applied sciences

MDPI

Article

Cooperative Content Caching Framework Using Cuckoo Search Optimization in Vehicular Edge Networks

Sardar Khaliq uz Zaman [1,*], Saad Mustafa [1], Hajira Abbasi [1], Tahir Maqsood [2], Faisal Rehman [1], Muhammad Amir Khan [1,*], Mushtaq Ahmed [3], Abeer D. Algarni [4] and Hela Elmannai [4]

[1] Department of Computer Science, COMSATS University Islamabad, Abbottabad Campus, Abbottabad 22060, Pakistan
[2] Department of Computer Science, COMSATS University Islamabad, Lahore Campus, Lahore 54000, Pakistan
[3] Software Engineering Department, Faculty of Science and Technology, ILMA University, Karachi 75190, Pakistan
[4] Department of Information Technology, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, P.O. Box 84428, Riyadh 11671, Saudi Arabia
* Correspondence: skhaleeq@cuiatd.edu.pk (S.K.u.Z.); amirkhan@cuiatd.edu.pk (M.A.K.)

Abstract: Vehicular edge networks (VENs) connect vehicles to share data and infotainment content collaboratively to improve network performance. Due to technological advancements, data growth is accelerating, making it difficult to always connect mobile devices and locations. For vehicle-to-vehicle (V2V) communication, vehicles are equipped with onboard units (OBU) and roadside units (RSU). Through back-haul, all user-uploaded data is cached in the cloud server's main database. Caching stores and delivers database data on demand. Pre-caching the data on the upcoming predicted server, closest to the user, before receiving the request will improve the system's performance. OBUs, RSUs, and base stations (BS) cache data in VENs to fulfill user requests rapidly. Pre-caching reduces data retrieval costs and times. Due to storage and computing expenses, complete data cannot be stored on a single device for vehicle caching. We reduce content delivery delays by using the cuckoo search optimization algorithm with cooperative content caching. Cooperation among end users in terms of data sharing with neighbors will positively affect delivery delays. The proposed model considers cooperative content caching based on popularity and accurate vehicle position prediction using K-means clustering. Performance is measured by caching cost, delivery cost, response time, and cache hit ratio. Regarding parameters, the new algorithm outperforms the alternative.

Keywords: content caching; content popularity; vehicular edge network; cuckoo search optimization

Citation: Zaman, S.K.u.; Mustafa, S.; Abbasi, H.; Maqsood, T.; Rehman, F.; Khan, M.A.; Ahmed, M.; Algarni, A.D.; Elmannai, H. Cooperative Content Caching Framework Using Cuckoo Search Optimization in Vehicular Edge Networks. *Appl. Sci.* **2023**, *13*, 780. https://doi.org/10.3390/app13020780

Academic Editor: Juan-Carlos Cano

Received: 11 November 2022
Revised: 6 December 2022
Accepted: 8 December 2022
Published: 5 January 2023

1. Introduction

Due to the rapid development of network technologies, wireless devices have been widely adopted over the past few decades, and people mostly rely on them. The devices are very large in numbers and need to be connected for reliable communication, especially in peer-to-peer (P2P) applications, where nodes are sharing the data instead of centralized servers [1]. The vehicular network provides vehicle-to-vehicle communications, but it isn't easy to fully satisfy the users in distributed and highly dynamic networks without accurately predicting requested data and nodes [1,2]. VEN uses dedicated short-range communications to ensure reliable data delivery in minimum time spam, and data are shared using Wi-Fi or cellular networks. Vehicular communication is performed by installing on-board units (OBUs) on vehicles [3]. The whole network architecture is based on the number of layers. The top layer is the cloud network (CN), connected to MECs through back-haul links that are further connected to RSUs. Roadside units are deployed near roads based on particular vehicular regions and managed by base stations [3], as shown in Figure 1.

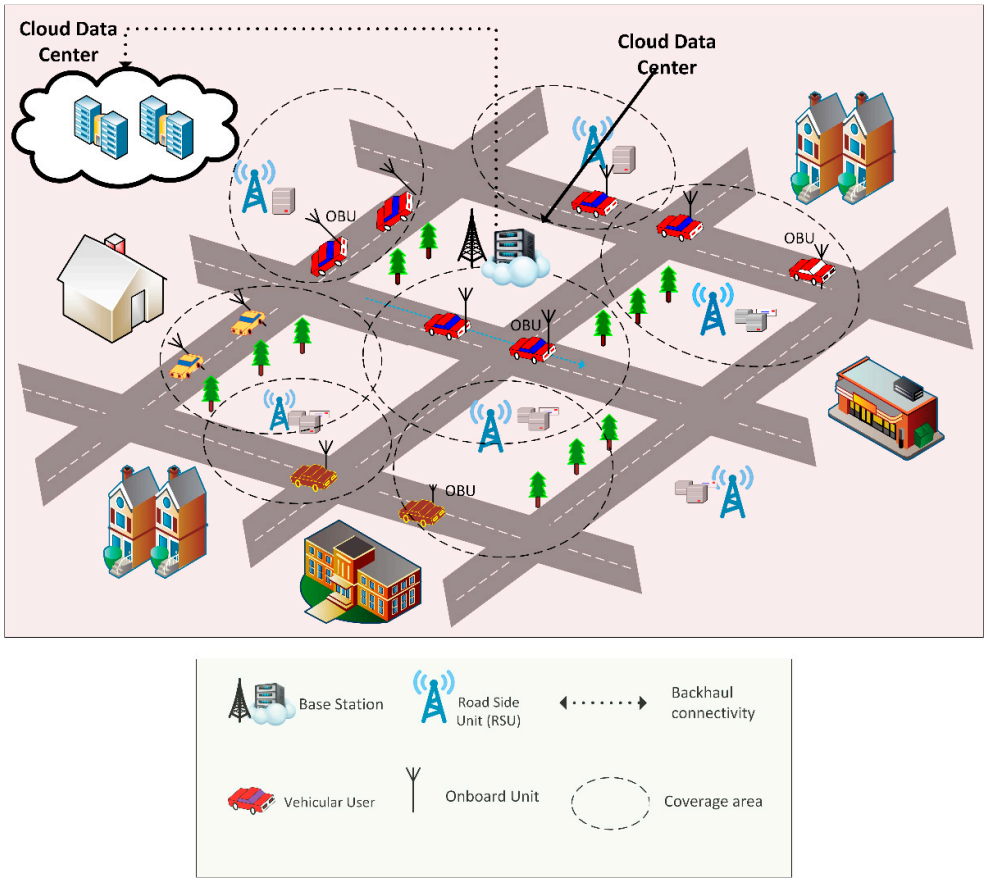

Figure 1. Vehicular edge network architecture.

As the vehicle passes through many RSUs in high mobility, complete content cannot be accessed from a single caching node [1]. Due to the highly dynamic vehicular network, caching the complete contents at a single RSU is problematic, as the time between RSU and the end node is tens of seconds [4]. Hence content placement and delivery are two separate issues in a short time. Users only demand more secure and efficient data access without being interested in content delivery location [5]. The vehicle fetches data from RSUs and shares the traffic information (traffic safety and management) with the other vehicles. If the content is not cached at RSU, it is fetched from the base station (BS), thus increasing the fetching cost and delay. Due to limited storage capacity, computational resources, and high mobility, pre-caching is required. Caching increases content availability and reduces the request response time; however, caching has numerous challenges that affect the network performance such as network topology, caching placement, delivery on time, and optimal route [6].

In [7], cost-optimal cooperative caching is proposed, considering the finite archive capacity and bandwidth, to achieve content popularity with freshness. The proposed scheme in [7] optimizes content placement and delivery with the cooperation of BS, RSU, and smart vehicles. Double time scale Markov optimization is used, as content updating time is much faster than the vehicle's movement. Data are updated before placement in the content delivery process by vehicle scheduling (routing table management) and allocating bandwidth on different time scales. Vehicles that recently entered in coverage area will remain for a long time in the zone; therefore, it is more reliable to store and share more data in Adhoc mode.

A tradeoff occurs in response time and content freshness in high-mobility networks. To address this problem, the content can be cached at RSU and then updated periodically, considering the optimal parameters [8]. Although to ensure the delivery of fresh content, resources are consumed. In [9], the authors optimized the throughput and energy

consumption by caching the content in vehicular environments using cellular networks. The data is cached on distributed vehicles that serve as task schedulers for user requests to minimize the response time. However, the given scheme lacks cooperation regarding data sharing among vehicles and servers, which affects the content freshness. Study [10] considers the cache deployment problem in long-term evolution vehicle-to-infrastructure (LTE-V2I) networks with highly dynamic VENs that jointly optimize the cache size with maximum average download percentage, considering the following parameters: vehicle speed, arrival rate, and content popularity. The proposed scheme uses a joint cost optimal algorithm to optimize the cost in terms of time. The Poisson process is used for vehicular mobility prediction and Zipf popularity distribution for content caching. It reduces the response time with data availability.

VENs enable vehicle-to-vehicle (V2V) and vehicles-to-infrastructure (V2I) communication. Data are cooperatively cached at RSUs and base stations and fetched from the cloud so vehicles can access the data from the nearest station [11]. The vehicular node comprises an on-board unit (OBU), and RSUs provide access to a wide range of coverage areas [12]. Both have limited storage capacity, but RSU's storage capacity is comparatively greater than OBU's. The RSU fetches the predicted content from the upper layers and then delivers it to the connected vehicles on demand, as shown in Figure 2.

