniques, the results in terms of both energy savings and network reliability. Although SCOCC's speedup increase is 9% lower than the others. SCOCC does not outperform static algorithms like HEFT and GBTSA in terms of execution time since it does not attempt to optimise the execution speed of whole jobs at once. However, SCOCC accomplishes crucial stability in the context of a rapidly moving vehicle and minimises the amount of energy needed for wireless communication.

Despite these difficulties, Taha et al. [17] have developed a strategy for achieving data secrecy on (CP-ABE). To carry out CP-ABE tasks VANET nodes, we construct a cluster of vehicles. To manage these dispersed microtasks, we utilise container orchestration framework, to construct vehicle cluster(s). In this approach, we make use of a group of variables that have an effect on 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 up the work amongst vehicles according to their final combined weight. We gauge the efficacy of our method by contrasting it with Kubernetes's mechanism for workload allocation, which takes into account solely the available resources in each car. In addition to utilising simulations to assess our method's effectiveness in terms of tran, we also take into account a number 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 VANETs architecture based on Intelligent Volunteer Computing. We provide criteria for choosing volunteers whose cars can complete the computationally demanding assignment. In this research, we apply a procedure to accurately select the volunteer vehicle for job execution by making predictions about the competence of various cars. The computing power of the best volunteer cars is predicted after extensive experimentation. We compared the effectiveness of nine distinct regression methods using free, downloadable datasets. The outcomes demonstrate the effectiveness of using these methods to foretell volunteers' performance. Results from a comparison of regression methods show that ridge regression and support vector regression are superior at cutting down on 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 classification with GBT, LR, MLPC, and SVM replicas were used to test the performance of our projected classifier in OMNET++ and SUMO. The suggested model is applied to a dataset that consists of both typical automobiles and those used in attacks. The simulation findings improve the assault categorisation with 99% accuracy, which is a significant improvement. The system obtained 94% and 97% accuracy while using LR and SVM, respectively. With 98% and 97% accuracy, respectively, the RF and GBT performed admirably. Because training and testing period does not rise proportionally with the sum of nodes in the network, the performance of the network has augmented since we 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 factoring in the change of VANETs. Overall performance in CO is significantly affected by how tasks are scheduled. To reduce the likelihood of task reallocations due to connection interruption between cars in VANETs, the uses a directed acyclic graph (DAG) to express the priority relationship between jobs before assigning them to neighbours. The simulation results shown 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 inside 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 project 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. 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 across blockchain nodes without negatively impacting the efficiency of the MEC system or the blockchain itself. 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 assure stability and dependability in (VANET) installations, Taha et al. [22] offer a strategy on Differential Evolution (MOTD-DE). To pick cars that meet the cluster algorithm's requirements, we employ Kubernetes container-based as the cluster algorithm. As a result, we can handle intricate operations for the benefit of data owner automobiles. When a vehicle in our cluster joins the group, its data will be made vehicle), and the fit complexity of individual jobs will be determined by a deep learning model. 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. We also make the assumption that the jobs are decoupled. We present scenarios in which the number of jobs, cars, CPU and memory values, and vehicle are all changed to assess the efficacy of our work. MOTD-DE is superior than four other popular methods, including 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 VCN provides cloud services based on resource-intensive machines; they are prohibitively expensive and do not support mobile devices. In the section that follows, we detail our proposal for a new virtual private network (VCN) that makes use of containerized microservices.

3. Proposed System

All of the cars discussed in this article are electric and have GPS and transmission devices with the capacity for direct V2V communication as well as cellular connectivity, allowing them to carry out the functions of other vehicles as asked by VM technology. A CV initiates computation offloading by first making a direct V2V communication request to other cars within the transmission radius, and then making a WV selection based on the information provided from those vehicles. We'll 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 issue of choosing which vehicle should carry out each job is a significant contributor to the total execution time. As a default, direct V2V connection is used to transmit data and 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 it takes and the amount of energy it takes to finish an application depend heavily on the task scheduling that decides which WV will be allocated t.

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 sends out a broadcast message asking for details about them. The vehicle that is willing to share its computer capitals with v_c responds to this message with its present position, its movement status, and the computing resources that it can share with v_c. All vehicles give priority to their own tasks, thus these resources are what's left over 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 quantity of energy used and the time it takes to finish the application both depend on how jobs are scheduled, it's important to discover an allocation strategy that minimises energy use while still meeting the deadline T for the application. Due to the wireless nature of the channel, data broadcast overhead is crucial in a vehicle network. It takes energy to relay data across wireless channels to other cars. There is no use in offloading jobs if the cost of doing so, including the cost of transmitting data, is more than 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 may be readily transformed to 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.