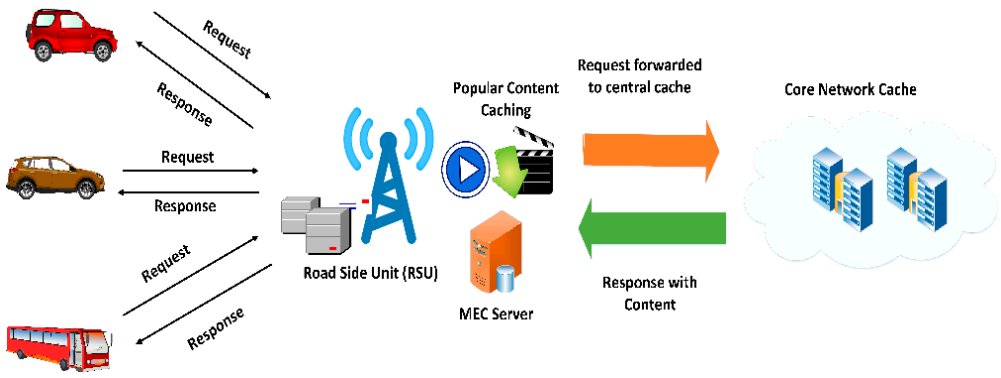

Figure 2. Content caching in VENs.

Problems in VENs are limited storage resources and rapid network topology change [13]. Pre-caching reduces the latency and increases the cache hit ratio; therefore, less bandwidth is required to access the data. Mostly content is cached on the bases of zones, freshness, and popularity. Response time increases to ensure content freshness, although some applications are not delay tolerant. Meanwhile, content popularity compromises freshness. Given work shows, that tradeoff occurs in freshness and popularity. An appropriate caching and optimization strategy can escalate the content availability to ensure the cache hit ratio and response time while considering the tradeoff factors. Moreover, to find an optimal solution to pre-cache the requested data on limited storage devices, so that data could be delivered on time to maximize the cache hit ratio. The major contributions of the paper are given as,

- We propose a cooperative content caching method for the VEN environment. The proposed system increases the cache hit ratio and resource utilization while reducing the network delay.
- We use the K-means clustering for multi-tier caching servers (base stations and road-side units) to cache the single content at multiple servers cooperatively.
- We develop communication models for vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I), content caching and delivery models that are further used by cuckoo search algorithm (CSA) to determine vehicle location and optimize content delivery.
- We conduct extensive experiments exploiting the geo-life trajectory data to predict the requesting node's position and MovieLens dataset for user contents.

- The experimental results reveal that the proposed framework outperforms the traditional ant colony optimization (ACO) in cooperative and non-cooperative caching scenarios.

The rest of the paper is organized as follows. Section 2 discusses the motivation behind the presented research with a gap analysis. Section 3 details the literature review. The problem statement is formulated in Section 4. In Section 5, different system models are discussed. Section 6 describes the proposed framework and techniques involved in it. The performance evaluation is described in Section 7. Section 8 concludes the paper, and finally, some limitations of the existing works compared with the proposed one, and some future research directions are discussed.

2. Motivation and Gap Analysis

This research aims to propose an efficient and cooperative optimization technique for V2V and V2I communication in VENs. In existing work, several meta-heuristics-based (nature-inspired) optimization techniques have been used to find the optimal path to store the data in the nearest station in the vehicular network, such as genetic algorithm (GA), simulated annealing (SA), particle swarm optimization (PSO), and ant colony optimization (ACO). GA is an evolutionary algorithm based on selecting a parent's chromosomes for reproduction. Crossover and mutation are carried out to obtain offspring or new solutions. It cannot process in the case of a single chromosome; it starts from many points, having a higher fitness value. GA cannot search from a particular area, so it takes much time to explore the whole search space [14]. The time complexity of GA is $O\ (n^{3/2}\ log\ n)$ when a gene has the same length. In PSO, the swarm of n particles flies over a search space, and during iteration, each particle updates its current position according to its past and neighbor's experience. If one particle flies in the wrong direction, it can mislead the whole swarm; moreover, it cannot work in the problem of scattering. PSO has random walks and is not suitable for continuous problems.

The expected runtime of PSO is $\Omega\ (n/log\ n)$ and $O\ (n\ log\ n)$, which is between the lower and upper bound. Its average case of asymptotic notation is $\Theta\ (n\ log\ n)$. In ACO, the initial parameter for ants is randomly assigned, and then the path length is calculated to find the optimal solution by updating the pheromone. Improper selection of initial value may lead to local optima [15]. The time complexity of ACO is $O\ (n^*(n-1)^*m^*\ T/2)$. As Big-O represents the worst-case scenario; so, it takes maximum time to process the large-scale problems and easily falls into the second-best solution, and many ants cause congestion.

In contrast, cuckoo search (CS) solves continuous problems. Cuckoo species inspire it, laying their eggs in the nest of other species, which increases their survival and productivity [16]. There is a 10% probability that the host bird discovers an egg and reproduces stably. Moreover, CS has great robustness in terms of convergence, assigning initial parameters and dependencies on other swarms [17]. Therefore, we use CS in our research to develop an optimized algorithm that finds the best path to pre-cache the requested content, minimize the response time and cache cost, and enhance the cache hit ratio.

3. Related Work

In this section, we provide a comprehensive literature review on caching concepts that highlight the part of content popularity, challenges, and cache requirements for VEN, a brief assessment of the optimal pathfinding, using meta-heuristic optimization techniques and pre-caching using cooperative content caching technique. Proposed solutions of different researchers have also been explained and briefly analyzed. The schemes discussed in the literature review are pre-caching content based on popularity and route optimization techniques.

Pre-caching improves performance and reduces response time for delay-sensitive applications in vehicular networks. The problem is how to make pre-caching more effective. In [17], the authors consider a large-scale infrastructure to improve the edge system capacity. Vehicular caching solves the given problem efficiently by caching mobile data on distributed vehicles to improve the system throughput and energy consumption. In [18], the proposed

scheme preaches the contents based on road topology and zone information. Pre caching zone selection (PCZS) algorithm is proposed using pre-caching node selection (PCNS) under the NS-3 platform to achieve service reliability with improved hit ratio, average delay, and mobility prediction. The algorithm finds the vehicle's motion and sojourn time (time to live in a zone) based on past data so that more data can be cached on a vehicle having a long sojourn time. This way, caching vehicles that stay in a zone for a long time can serve more neighboring vehicles within a minimum period.

RSU-based edge caching scheme is proposed in [19] using cross entropy content placement algorithm to minimize the load of the back-haul network. Only popular data is cached in RSU using Zipf distribution. If the requested content is unavailable, then the current RSU fetches the content from neighboring RSUs. Hence data are cached at RSUs cooperatively in case of non-availability of requested data; it is retrieved from the linking base station. Poisson distribution is used to predict the arrival of vehicles in the coverage area of RSUs. The coverage area of RSU is further divided into several zones, considering the vehicle's velocity and traffic density. Different transmission rates are allocated to these zones. A zone at the RSU's coverage area's entrance caches more data. The proposed scheme reduces the Latency while improving the cache utilization; as a result, it gives higher throughput.

In [20], a cooperative peer-to-peer (P2P) caching scheme is proposed to share the information among vehicles rather than fetching it again and again from RSU to reduce the response time. Markov chain is used to model the randomly changing behavior of vehicular networks with three states [21]. Probability is calculated to replace the existing data with newly arrived data based on the time consumed in the waiting state and the frequency of accessing data in a given time. The result outperforms in terms of congestion, delay, and hit ratio; however, it has a lack of server (V2I) cooperation that affects the content freshness and scalability.

In [22], cloud-based VANET architecture data are cached at numerous layers, named vehicular cloud (VC), local internet cloud (LIC), and then at RSUs cooperatively, to improve the service response time [23]. Caching placement is done by convex optimization problem and simulated annealing (SA), to achieve the caching gain with average low latency. In highway scenarios, vehicles move in identical directions; therefore, a platoon-based mobility model is used. If the requested content is cached in advance by any vehicular cloud member, then it will be satisfied by V2V communication, otherwise fetching will be conducted in V2I communication. Finally, the request is satisfied via a remote server. To maximize the caching gain, all VC members collaboratively decide about caching. SA is a trajectory-based search algorithm starting with an initial guess and final solution at a high temperature then gradually cools down the system. It accepts the solution if it is better, otherwise, it relies on the probability that causes weak exploitation. SA often converges very slowly, and often finds global optimal solutions at the expense of a large number of function evaluations.

The study [24] focuses on the distribution of multimedia and large-size files in high-mobility VANETs, by caching the updated contents at RSUs. Three algorithms are used to retrieve the files at RSUs: (1) optimal one: reduces downloading time by exhaustive searching but computational complexity will increase; (2) sub-optimal: allocates the most requesting files (using Zipf's distribution) by doing substitution frequently; (3) greedy algorithm: fully occupies the RSUs using current best caching after clearing it. The complexity of these algorithms is calculated with the number of RSUs, storage capacity of an RSU, and vehicle speed. Requesting vehicle's entrance time is calculated first, and then the large file is divided into small parts. Then, distributed in RSUs cooperatively; that are deployed in the same sequence. In this way, the average download time of the vehicle for large files will reduce until and unless the vehicle remains in the same direction. The result shows that the average delay reduces by 70% as compared to no caching in RSU.

As compared to other networks, available resources in VENs are limited. Therefore, proper utilization of these resources is compulsory; that can be done by clustering of nodes.

Study [25] uses the nature of whales for cluster optimization. The given study outperforms grey wolf optimization (GWO) and ant lion optimization (ALO) in the number of cluster heads, transmission range and size of search space, and number of nodes [26].

Cooperative multi-tier edge caching is proposed, in which location-based and popular contents are fetched cooperatively from servers in [12]. Large files are scrambled first using fountain code and then cached at RSU/BS, based on optimal location and data segments. Cached contents are analyzed by the downloading rate and the time a vehicle takes to complete the coverage area of a server. The knapsack problem is formulated for content placement and the ant colony optimization algorithm (ACO) is used for optimality. These components lead to the shortest path with a rapid solution and distributed computation to avoid premature convergence. More ants give the best optimal solution in case of local search. It finds the optimal route, but the shortest path causes increased congestion. Moreover, it is not suitable for large search spaces.

According to the study [15], a vehicle fetches the contents from RSU or BS at a time, similarly, the information-centric network (ICN) relies on the host-centric model, which reduces congestion and improves delivery regarding data storage. Moreover, ICN has a simple configuration with data-level security. Proactive caching with mobility prediction (PCMP) is proposed using ICN, based on named data network (NDN) architecture to download the content. At first, it calculates the number of chunks to download and cache the data at the next predicted RSU. Long short-time memory (LSTM) is used to predict the next RSU in any nearest direction. In the LSTM model, Euclidean distance is calculated for more accurate prediction in the current and next RSU. RSU selects the next hop and sends the pre-requested chunks to it. The study outer performs in delay, hop selection, and cache utilization, but focuses only on RSU's prediction and caching, while there is a lack of V2V caching.

In [27], the optimized link state routing (OLSR) protocol is used to moderate the performance of VANET. A routing table is maintained for all routes. The harmony search (HS) algorithm is used to configure the parameters of OLSR by coupling two stages: roulette wheel selection and tournament selection. It sends a hello message among all nodes; after receiving a hello message, each node creates its network topology independently and runs the Dijkstra algorithm to select the shortest route towards the destination. OLSR tags every message with a sequence number to differentiate the fresh and previous information. It is an efficient method to deliver the data through an optimal path, but blind flooding can cause operation costs.

In [28], safety-based applications broadcast a basic safety message (BSM), which contains information about vehicles such as location and speed, to detect and avoid collisions. To improve privacy at the application layer, genetic algorithm (GA) and public key infrastructure (PKI) are used. GA helps to find the best neighbor which actually has cached the requested data. For the privacy of requesting nodes, PKI is used, to hide the hardware and software information of the device from third parties. Ns-3 simulator is used for server and vehicular applications. GA needs limited parameter setting and initializes itself from possible solutions instead of a single solution. The main drawbacks of GA are the lack of fast convergence towards the optimal solution and the long processing time required for optimality.

Study [14] focuses on enhancing the bandwidth efficiency of I2V communication considering software-define vehicular networks (SDVN). Where, RSUs are connected with the controller, to take the scheduling decisions based on received requests from vehicles. RSUs and vehicles function as a data node, connected via a backbone network with SDN controllers. When a vehicle enters a particular coverage area, it sends location, velocity, and direction information with requested and cached data information to the corresponding RSU through beacon messages. The information is collected by the controller of the corresponding RSU through a wired connection. Then, the decision is made and notified to RSU by the controller. If two vehicles have the same cached data packets, i.e., P1 and P2, then XOR operation (P = P1 XOR P2) will be performed to decode the packet.

In conventional approaches, two time slots are required to broadcast the two packets separately, but with the help of coding, it is completed in a single time slot to enhance the efficiency of broadcast bandwidth. Binary particle swarm optimization-based coding scheduling (BPSO-CS) algorithm is used, in which the particle determines a set of packets for RSU to serve the vehicles to minimize the broadcast time. The fitness function is used to evaluate the particles and then updating is done to speed up the searching convergence. SUMO is used for simulation to trace the vehicles according to the Manhattan mobility model. The drawback of PSO is it does not create new birds from parents such as GA, as the particle moves towards the destination using the current best position/velocity, and the slow convergence time is another drawback of PSO.

Edge caching reduces the latency and balances the network load [29]. The problem in caching is which file, at what time, and at which node should be cached. Caching based on popularity is not enough to satisfy users in high mobility, although an accurate path/route prediction also helps to solve this problem. In [30], a new caching strategy is proposed based on the GA, to optimize the best and shortest path and the Markov model, to accurately predict the next route on the basis of past/ historical data. User clustering is done by the same behavior that helps to adjust the upcoming cell. Pre-caching the popular content may not be effective in case of dramatically different content requests; moreover, search is limited to the global best. In the case of a main road or single path, route accuracy will be high.

In study [31], a comparison of simulated annealing (SA), ACO, and GA is performed based on the traveling salesman problem. Performance is evaluated in terms of the shortest distance and execution time. The result shows that SA has the shortest execution time (<1 s) and performs average in the shortest distance, while ACO performs better for the shortest distance, but it takes a long execution time. Furthermore, GA is fast and easy in terms of computational resources but there is no optimal solution in both the shortest path and time. However, ACO is greedier and gives better results with large problem sizes with more ants. Algorithms such as GA and PSO are population based and have more challenging objective problems that may have noisy function evaluations and many global optima. The number of candidate solutions causes robustness to the find the local optima. While cross-entropy and simulated annealing use random numbers for objective functions, and a lot of sampling is required, they are able to handle the problems with local optima [32]. As mentioned in the related work, several optimization techniques have been used in VEN such as GA, PSO, ACO, SA, and cross entropy-based caching. Study [16] shows the comparison of optimization algorithms, in which cuckoo search performs better than GA, PSO, ACO, and SA in terms of global and local search. A better algorithm uses less computation and less iteration. Cuckoo search (CS) is more efficient than GA and SA in random walk and similarity between eggs can produce a better new solution in terms of good mixing ability. The way that an algorithm converges is critically important impacting the speed, hence the minimum computational cost (in terms of time) is needed to find an optimal solution.

For intelligent transportation systems, the internet of things (IoTs) is widely used in VANETs; the number of sensors is deployed in vehicles [33]. For effective information routing, finding the shortest path is very challenging. This paper uses a discrete cuckoo search based on Levy flight and random walk against GA to find the optimal path in minimum time spam [34]. The inverse mutation operator is used for exploitation in CS to find the optimal path. If a vehicle wants to pass a message but there is already traffic congestion in the case of a road accident and traffic jam, then a message cannot be broadcast. Thus, a loop-free (single route) is required to deliver a message in high mobility; CS based on levy flight with a random walk can find the path, by flooding the welcome message. But a random walk is not suitable for the continuous optimization problem.

Study [35] focused on the discrete cuckoo search algorithm (DCSO) using MPI and Beowulf cluster to improve the quality of the traveling salesman problem for small datasets. Code runs sequentially ‘n’ times, while parallel in ‘n’ nodes. MPI and Beowulf cluster

are created to achieve parallelism. The results show that sequential programming may be faster for small datasets, but for large datasets, it takes more time. As time complexity of CS is much better than other nature-inspired meta-heuristic algorithms in terms of assigning initial parameters, system dependencies, and storage. We will use a novel cuckoo search algorithm based on Gaussian distribution instead of levy flight; because the random steps that are taken from levy distribution are large. A large step has infinite variance and mean. Gaussian distribution takes smaller steps without missing the optimal path; moreover, it has a high convergence rate [36].

In [37], the authors proposed that CS WKCA (weighted K-medoids clustering algorithm) with clustering provides routing information in vehicular networks to reduce road accidents and develop an intelligent transportation system. The cluster head will change periodically due to mobility; moreover, the nodes that are not in the neighbor are called worst nodes. Therefore, the border node is replaced with a robust one for safety measurement. While in our proposed scheme, clustering information is handled by RSUs in the form of a routing table. There is no need to change the cluster head repeatedly. The given work provides user safety in mobility but does not predict the caching node to provide the data to the expected upcoming user.

In [38], the authors determine the optimal path with the lowest routing cost using a hybrid whale dragonfly optimization approach with rider-integrated cuckoo search (RI-CS) to find the best route with minimal routing cost. The study shows the denial-of-service attack and allocates bandwidth to the devices in VANET, by evaluating the targeted energy consumption. It searches for the device with low or dropped bandwidth and then provides the required bandwidth by measuring the distance. But pre-caching and clustering are not used for end-node cooperation to minimize the delay ratio and to gain more throughputs.

The authors in [39] proposed a giraffe-kicking optimization algorithm (GKO). It uses hybrid C-means to find the redundant sensor nodes to avoid energy consumption and to improve packet delivery consumption. As C-means is an iterative clustering to avoid dissimilarities among the data points. Table 1 summarizes the significant related works.

Table 1. Comparison of the related work.