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

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 V2V message and execution the task. v_{j} can perform ti 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 vehicles 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)×ipb_{i}, divided by the sum of orders that can be treated by v_{j} each second, ips_{j}.

$$w_{ij} = \frac{D_0(t_i) \times ipb_i}{ipb_j} \quad (3)$$

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 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 ti's i-predecessors have finished their operation and transferred the result to v_{j}.

$$LFT_{\mathrm{i}} = \max_{t_l \in pred(t_i)} \left\{ FT_l + \frac{D_0(t_l)}{b} \right\}, \quad \forall i \in \{1,2,\ldots.,n_w\} \quad (5)$$

where $pred(t_i)$ is a set of t_i's, and b is the data broadcast speed among vehicles.

A DAG already represents of tasks that are scheduled to be completed by a WV. A forerunner task must be finished before its successor may begin. When all of v_j's tasks with greater priority than t_i have been finished, v_j is free to work on t_i. by AT_{ij}.

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

The initial start period when t_i can run on v_{j}, EST_{ij}, is the latter of LFT_{i} and AT_{ij}.

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

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 transmission the consequence of t_{i} from v_{j} to WVs having t_{i}'s i-successors, which can be got as shadows.

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

For t_i to be executed in v_{j}, To ensure that all higher priority jobs are finished, the starting time, ST_i, for t_i must meet 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 processors in all cars require the same amount of power to execute an instruction, mapping produce greater change in data transmission energy than 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 much like with desktop and laptop computers, only a select few companies develop these chips. Therefore, it makes little difference where in the world tasks are executed, and this research centres on the energy expended during data transmission.

We use the energy perfect from Liu's research [33] 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 empty space, where d_jk is the separation between two points in the universe denoted by v_j and v_k. Therefore, the total amount of energy, denoted by the symbol E_tr, 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_i's i- end 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 $v_{\text{v j}}$ with ti is out of direct communiqué variety at the time t_j is finished, and ns(t_i) is the sum of vehicles of t_j.

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

where c_{ce} is the cost of cellular networks and cdi is the cost of communication. Because using a cellular network is more expensive each transmission, the total cost rises as its use increases. We factor in the potential expense of using the cellular network by thinking about the price that would need to be 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, correspondingly. As a result, we contend that direct DSRC connection is preferable to cellular communication for compute offloading whenever available.

Therefore, we essential 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:

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

In order to minimize the above equation values, the research work proposes an improved ARO model.

3.3. Basic Steps of Artificial Rabbits Optimization (ARO)

Two natural rules of rabbit survival—detour hiding—form the basis of the ARO algorithm's proposal [24]. One such tactic is called "detour foraging," and it involves rabbits eating 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. Size of the artificial rabbit colony is a design variable with dimension d. is N, 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.

Both exploration and exploitation are taken into account by the metaheuristic algorithm, although detour foraging focuses on the former. In order 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.

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

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

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

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

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

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

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 phoney 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 a random permutation of integers from 1 to d is denoted by the randp symbol.. r_1, r_2, and r_3 are stochastic numbers from 0 to 1. L signifies the running length, This is the rate of travel when on a foraging detour. There is a normal distribution for n1. The random number n_1 from a normal distribution best represents the disturbance. The final term in Equation (18) can be perturbed to aid ARO in avoiding local extremum and conducting 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, we provide a working definition of how rabbits come up with their burrows on their own. When the ith rabbit digs the jth hole, it is called a

:

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

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

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

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

where $i = 1, \dots, N$ and $j = 1, \dots, d$, and n_2 shadows the distribution. H signifies the hidden limit that reductions linearly from 1 to $1/T_{max}$ perturbations. Fig. 1 demonstrations the alteration in the worth of 1000 iterations. The lessening trend in H values shown in the image indicates a steady progression as the repetitions go.

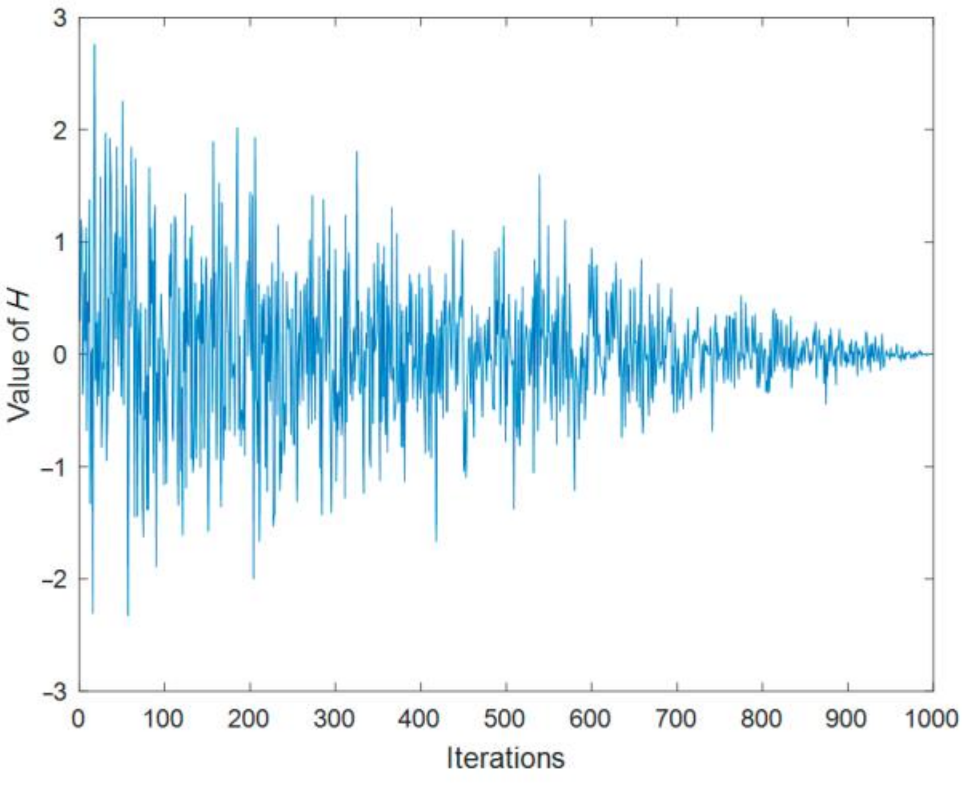

Figure 1. The alteration of H over the course of 1000 repetitions.

Below is the formula for updating the random concealment technique.:

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

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

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

Equation (31) is used to update the site of the ith artificial rabbit after the two update procedures have been applied.

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

An adaptive update is shown here in the form of an equation. Based on the adaption value, the rabbit will decide on its own whether to remain in its present location or search for a new one. In an optimisation algorithm, populations often choose carrying out the exploration phase early on and phase later on. The rabbits' dwindling vitality is used by ARO to devise a searching strategy that mimics the discovery-to-exploitation cycle. In the algorithm of the synthetic rabbit, we define the energy factor as:

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

where (r, r) are two random numbers among zero and one. By analysing the data shown in the image, we see that the value of A is dropping as a whole, which ensures a smooth change from as the iterations go.

3.4. Hybrid Artificial Rabbits Optimization

Since hybrid optimisation procedures are the result of targeted changes to the original method, which boost the various performances of the algorithm, they find widespread usage in practical engineering. When applied to the problem of wave front shaping, for instance, Liu's suggested novel hybrid method combines particle swarm optimisation with a single layer neural network to take use of the strengths of both approaches [25]. For the clustered vehicle routeing problem, Islam uses particle swarm optimisation (PSO) and variable neighbourhood search (VNS), with PSO fusing the different solutions and VNS bringing them to local optimums [26]. Devarapalli [27] suggested a cosine approach for addressing the challenge of power system stabiliser parameter adjustment in a system. As a means of compensating for the ARO algorithm's subpar accuracy and propensity to quickly descend into locally optimal solutions, we propose a hybrid, improved LARO algorithm that incorporates and then apply it to engineering optimisation problems. 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 frequently algorithms, primarily to add energy to the updates to the algorithms.. This law of generating random numbers can aid in introducing dynamism into update techniques and allowing them to escape from stagnant local minima. The following equation describes the Lévy distribution.

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

where t is the step size and is determined by (34). Equations (34)-(37) provide the formulae for determining the Lévy flight's step size.

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

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

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

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

where σ_u and σ_v are distinct as assumed in Equations (36) and (378). Both u and v obey mean 0 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 phase of random concealment, we use the random statistics generated by the Lévy flight method in place of the r4 random numbers. To prevent ARO from being stuck on local candidate solutions during the exploitation phase, which includes the random concealment step, we include Lévy flight 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 an is a constant value of 0.1.

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

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 the rabbits that are far from the ideal solution in order to get them closer to the size of the rabbit that is in the optimal location. 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.

We start by settling on a cutoff point. Until the limit case is achieved, the threshold value will be lowered. The following equation demonstrates how SO determines which rabbit location is optimum given the current rabbit dimension and the set of rabbit locations under consideration.

.