References	Year	Technique	Problem	Considered Parameters	Limitations
[18]	2020	PCZS, PCNS with N-S3 simulator	Pre-caching for none delay tolerate applications in vehicular networks	Dynamic adaptability, reduce response time and increase cache hit ratio	Accurate location prediction
[19]	2018	Cross entropy with Poisson distribution	Caching selective data at RSUs because of large content size and short storage capacity in fully distributed networks	Improved performance in delay, cache hit ratio and overhead	
[22]	2017	SA	Improved caching placement policy to maximise caching gain	Minimise latency and improve user's quality of experience	Cost, find only global best solution
[24]	2018	Optimal1, Zipf distribution	Delivering multimedia files within limited storage capacity	Pre-caching files to reduce download time, efficient delivery	Storage capacity, high mobility
[25]	2021	Whale optimisation	Based on the clustering to find the optimal path	Cluster head manage the cluster for better performance	Time, distributed nature
[12]	2020	ACO and knapsack	Fetching location based and popular contents from multiple caching servers.	Delay, service cost, finds local best solution	Coverage, reliability, robustness

Table 1. *Cont.*

References	Year	Technique	Problem	Considered Parameters	Limitations
[15]	2018	LSTM with Euclidean distance	To improve content delivery in host centric network	Accurate prediction of next caching server to utilise cache, reduce delay	High mobility, scalability, V2X
[27]	2021	Harmony search routing protocol	To find the shortest route using OLSR		Blind flooding
[28]	2020	PSO	Enhancing BW efficiency in I2V communication	Minimises broadcast time	Convergence time is slow
[14]	2019	GA with Morkov chain	Cache popular content with accurate path prediction on the basis of past data or history	Reduces latency and data traffic load	
[16], [32]	2015, 2020,	Comparison of nature-inspired algorithms and CS	Comparison between metaheuristic optimisation algorithms in terms of TSP	Initial parameter, dependency, memory and time complexity	CS performs better but recognition of egg by host bird can cause problem.
[34], [35]	2019	Comparison of meta-heuristic algorithm	Minimum path	Delay, hit ratio	CS performs better
[37]	2022	Optimized routing with CS-WKCA	K-medoids clustering to find dissimilarities	Avoid road accident	K-medoids is more complicated and unable to recover from database
[38]	2022	Whale dragonfly with RI-CS	Rider-integrated CS is used to find best rout with minimal cost	Provide bandwidth to dropped BW devices, evaluate energy consumption	No clustering therefore lack of cooperation among end users
[39]	2022	Giraffe-kicking optimization in VANETs	Hybrid C-means iterative clustering	Avoid energy consumptions	No pre-caching to reduce delivery delay

The above-mentioned related work shows that CS outer performs optimal route finding and popularity prediction on the basis of user preferences to pre-cache the data, which is a more suitable option to entertain the user conveniently. It will also improve the cache hit ratio while caching more files nearest to the user in high mobility. Complexity analysis of the state-of-the-art and baseline techniques is shown in Table 2 and the abbreviations used in the study are defined in Table 3.

Table 2. Complexities comparison of significant algorithms.

Algorithm	Parameters	Complexity	Justification
GA	Population size, mutation, crossover	$O\ (n^* m^* c)$ $O\ (n^{3/2}\ log\ n)$	All three operations will conducted multiple times, which increases its complexity
PSO	Number of particles (N), acceleration co-efficient, inertia weight, neighboring range, number of iterations, random values	$\Omega\ (n/log\ n)$ and $O\ (n\ log\ n)$ *that is* $\Theta\ (n\ log\ n)$ *Between lower and upper bound*	Stability problem restricts the success rate, secondly need memory to update velocity If N increases, complexity increases
ACO	Number of ants, evaporation rate, alpha and beta (control parameters)	Average case: $O\ (n)$ Worst case: $O\ (m^*n)$	n $\rightarrow$ number of ants If n increases complexity will also increase

Table 2. *Cont.*

Algorithm	Parameters	Complexity	Justification
CS	Population (n), egg-matching probability, number of iterations ($Rmax$), step size/Dimension size (D)	Best case: *O (1)* *O (n. D. Rmax)*	N = 1 always same due to 1 cuckoo, Dimension range (0,1), if number of iteration increases it will not significantly affect the complexity
1 < log n< n < n Log n< n^2 < n^3< 2^X< n^X → Run time from better to worse			

Table 3. Abbreviations and their definitions.

S. No	Abbreviation	Full Form
1.	RSUs	Road site units
2.	MBS	Mobile base station
3.	PCZS	Pre-caching zone selection
4.	PCNS	Pre-caching node selection
5.	P2P	Peer to peer
6.	V2I	Vehicle to infrastructure
7.	OBUs	On-board units
8.	VEN	Vehicular edge network
9.	VANET	Vehicular ad-hoc network
10.	MECs	Mobile edge computing devices
11.	SDVN	Software Define Vehicular Network
12.	PKI	Public key infrastructure
13.	OLSR	Optimized link state routing
14.	ICN	Information centric network
15.	PCMP	Proactive caching with mobility prediction
16.	V2V	Vehicle to vehicle
17.	ACO	Ant colony optimization
18.	PSO	Particle swarm optimization
19.	GA	Genetic algorithm
20.	SA	Simulated annealing
21.	CS	Cuckoo search
22.	LTE-V2I	Long-term evolution vehicle to infrastructure
23.	BS	Base station

4. Problem Statement

Existing works focus on cooperative content caching and nature-inspired optimization techniques with the dependent nature of the algorithm, which can cause failure in the case of path loss and continuous problems. Related work shows that COOP caching with clustering and CS finds both local and global optimum solutions. CS also performs better in terms of time complexity. By using proposed optimization algorithms and cooperative content caching techniques on cross-tier caching servers using popular content, the network performance is improved in terms of cache hit ratio, network delay, and resource utilization.

5. System Model

We consider a multi-tier VEN, including vehicles (end-user), RSUs, and MBS. MBS and Backhaul links are connected to the cloud server and database. In VEN, end-users/vehicles

transmit their data to the connected RSU. RSU covers the circular region of a minimum of 100 m near the roads to facilitate the users in high mobility. The area of RSU is not overlapped; the vehicle that is not in range of RSU will be directly connected to MBS.

We used cooperative edge caching with K-means for multi-tier caching servers (MBS and RSUs) to cache the single content at multiple servers cooperatively. If the requested content is available in these servers, then it can be fetched by the vehicle through the coverage area. Content delivery from a remote server will increase both delay and cost. First, we focus on the coverage area of MBS, where the number of RSUs is deployed. We consider the number of RSU as D = {D_1, D_2, D3 ... ... D_N} coverage areas of RSU and MBS as D_B and D_R, set of vehicles is V = {V_1, V_2, V_3 ... ... V_N}. RSU, MBS, and vehicles are equipped with caching capacities, such as S_{MBS}, S_{RSU}, and S_V. MBS acts as a network management controller, collecting data from edge servers and vehicles and making caching decisions. After that, MBS will share the copy of cached data to the connected RSUs to minimize the service delay and cost in terms of bandwidth (BW) utilization. Edge servers share their working state with the controller server, determining whether the server is overloaded or underloaded before transferring the workload. This external layer resource management process can be accomplished by managing the flow of data in access control sources while operation on VMs of edge servers remains undisturbed [40].

5.1. Vehicle to Base Station (V2B) Communication Model

If we assume that MBS serves the vehicle, then the transmission rate $R_{B\cdot i}(t)$ is calculated with the help of Equation (1).

$$R_{B,\,i}\,(t) = B_i\,(t) \cdot log_2\,(1 + (P_T \cdot L(D_i(x))/\sigma^2)) \tag{1}$$

where $B_i\,(t)$ denotes the bandwidth of the vehicle at time t, P_T expresses the transmission capacity of the base station (BS), L shows the average height of the antenna in meters, and $D_i(x)$ denotes the distance between the vehicle and MBS in kilometers.

5.2. Vehicle to Road Side Unit (V2R) Communication Model

Vehicle to RSU communication; coverage area is divided into zones $k = 7$, range of each zone is d_1, d_2, d_3 ... d_k, transmission rate at each zone is denoted as R_k and equally allocated to each vehicle. The bit rate of node at time t from RSU to V is expressed in Equation (2).

$$R_{R,i}(t) = R_k/N_k\,(t) \tag{2}$$

where $N_k\,(t)$ shows the number of vehicles in kth zone at time t. The amount of data provided by RSU as a download service to all corresponding nodes is expressed in Equation (3).

$$T_R = \sum_{k=1}^{7} N_k \cdot \frac{dk}{E[v]} \cdot \frac{Rk}{Nk} \tag{3}$$

There is a control buffer in RSU with the size of T_R that stores all the data requested by the node and deletes the data when the buffer overflows.

5.3. Content Caching

Data are cached on MBS, RSUs, and vehicles, as caching capacity of the vehicle is limited, and most recently fetched data are cached in it. While RSU and MBS servers have more caching and computation capability, both can serve a number of users at a time. These vehicular edge computing servers (MBS, RSU) download all the data from the cloud server and database back-haul link. Popular data are cached in BS, while location-based data are cached in RSU, and vehicle caches recently used data. After predicting, popularity data are downloaded one time on BS and all RSUs cooperatively receive copies of downloaded content from BS. There is no need to download the content again by all RSUs; end-users/vehicles will fetch that content from their controlling server. All the

vehicles within 100 m are controlled by a single RSU; however, 1 MBS can serve a number of RSUs at a time server.

5.4. Content Delivery Model

If the vehicle sends a content request, the corresponding controller will process it. RSU makes the downloading decision on the basis of a request. If the content is recently cached by a node in its coverage area, RSU will send the node's location (longitude and latitude) to the requesting node; then the requesting node will fetch the content from a specific node through V2V communication. Popular files (PF) can be fetched by vehicles at any location within the coverage area of MBS; however, location files (LF) are served by RSU. First, RSU will check in neighboring RSUs, in case of non-availability, PF is downloaded from MBS. After that, if there is again a request for the same content, it will be stored in RSU's cache and will entertain the user with minimum delay. In the case of successful V2V communication, the probability of downloading files will be minimum ($P_{V2V} < 1$ and >0), while in V2I communication the probability will be maximum ($P_{V2I} = 1$), and the probability of V2I is calculated using Equation (4).

$$P_{V2I} = 1 - P_r\ (A_{ji} > S_f) \tag{4}$$

where A_{ji} represents the amount of data transmitted during V2V communication and S_f denotes the size of the file in packets

5.5. Delay Calculation Model

In *V2I* communication, the transmission process determines the size and delay requirement of delivering file *f*. N_t^f denotes the number of file segments at time *t*. The duration of fetched file segments is defined as T_n^f, $n = 1,2,3 \ldots Ntf$ and downloaded data volume during the fetching time period is S_n^f, $n = 1,2,3 \ldots N_t^f$. If data are downloaded from RSU, then the amount of downloaded data in nth segment will be $S_n^f = S_R^f$ and the transmission delay of each packet delivered by RSU is defined using Equation (5).

$$t_R^f = T_n^f / S_R^f \tag{5}$$

Here, the number of fetched files in a limited time duration is divided by the total amount of downloaded data. The transmission delay from MBS is calculated using Equation (6).

$$t_B^f = L/R_B^f \tag{6}$$

where L is the size of delivered packets divided by the average transmission rate from the MBS. To evaluate the delay performance, the mean delay of total downloaded packets is calculated.