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

where dd_j is how far apart each rabbit is in every direction. The distant and near rabbit locations are computed when dd_j is larger than the value we set. Next, we 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 among 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. The ARO algorithm is improved by these tweaks, which lead to greater convergence and population diversity as well as higher-quality solutions. Listed below are LARO's methods in detail.

Step1: 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, iterations T_{Max};

Step2: Pick many locations at random for the rabbits and determine their fitness levels. Identify the bunny who is strategically placed.;

Step3: The value of the energy component A may be determined using Equation (32). If A > 1, pick any rabbit at random from each set;

Step4: Using Equations (19)–(22), determine R's value. Apply the approach of detour foraging by solving Equation (18). Then, using Equation (31), determine the new rabbit position's adaption value and update it.

Step5: 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;

Step6: The distance of each site from the rabbit dimension to the best rabbit position is intended by Equation (39);

Step7: If dd_{j} > Threshold, regulate the near size d_{f} and count the sum of df. If dd_{j} ≤ Threshold, regulate the far dc and sum the sum of d_{c}. Then compute src from the calculated dd_{j} by Equation (40);

Step8: If $src <= 0$ and $d_f > d_c$, execute Equation (41) and rabbit's location;

Step9: 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 does not add any complexity while simplifying how ARO is updated. Assessing the difficulty of addressing real-world situations is difficult, but calculations of complexity can help. The difficulty is related with the size of false rabbits N, d, and T_{Max}. The total difficulty of the rabbit's procedure is as shadows.

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

$$= O(T_{Max}\mathrm{Nd} + T_{Max}\mathrm{N} + \mathrm{N}) \quad (42)$$

With selective opposition, every facet of every possible rabbit habitat is taken into account. 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. We used Veins [30] to think about how the road actually works and how cars travel over it. 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 the processing power of all cars falls somewhere in the middle. The maximum rate of data transfer between cars is 25 Mbps. We 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 speed of data transfer between cars has been set at 25 Mbps.

Due of the simulation's narrow emphasis, we don't account for congestion by having trucks dump data in any particular order; instead, we monitor how well different algorithms divide up the work. The assignment of work to a worker car is not directly responsible for the congestion. Meanwhile, some works, such as [31], examine data offloading load balancing in a scenario where many vehicles deliver packets at once. In addition, we do not analyse the scenario in which is meaningless for the following presentation measures, as is the case with most prior research on compute offloading.

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 how many simulation runs were performed, how many packets were generated by the source node in the uth simulation run, and how many packets were received by the surface sink in the uth simulation run.

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 represent that the PDR Analysis. In this analysis, we have used different models as GWO, Whale, Butterfly with Proposed model. 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 and reached the PDR value of 200 runs at 0.29 and reached the PDR value of 300 runs at 0.39 and reached the PDR value of 400 runs at 0.42 and also reached the PDR value of 500 runs at 0.52 and another reached the PDR value of 600 runs at 0.52 and reached the PDR value of 700 runs at 0.52 respectively. 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.