6. Cooperative Content Caching Framework

We proposed a novel framework that works in a cooperative behavior with CS to find the optimality, under multiple tiers. Although CS is used in various research, the uniqueness of our technique is to provide pre-caching while predicting the upcoming server and neighboring nodes. Pre-caching, mobility prediction, and cooperative behavior with CS to find the optimal path increases its overall performance in terms of delivery delay, hit ratio, and caching cost. First, end-user rating data are acquired, including their interests, such as the types of movies and content they prefer. Then, the server caches the most popular material. Content caching helps users and reduces network traffic. Second, send the requested content using the best route. We employ K-means for caching on cross-tier caching servers and cuckoo search for path optimality due to its improved speed. In this work, only popular content is cached in the MBS and surrounding RSUs based on user preferences and popularity estimates. As a user requests content, it is already cached on the server and provided quickly. But if a user requests content that has not been cached, the request will be sent to a central cache such as MBS. MBS's popularity is conveyed to

RSUs in its area. The MEC server first downloads content, then distributes cached data to associated RSUs. We employ Geo-life Trajectory to anticipate the requesting node's position. User-ID, spot-ID, location name, longitude, latitude, and time are the key attributes of the data set to determine content popularity. After calculating popularity based on user-ranked movies, data is cooperatively pre-cached on caching servers.

6.1. Caching Techniques

We propose cooperative content caching (COOP) with K-means clustering to cache data using the cuckoo search algorithm in VENs to determine the optimal path for content delivery. We devised a caching approach that takes user preferences into account. User opinion is reflected in content ratings. The top suggested movies are cached at RSUs and MBS using COOP. When a user requests a video, the server checks its cache and delivers it if it is there. This is a cache hit; the server boosts its popularity after delivering the content. If the requested content isn't in the cache, this is a cache miss. The total number of files in RSU is calculated using Equation (7).

$$F_R = M + \sum_{R=1}^{n} N_R \tag{7}$$

Caching capacities of MBS, RSU, and vehicles are S_{MBS}, S_{RSU}, and S_V in GBs. Caching constraints are explained below whose purpose is to cache the files less than the total caching capacity, i.e., $S_V < S_{RSU} < S_{MBS}$.

a. Total number of files cached by RSU is $\sum_{f=1}^{F} {s_R}^F \leq S_{RSU}$
b. Total number of files cached by MBS is $\sum_{f=1}^{F} {s_B}^F \leq S_{MBS}$
c. Total number of files cached by vehicle is $\sum_{f=1}^{F} {s_V}^F \leq S_V$

Content delivery follows caching. Find the ideal path to deliver content to a node. The existing approach employs ACO, explained later.

6.1.1. Cooperative Content Caching (COOP)

In COOP, the MBS server calculates content popularity based on user preferences, then downloads popular data through a back-haul link and delivers PF content to RSUs on demand. Neighboring RSUs share data instead of downloading it from a BS server or controller. RSUs cache and serve location-based data such as GPS and weather forecasts. Cross-tier collaboration between MBS and RSUs eliminates the need to recalculate and download content because RSU caches it after fetching it from MBS, as demonstrated in Figure 3. COOP prevents data duplication and saves time and money. For every file content placement, Equation (8) is used since RSU will not download all packets of a single file due to limited caching capacity; caching resources would be wasted because requesting nodes are mobile.

$$S^f = ({S_B}^f \cdot {S_R}^f) \tag{8}$$

To avoid redundancy and maximize cache capacity, we use the following caching limitations as expressed in Equations (9) and (10), respectively.

$${S_B}^f + \sum_{R=1}^{n} {s_R}^f \leq S^f \tag{9}$$

$${S_R}^f \leq S_{RSU} \; and \; {S_B}^f \leq S_{MBS} \tag{10}$$

Content placement using COOP is formulated as shown in Equation (11), where n denotes the number of possible placing content for file f.

$${T_R}^f(n) = {S_B}^f(n) \cdot {S_R}^f(n) \tag{11}$$

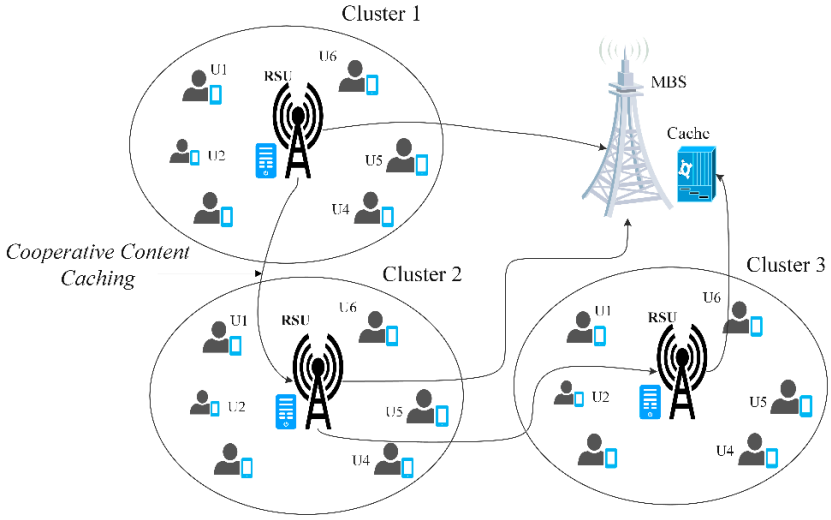

Figure 3. Cooperative content caching (COOP).

6.1.2. COOP with K-Means Clustering

We also utilize K-means to cluster the users managed by RSU. As the user sends a request to RSU, it checks data in nearby clusters. If data are located in the first or third cluster, RSU stops searching and sends the caching node's information to the requesting node, as shown in Figure 4. In this technique, the requests are fulfilled by surrounding nodes through V2V communication, and no server is required. Algorithm 1 shows the working of Cooperative Content Caching.

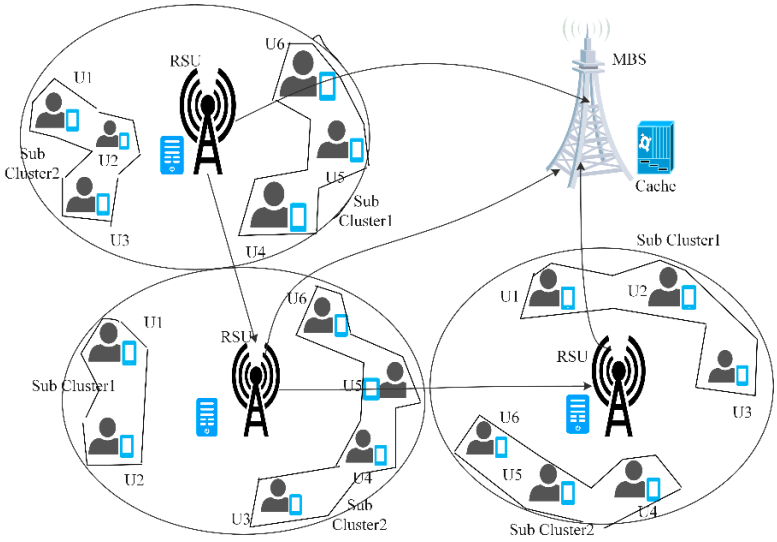

Figure 4. COOP using K-means clustering.

First, the number of vehicles and the total number of RSUs at time t within a single MBS are initialized. Popularity is calculated in MBS and, popular data are cached on RSUs. Each vehicle is handled in a parallel fashion. When the vehicle sends a request to the server, RSU searches the requested content in its clusters and the vehicle gets data directly from that node. Otherwise, RSU searches in its local cache; after finding the content in its cache, it is directly delivered to the node, elsewhere downloaded from MBS.

Algorithm 1: Cooperative Content Caching on MBS, RSU, and Vehicles

Begin
$N_V{}^t$: set of vehicles in entering into RSU at time t
$N_{RSU}{}^n$: n = {1,2,3 . . . . . 10} Number of RSUs within one MBS
For each vehicle v ϵ N_V parallel do
If (V_i sends request)
for i = 1 to k, K= $N_V{}^f$ do
Compute Popularity (PF_B, $N_V{}^f$)
$P_{j,k} = P_{r,j}.\ (1/\ K)/\sum_{n=1}^{Fj} 1/n$ calculating popularity PF_B at MBS
$S^f = (S_B{}^f.\ S_R{}^f)$: Content placement on RSU and MBS
End for
Analyze the request x
$S_R{}^f$ = total $Pf_B \rightarrow$ PF cached in RSU storage
for j = 1: Find j ϵ C_k: C_k = number of clusters
if (PF/ LF == $N_V{}^f$): File found in cluster
Data passed to vehicle
else if (x = = $S_R{}^f$)
Sense the file at RSU
Compute V2R communication
else if
Download from MBS
End if
End for

6.2. Popularity Prediction

After getting user preferences, we estimate the overall popularity of movies. Those movies which are rated by most of the users and whose mean rating is higher are considered the top-most popular movies. MBS predicts the popularity, downloads that content, and delivers popular content to the corresponding RSUs. When a user's request arrives at the RSU, then content will be provided directly to the user without using the back-haul bandwidth. Otherwise, the content will be fetched from the core network through the back-haul link with extra overhead and delay.