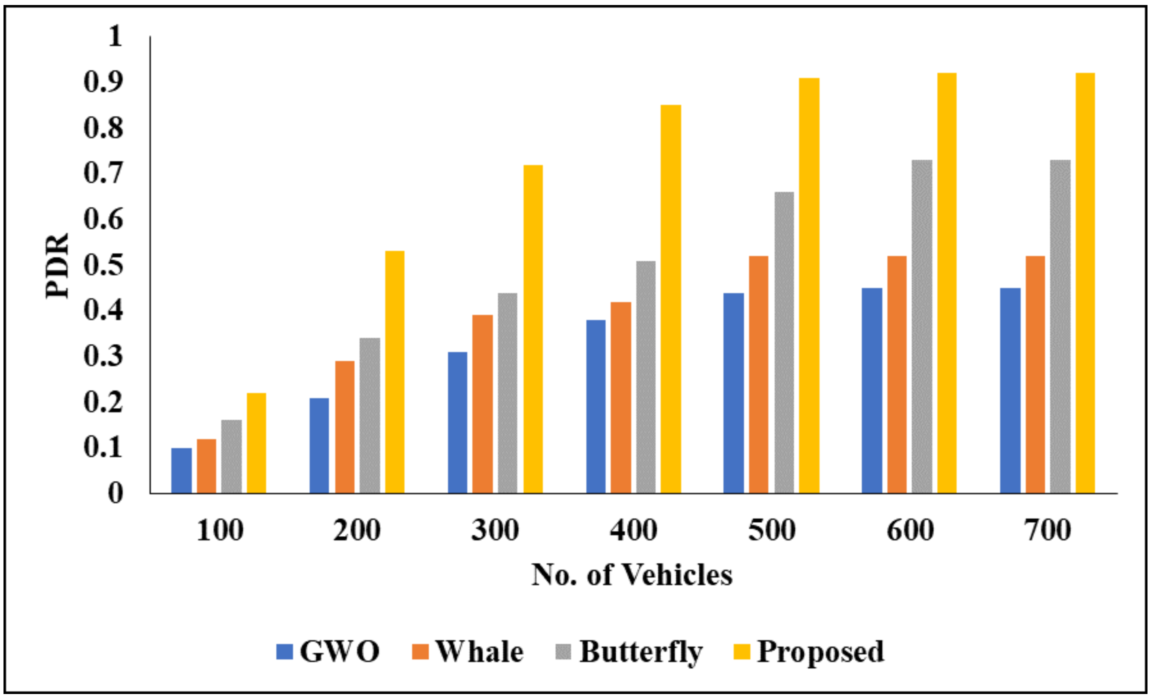

Figure 2: Graphical Analysis for PDR

.

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 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:

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

where P_r, RP_{um}, and TP_{um} are 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

In table 2 and figure 3 represent that the delay analysis validation of various models. In this analysis, we have used different models as GWO, Whale, Butterfly with 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. 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.

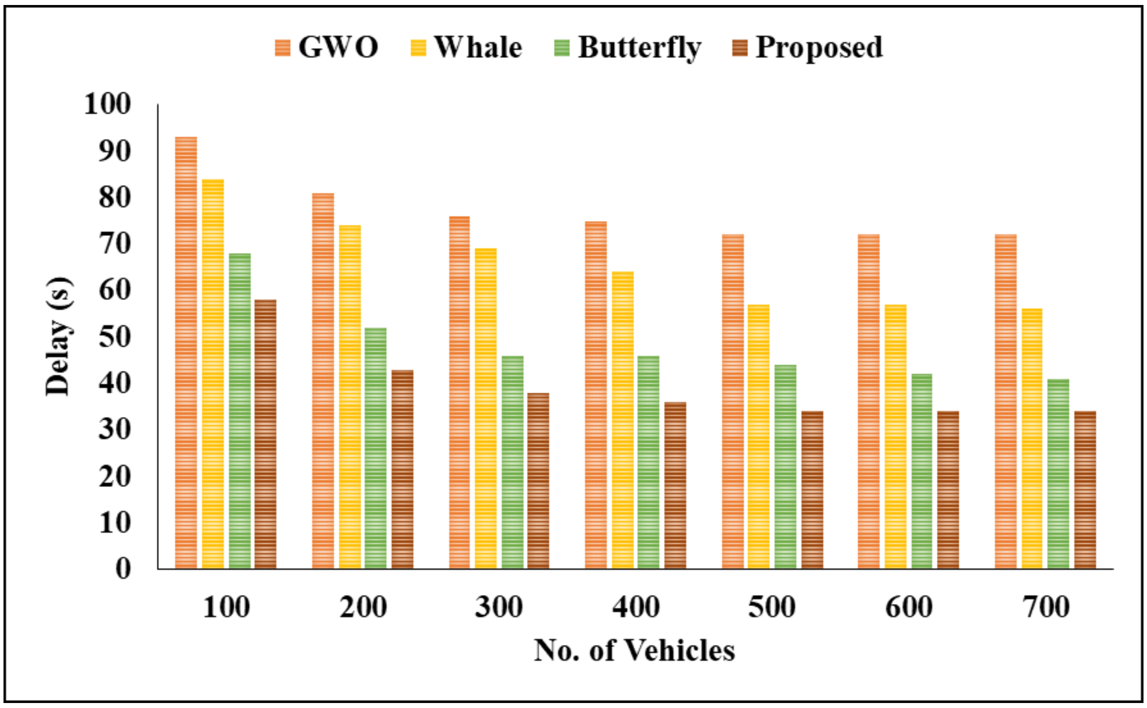

Figure 3: Validation of Various models

.

4.3. Average energy consumption

The average volume of energy used for a successful transmission comes from the packets' transmission and reception, as well as the packets' overhearing by the 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 Ptoposed 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 above table 3 and figure 4 represent that the Validation of Proposed model 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. 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.

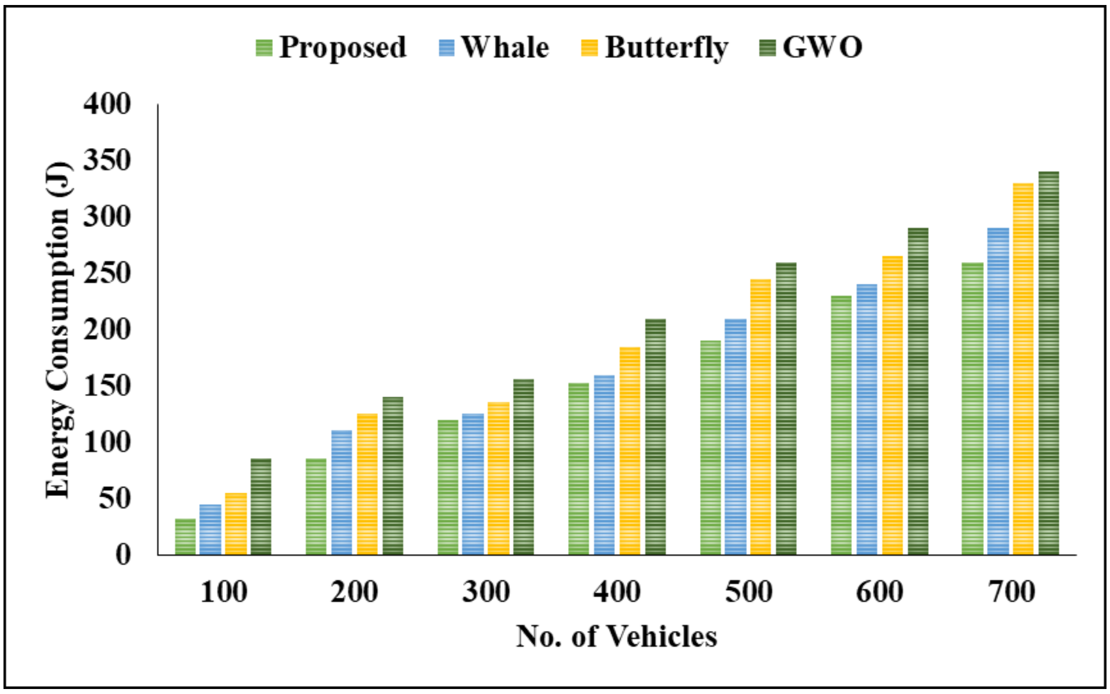

Figure 4: Graphical Representation of various models

4.4. Average network lifetime

For at least this long, the network would function normally. In this study, we define network lifespan as the period until the first participating sensor node in the network loses power. Therefore, we 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. For a statistical analysis of 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 time 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 above table 4 and figure 5 represent that the analysis based on network lifetime in 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. 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.