Location-based files served by RSU are represented as, $N_R = LF^R$, and popular files served by MBS are expressed as, $M = PF^B$, hence the total number of files stored in RSU can be determined by Equation (12).

$$F_R = M + \sum_{R=1}^{n} N_R{}^f \tag{12}$$

The popularity is calculated by two variables, F_j showing the number of files at MBS, and F_i representing the number of files at RSU keeping the K as a constant of popularity. Equation (13) finally expresses the content popularity.

$$P_{j,k} = P_{r,j} \cdot (1/\ K)/\sum_{n=1}^{Fj} 1/n \tag{13}$$

6.3. Optimisation Using Cuckoo Search

In our model, we use nature inspired optimization technique named cuckoo search (CS), to find the shortest path with minimum time spam and cost, as shown in Figure 5. Each cuckoo lays eggs in a number of host nests but the cuckoo will find a host bird with high quality based on the probability. Cuckoo search with Levy flight will take a lot of smaller steps after taking random longer steps. CS parameters are Inertia location or the cuckoo itself, number of host nests, number of step sizes, number of eggs, and brood parasitism. Algorithm 2 shows the working of Cuckoo search.

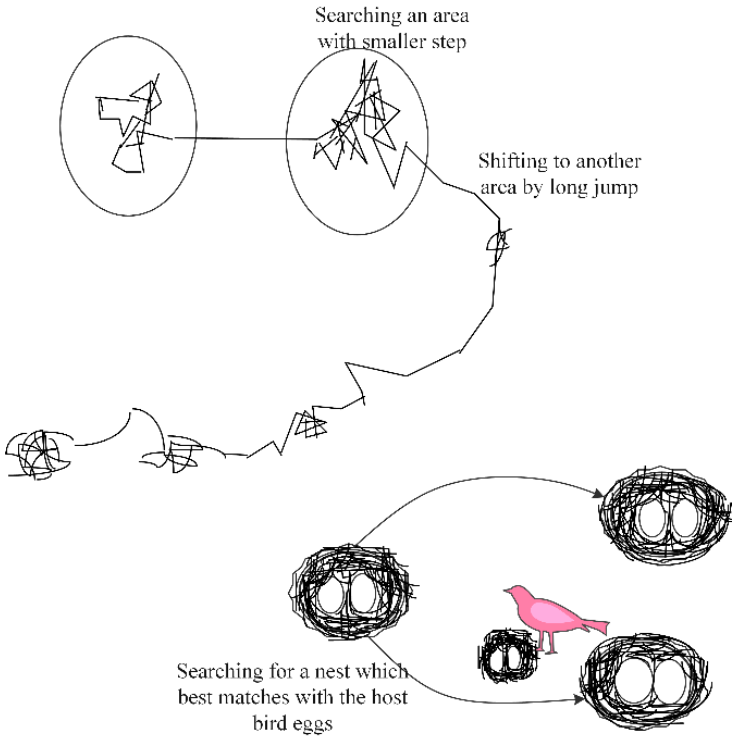

Figure 5. Working of cuckoo search.

As compared to ACO, the CS has less parameters; there is no dependency to take the next step and brood parasitism is the best parameter to find better output; moreover, it will not stick into local optima. The objective function is shown in Equation (14), where a new iteration is taken by cuckoo p, using current iteration $x_p{}^t$, Levy flight distribution function is used to update the solution *Levy* (λ), while α is the step size.

$$f(x) \rightarrow x_p{}^{t+1} = x_p{}^t + \alpha \oplus Levy\ (\lambda) \tag{14}$$

Algorithm 2: Content Caching Using Cuckoo Search

Require: Initial route population for vehicle *V:* V_P, Best route V_R*, Reliability of best route *V**

1. **Begin**
2. **Compute**. λ, α,
3. **For** k = 1 to N_P do
4. **Compute:** u, v and step of levy flight as Levy(λ) = (u/v − λ)
5. **Compute D:** distance between current route V^k and best route V*
6. **Compute:** step size h = D. α. Levy (λ)
7. Apply inverse mutation operator f_{inv} to find new route
8. $V^k = f_{inv}$ (V^k, h, Q1, Q2)
9. Compute reliability of new route
10. $R_P \leftarrow \{R_P, R^k\}$: reliability score of new route
11. $V_P \leftarrow \{V_P, V^k\}$: store new reliable route
12. **If** $R^k > R^*$ then
13. Update R* with new one R^k
14. Update V^k and V^*
15. **End if**
16. **End for**
17. **Return** best route and reliability score

6.4. Content Caching with Clustering

First, MBS and RSUs will cache popular information together. When a user requests content, it is already cached on the server and provided quickly. If a user requests content that has not been cached on the server, the request will be sent to the central cache. Another component of our research is to create clusters based on user requests; when a car sends a request, RSU searches the cluster for the data and provides it instantly. Clustering saves time because the entire coverage area is not explored.

7. Performance Evaluation and Results

We simulated the vehicular edge environment consisting of the edge servers (RSU and MBS) and several users located within the coverage area of the RSUs. We use 3 MBS and 10 RSUs in each MBS and exploit the K-Means clustering technique for user grouping. When a vehicle sends requests to the RSU, it searches that content in those clusters near users. The coverage area of RSU is 100 m, and MBS is 1000 m. If RSU finds the content, it sends that node's location to the requesting node. By clustering, there is no need to search out all users in the coverage area. The dataset used in our experiment is MovieLens, collected from the MovieLens website [41]. We have only considered those movies that have more than 10 user rates. To decide which are most popular, we set two parameters: first, the high rating given by users to different movies and second, how many users rate a movie. At the end, those movies that most users rate get high preference as compared to those which are rated by only a few users. In order to simulate the users' content request process, we consider the movie ratings given by a user as the request for that movie just as assumed in [42]. We compared the proposed technique (CS) with the state-of-the-art scheme (ACO) to assess the performance based on cache hit ratio, delay, delivery cost, and caching cost. Table 4 details the simulation settings and implementation parameters.

Table 4. Simulation settings and modeling parameters.

S. No	Parameters	Value
1.	Caching buffer	100–700 MB
2.	Network size	40, 60, 80, 100, 120
3.	No. of contents	1000
4.	Content size	10 MB
5.	Vehicle speed	25–75 km/h
6.	Length of cluster (L)	50 m, 35 m, 20 m
7.	Number of iterations	50
8.	Rate of model learning	0.002
9.	Validations failures	8
10.	Hidden layers	5
11.	Number of polling layers	3
12.	Training data size	897,560
13.	Testing data size	102,440 (11%)
14.	Time taken to train the model	617 min
15.	Testing time	0.043 ms
16	Exponential decay (λ)	0.05
17.	No of users	50–500
18.	RSU transmission range	100 m

7.1. Non-Cooperative Content Caching (Non-COOP)

In non-cooperative caching, MBS and RSUs make their caching decisions independently. If they want to cache the popular content, then popularity will be calculated on each caching server. There is no cooperation to calculate and download the popular content cooperatively, which leads to computational cost and time consumption; moreover, data duplication will also occur on the same tier.

7.1.1. Reactive Caching Model

In the reactive caching technique, when a vehicle sends a request to RSU, it fetches the content from BS or the back-haul server. There is no predicted pre-cached data; RSU will download the data as requests are received [29].

7.1.2. Pro-Active Content Caching Model

Although in proactive caching there is no cooperation between neighboring RSUs and BSs, data are pre-cached on RSU. So, pre-caching is performed here. It takes the same time to download the content from the back-haul link as reactive caching takes. But the delay in delivering the content to the vehicle will be short as compared to reactive caching; due to its pre-caching capability [29].

7.1.3. Cooperative Content Caching (COOP) + K-Means

According to this strategy, we have chosen the list of popular predicted contents and cached them at MBS, then PF content is delivered to the RSUs on demand. Neighboring RSUs share the data cooperatively before requesting to the MBS server. After fetching the data from RSU the vehicle stores the data in its cache for a limited time. At first, when the user sends a request to the RSU, RSU searches the requested data in its clusters and sends the information of caching device to the requesting node; in this way, the devices can get that data from their neighbors with minimum delay.

7.2. Cache Hit Ratio, Caching Cost, Delivery Cost and Delay with Different Cache Sizes

Figure 6 shows that by increasing cache size, the availability of different types of content will increase. In case of content availability, more users can get access to the local cache so the hit ratio will also improve. In the case of maximum cache size COOP (CS), the hit ratio is almost 27% better than COOP (ACO) and around about 43% better than proactive caching. As there is no cooperation in RSUs and in reactive caching, it goes in the same fashion and does not increase, because it will download the content from the upper layer server each time. In our proposed model, the server searches the content in its clusters first and there is no need to search the whole data set, as data are cooperatively cached in the server so the hit ratio will be better.