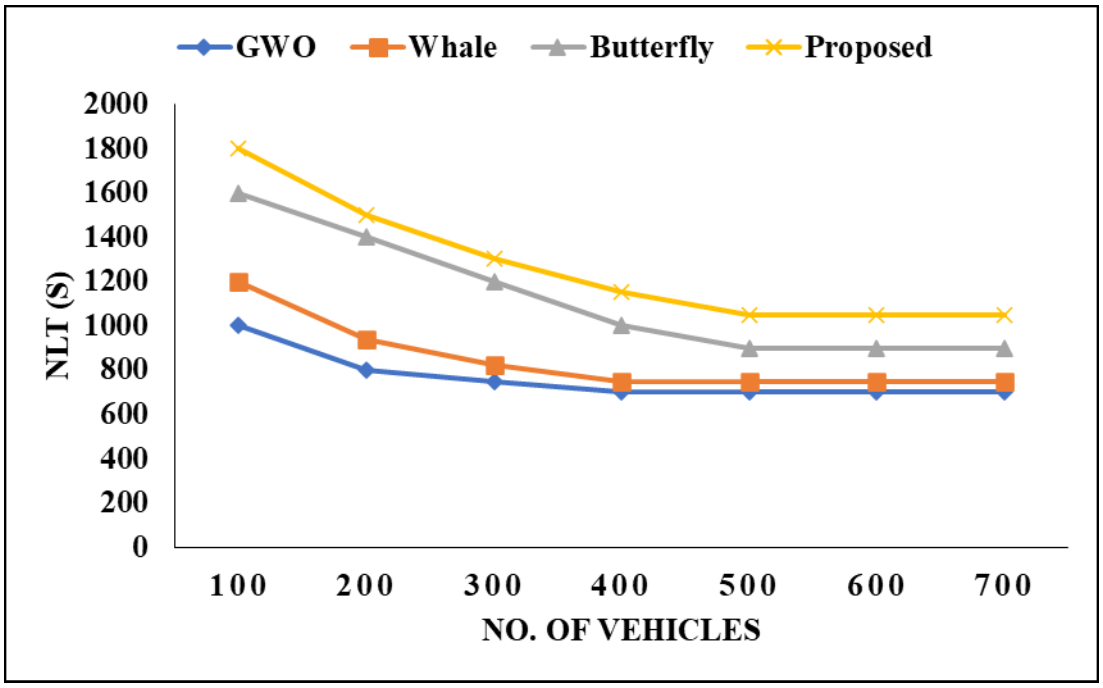

Figure 5: NLT Analysis

5. Conclusion

In order to create a reliable VANET cloud and carry out an energy- compute offloading among vehicles, the LARO model was presented in this work. The suggested approach achieves this by i) delaying job task as much as feasible and ii) taking into account the aloofness between a client vehicle and a worker vehicle when picking worker cars. Decreases in the average inter-vehicle distance and the time among job assignment and execution aided VANET stability and energy usage, respectively. Since the suggested LARO model does not attempt to optimise the execution speed of whole jobs at once, it does not outperform the static algorithms like GWO, Whale, and butterfly. LARO, on the other hand, maintains crucial steadiness in the context of a rapidly moving vehicle while also minimising the amount of power needed for wireless transmission. To lower application expenses, both in terms of storage and processing time, we will propose a revolutionary serverless approach in further work. All dynamic and mobility-aware failures will be handled by the reinforcement aware offloading and scheduling algorithm, according to the study's recommendations.

Declarations

Ethical Approval

No human trails or animal trails are conducted for this research work

Competing interests

None declared.

Authors' contributions

The papers' concept is designed by Dr. R. Mohan Das. Under his guidance, the paper is written by Arunadevi Thirumalraj. The flow of the work, correction and English grammatical errors are corrected by T. Rajesh.

Funding

No funding are received for this project

Availability of data and materials

No data are available.

References

[1] Khayyat, M., Elgendy, I.A., Muthanna, A., Alshahrani, A.S., Alharbi, S. and Koucheryavy, A., 2020. Advanced deep learning-based computational offloading for multilevel vehicular edge-cloud computing networks. IEEE Access, 8, pp.137052-137062.

[2] Xu, S. and Guo, C., 2020. Computation offloading in a cognitive vehicular networks with vehicular cloud computing and remote cloud computing. Sensors, 20(23), p.6820.

[3] Xu, S., Guo, C., Hu, R.Q. and Qian, Y., 2021. Blockchain-Inspired Secure Computation Offloading in a Vehicular Cloud Network. IEEE Internet of Things Journal, 9(16), pp.14723-14740.

[4] Li, X., Dang, Y., Aazam, M., Peng, X., Chen, T. and Chen, C., 2020. Energy-efficient computation offloading in vehicular edge cloud computing. IEEE Access, 8, pp.37632-37644.

[5] LiWang, M., Gao, Z., Hosseinalipour, S. and Dai, H., 2020, June. Multi-task offloading over vehicular clouds under graph-based representation. In ICC 2020-2020 IEEE International Conference on Communications (ICC) (pp. 1-7). IEEE.

[6] de Souza, A.B., Rego, P.A.L., Rocha, P.H.G., Carneiro, T. and de Souza, J.N., 2020, December. A task offloading scheme for wave vehicular clouds and 5G mobile edge computing. In GLOBECOM 2020-2020 IEEE Global Communications Conference (pp. 1-6). IEEE.

[7] Liu, Z., Dai, P., Xing, H., Yu, Z. and Zhang, W., 2021. A distributed algorithm for task offloading in vehicular networks with hybrid fog/cloud computing. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 52(7), pp.4388-4401.

[8] Lakhan, A., Ahmad, M., Bilal, M., Jolfaei, A. and Mehmood, R.M., 2021. Mobility aware blockchain enabled offloading and scheduling in vehicular fog cloud computing. IEEE Transactions on Intelligent Transportation Systems, 22(7), pp.4212-4223.

[9] Raza, S., Liu, W., Ahmed, M., Anwar, M.R., Mirza, M.A., Sun, Q. and Wang, S., 2020. An efficient task offloading scheme in vehicular edge computing. Journal of Cloud Computing, 9, pp.1-14.

[10] Shakarami, A., Ghobaei-Arani, M., Masdari, M. and Hosseinzadeh, M., 2020. A survey on the computation offloading approaches in mobile edge/cloud computing environment: a stochastic-based perspective. Journal of Grid Computing, 18, pp.639-671.

[11] Kang, Y., Liu, Z., Chen, Q. and Dai, Y., 2020. Joint task offloading and resource allocation strategy for DiffServ in vehicular cloud system. Wireless Communications and Mobile Computing, 2020, pp.1-15.

[12] Sun, J., Gu, Q., Zheng, T., Dong, P., Valera, A. and Qin, Y., 2020. Joint optimization of computation offloading and task scheduling in vehicular edge computing networks. Ieee Access, 8, pp.10466-10477.

[13] Tang, L., Tang, B., Zhang, L., Guo, F. and He, H., 2021. Joint optimization of network selection and task offloading for vehicular edge computing. Journal of Cloud Computing, 10(1), pp.1-13.

[14] Guo, H., Liu, J., Ren, J. and Zhang, Y., 2020. Intelligent task offloading in vehicular edge computing networks. IEEE Wireless Communications, 27(4), pp.126-132.

[15] Li, H., Huang, H. and Qian, Z., 2021, September. Latency-aware batch task offloading for vehicular cloud: Maximizing submodular bandit. In 2021 IEEE 14th International Conference on Cloud Computing (CLOUD) (pp. 584-593). IEEE.

[16] Gong, M., Yoo, Y. and Ahn, S., 2023. Vehicular Cloud Forming and Task Scheduling for Energy-efficient Cooperative Computing. IEEE Access.

[17] Taha, M.B., Alrabaee, S. and Choo, K.K.R., 2023. Efficient resource management of micro-services in vanets. IEEE Transactions on Intelligent Transportation Systems.

[18] Haris, M., Shah, M.A. and Maple, C., 2023. Internet of Intelligent Vehicles (IoIV): An Intelligent VANET Based Computing via Predictive Modeling. IEEE Access.

[19] Rashid, K., Saeed, Y., Ali, A., Jamil, F., Alkanhel, R. and Muthanna, A., 2023. An Adaptive Real-Time Malicious Node Detection Framework Using Machine Learning in Vehicular Ad-Hoc Networks (VANETs). Sensors, 23(5), p.2594.

[20] Gong, M., Yoo, Y. and Ahn, S., 2022. Adaptive Computation Offloading with Task Scheduling Minimizing Reallocation in VANETs. Electronics, 11(7), p.1106.

[21] Poongodi, M., Bourouis, S., Ahmed, A.N., Vijayaragavan, M., Venkatesan, K.G.S., Alhakami, W. and Hamdi, M., 2022. A novel secured multi-access edge computing based vanet with neuro fuzzy systems based blockchain framework. Computer Communications, 192, pp.48-56.

[22] Taha, M.B., Talhi, C., Ould-Slimane, H., Alrabaee, S. and Choo, K.K.R., 2022. A multi-objective approach based on differential evolution and deep learning algorithms for VANETs. IEEE Transactions on Vehicular Technology.

[23] L. Zou, A. Javed, and G.-M. Muntean, ‘‘Smart mobile device power consumption measurement for video streaming in wireless environments: WiFi vs. LTE,’’ in Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcast. (BMSB), Jun. 2017, pp. 1–6.

[24] Wang, L.; Cao, Q.; Zhang, Z.; Mirjalili, S.; Zhao, W. Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems. Eng. Appl. Artif. Intell. 2022, 114, 105082.

[25]. Liu, K.; Zhang, H.; Zhang, B.; Liu, Q. Hybrid optimization algorithm based on neural networks and its application in wavefront shaping. Opt. Express 2021, 29, 15517–15527.

[26]. Islam, M.A.; Gajpal, Y.; ElMekkawy, T.Y. Hybrid particle swarm optimization algorithm for solving the clustered vehicle routing problem. Appl. Soft Comput. 2021, 110, 107655.

[27]. Devarapalli, R.; Bhattacharyya, B. A hybrid modified grey wolf optimization-sine cosine algorithm-based power system stabilizer parameter tuning in a multimachine power system. Optim. Control. Appl. Methods 2020, 41, 1143–1159.

[28] Service, T.C. A No Free Lunch theorem for multi-objective optimization. Inf. Process. Lett. 2010, 110, 917–923.

[29] Arini, F.Y.; Chiewchanwattana, S.; Soomlek, C.; Sunat, K. Joint Opposite Selection (JOS): A premiere joint of selective leading opposition and dynamic opposite enhanced Harris' hawks optimization for solving single-objective problems. Expert Syst. Appl. 2022, 188, 116001.

[30] C. Sommer, D. Eckhoff, A. Brummer, D. S. Buse, F. Hagenauer, S. Joerer, and M. Segata, ‘‘Veins: The open source vehicular network simulation framework,’’ in Recent Advances in Network Simulation. Cham, Switzerland: Springer, 2019, pp. 215–252.

[31] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, ‘‘Joint load balancing and offloading in vehicular edge computing and networks,’’ IEEE Internet Things J., vol. 6, no. 3, pp. 4377–4387, Jun. 2018.