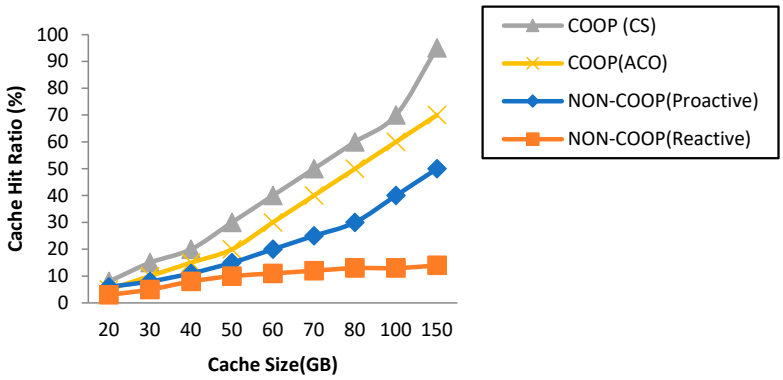

Figure 6. Cache hit ratio with different cache sizes.

Figure 7 shows that data are required according to the caching capacity, if there is more capacity it will store more data. Caching capacity is bought in BW. There is almost a 10% difference in the result when the caching capacity is higher and in the case of minimum cache size, it is 7% in the COOP (ACO) comparison graph. In COOP (ACO) and NON-COOP (proactive), there is a little bit of difference because it also pre-caches the content but there is a lack of clustering. In reactive caching, there is a very small amount of cache to store and process the data; in case of increasing cache size, we have to buy the cache.

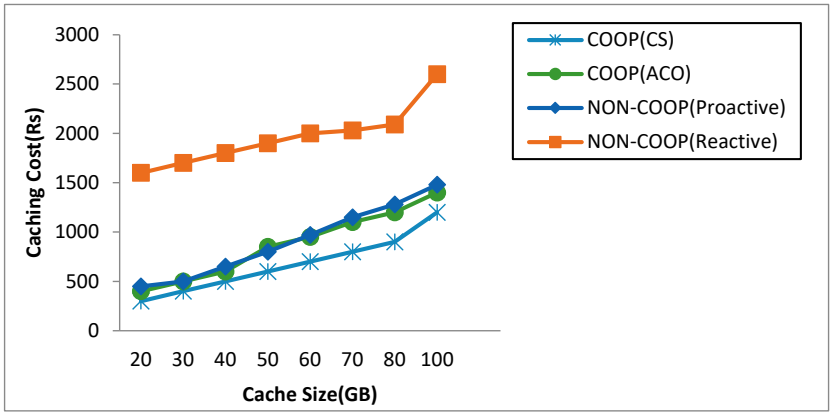

Figure 7. Caching cost with different cache sizes.

Figure 8 shows that increasing cache sizes data delivery costs will be minimized because requested data will be pre-cached in the server and not need to fetch content from the upper layer. When the cache size is 150 (GB), the caching cost in COOP (ACO) is 28%, while in COOP (CS) it is 25%. In proactive, when the cache size is less than the delivery, the cost is almost 25% less, but it fluctuates after some time, in case of not finding the content in its local cache. While in the reactive cache, it always downloads content from the back-haul server. Hence, the delivery cost will increase more.

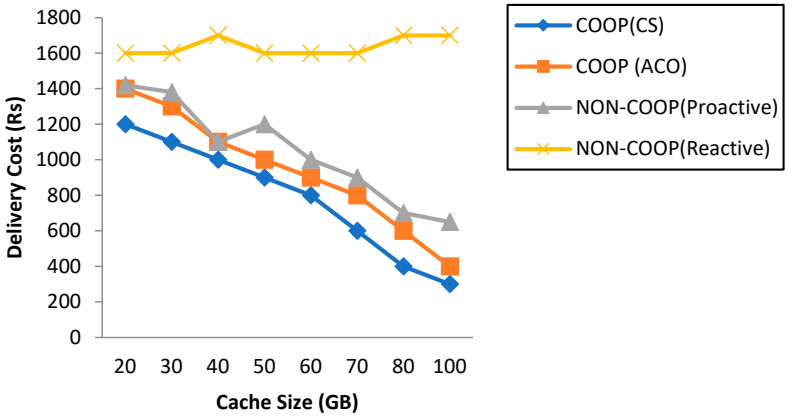

Figure 8. Delivery cost with different cache sizes.

Figure 9 shows that by increasing cache sizes more files will be stored in a cache, so a greater number of users can access the data from the nearest servers then the cache hit ratio will definitely increase. The results show that there is a 1% difference in delay when the cache size is 20 (GB). While increasing cache size delay is almost 10% less in our proposed technique. In reactive caching, the delay is 20% more than our proposed technique, as there is no cache and cooperation. The reason is that in COOP (CS) due to clustering and optimal route finding, there will be a minimum delay to send and receive requests.

7.3. Delay and Caching Cost While Increasing Number of Files

The given results show that caching more files on the server increases the availability of data on demand. There will be a minimum delay because files will already be cached on corresponding servers and there is no need to download files from the back-haul server.

As the number of files increases, the availability of content will be high so the hit ratio will also increase. Figure 10 shows that in reactive caching when the number of files will increase, the hit ratio will not be good due to non-caching devices, and each time data is downloaded from the back-haul server and remains the same in indifferent cases. In proactive caching, the hit ratio is 45% with an increasing number of files, while COOP

(ACO) performs much better due to its cooperative nature. However, in COOP (CS), the hit ratio is almost 90% due to joint cooperation among RSU and BS, clustering, and CS optimization algorithm.

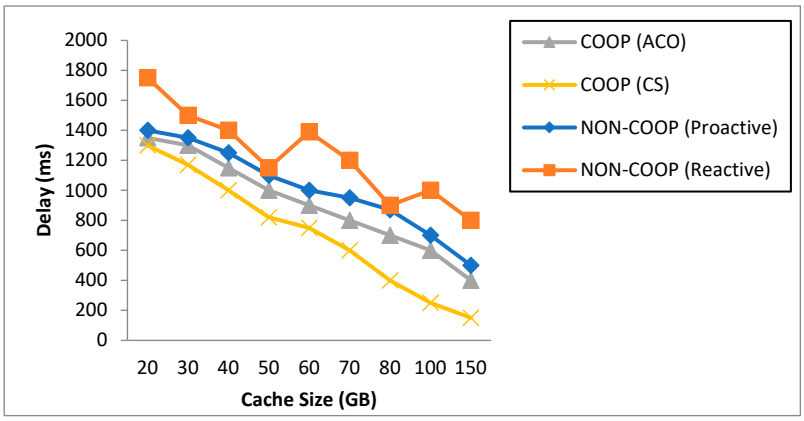

Figure 9. Delay with different cache sizes.

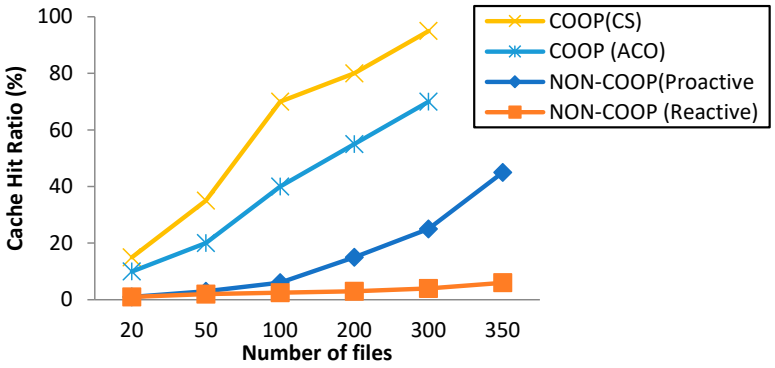

Figure 10. Cache hit ratio with increasing number of files.

If more files are available in the local cache, the time to deliver that content is less. Figure 11 shows if there are 20 files in the cache then the delay will be 10% more as compared to COOP (CS). Similarly, when there are 300 files then the delay of COOP (ACO) will be 20% more than COOP (CS). In proactive caching, the delay is double due to its non-cooperative nature. It fluctuates more in case of non-availability of content, it will fetch from the back-haul server. While reactive cannot store more files due to the unavailability of the cache.

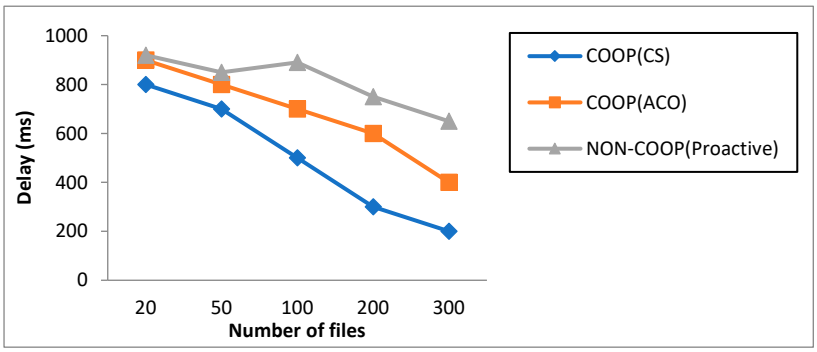

Figure 11. Delay with increasing number of files.

Figure 12 shows that by increasing the number of files in the cache, caching cost will also consume in terms of the price of BW consumption. In COOP (CS), data are

collaboratively shared among servers, so there is no need to download files again. As a result, the cost will be minimized. When there are only 20 files, the caching cost is almost the same at 12%, but when there are 300 files, the cost of COOP (ACO) is 87%, while the cost of COOP (CS) is 75%, the same as proactive caching, which is 12% costly than our proposed technique. Data are stored in clusters that are more organized and easier to maintain.

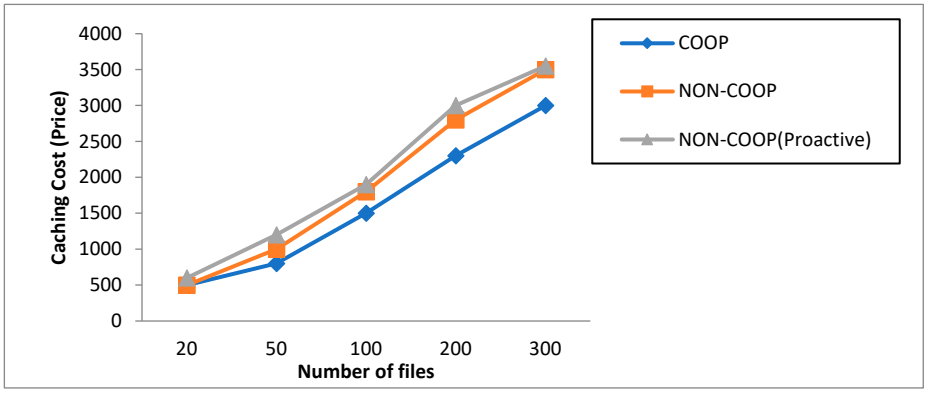

Figure 12. Caching cost with increasing number of files.

7.4. Comparison of Number of Users with Delay

In Figure 13, caching with the cuckoo search algorithm and ant colony is compared in terms of increasing the number of users with delay in finding the optimal path to reach the requesting node, where required content is cached.

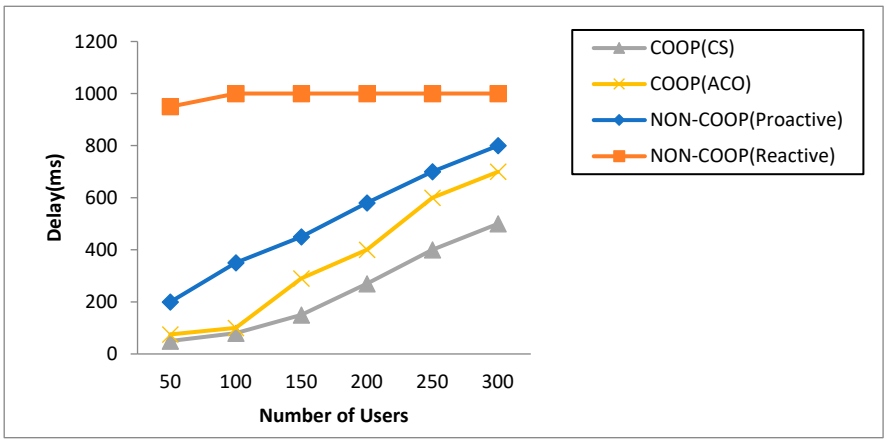

Figure 13. Delay with the increasing number of users.

The results reveal that COOP (CS) performs better than COOP (ACO). Because the convergence time of CS is more than ACO, CS takes random steps and each step is independently chosen, while ACO takes continuous and small steps, and each new step depends on the previous steps. Due to dependencies and continuous search of ACO, it takes more time to find the optimal path, so it causes an increase in delay. Cooperatively caching the popular content will also reduce the caching and delivering cost; moreover, searching data from clusters also reduces the response time while increasing the hit ratio. Overall, the proposed technique outperforms in terms of delay, hit ratio, caching, and delivery cost.

8. Conclusions

In this study, we explored content caching in vehicular edge networks to improve performance through cache utilization, increase the cache hit ratio, and reduce response time. This study indicates that caching can minimize network traffic and enhance user productivity. It has been observed that pre-caching content depending on its popularity,

increases the cache-hit ratio. The approach caches data based on the content's popularity as determined by user choices and the cooperative content caching technique. RSU anticipates the popularity of data and distributes this information with neighboring RSUs to prevent data duplication on servers nearby. RSU also encounters the node's mobility by using the Morkov model to predict it based on previous observations, distance, and time. Thus, when a user requests data not present in its local cache, it is retrieved from neighboring servers instead of downloaded over the back-haul link. In addition, the user's longitude and latitude have changed due to mobility. Therefore, RSU delivers the content to the user's true location after discovering it. We then use a meta-heuristic optimization technique to discover the fastest path to the desired material. After analyzing several optimization algorithms inspired by nature, it was concluded that CS is superior at discovering the optimal path. COOP (CS) can substantially improve cache performance, cache hit ratio, and response time. The simulated trials are conducted with actual Movielens data, and the experimental results reveal that COOP outperforms the cache hit ratio and response time in vehicle edge networks.

9. Limitations and Future Research Direction

The present technique considers the content's popularity, which boosts the cache hit ratio, but the response time is still somewhat slow. There is a lack of collaboration between caching servers before data caching. For future work, we should emphasize different types of dynamic optimization problems with randomly changed global optima and caching different types of content simultaneously with textual data. In addition, popular content is pre-cached in this study and sent to mobile consumers using optimal routing. If the content is unpopular, it is retrieved from the back-haul server, which increases the response time and service costs. Our future emphasis will be placed on caching various types of content. Those contents include movies and videos. We also intend to exploit the neural networks-based recommendation technique to predict popular content for caching purposes and compare it with our results. In addition, recurrent neural networks can also be utilized to enhance the accuracy of user location prediction.

Author Contributions: Conceptualization, S.K.u.Z., S.M. and H.A.; methodology, T.M., A.D.A. and H.E.; software, S.K.u.Z. and M.A.; validation, F.R., H.A., T.M. and M.A.K.; formal analysis, S.K.u.Z., M.A. and F.R; investigation, S.K.u.Z.; resources, H.E. and A.D.A.; data curation, M.A.K. and S.M.; writing—original draft preparation, S.K.u.Z., H.A. and M.A.; writing—review and editing, S.K.u.Z., H.A. and M.A.K.; visualization, S.M. and F.R. supervision, S.K.u.Z. and T.M.; project administration, S.K.u.Z., H.E., A.D.A. and M.A.K.; funding acquisition, A.D.A. and H.E. All authors have read and agreed to the published version of the manuscript.

Funding: Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2023R51), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No data were used to support this study. We have conducted simulations to evaluate the performance of the proposed protocol. However, any query about the research conducted in this paper is highly appreciated and can be asked from the author (Muhammad Amir Khan) upon request.

Acknowledgments: The authors appreciate the support from Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2023R51), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Conflicts of Interest: The authors declare that there is no conflict of interest.

References

1. Al-Badarneh, J.; Jararweh, Y.; Al-Ayyoub, M.; Fontes, R.; Al-Smadi, M.; Rothenberg, C. Cooperative mobile edge computing system for VANET-based software-defined content delivery. *Comput. Electr. Eng.* **2018**, *71*, 388–397. [CrossRef]
2. Hu, L.; Qian, Y.; Chen, M.; Hossain, M.S.; Muhammad, G. Proactive Cache-Based Location Privacy Preserving for Vehicle Networks. *IEEE Wirel. Commun.* **2018**, *25*, 77–83. [CrossRef]
3. Zhao, Z.; Guardalben, L.; Karimzadeh, M.; Silva, J.; Braun, T.; Sargento, S. Mobility Prediction-Assisted Over-the-Top Edge Prefetching for Hierarchical VANETs. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 1786–1801. [CrossRef]
4. Zhang, Y.; Li, C.; Luan, T.H.; Fu, Y.; Wang, H. Prediction Based Vehicular Caching: Where and What to Cache? *Mob. Netw. Appl.* **2019**, *25*, 760–771. [CrossRef]
5. Yasir, M.; Zaman, S.K.U.; Maqsood, T.; Rehman, F.; Mustafa, S. CoPUP: Content popularity and user preferences aware content caching framework in mobile edge computing. *Clust. Comput.* **2022**, 1–15. [CrossRef]
6. Qazi, F.; Khalid, O.; Bin Rais, R.N.; Khan, I.A.; Khan, A.U.R. Optimal Content Caching in Content-Centric Networks. *Wirel. Commun. Mob. Comput.* **2019**, *2019*, 6373960. [CrossRef]
7. Qiao, G.; Leng, S.; Maharjan, S.; Zhang, Y.; Ansari, N. Deep Reinforcement Learning for Cooperative Content Caching in Vehicular Edge Computing and Networks. *IEEE Internet Things J.* **2019**, *7*, 247–257. [CrossRef]
8. Zaman, S.K.U.; Jehangiri, A.I.; Maqsood, T.; Umar, A.I.; Khan, M.A.; Jhanjhi, N.Z.; Shorfuzzaman, M.; Masud, M. COME-UP: Computation Offloading in Mobile Edge Computing with LSTM Based User Direction Prediction. *Appl. Sci.* **2022**, *12*, 3312. [CrossRef]
9. Zhang, Y.; Li, C.; Luan, T.H.; Fu, Y.; Zhu, L. Caching on Vehicles: A Lyapunov Based Online Algorithm. In *International Conference on Ad Hoc Networks*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 15–24.
10. Luo, Z.; LiWang, M.; Huang, L.; Du, X.; Guizani, M. Caching mechanism for mobile edge computing in V2I networks. *Trans. Emerg. Telecommun. Technol.* **2019**, *30*, e3689. [CrossRef]
11. Mustafa, S.; Bilal, K.; Malik, S.U.R.; Madani, S.A. SLA-Aware Energy Efficient Resource Management for Cloud Environments. *IEEE Access* **2018**, *6*, 15004–15020. [CrossRef]
12. Chen, J.; Wu, H.; Yang, P.; Lyu, F.; Shen, X. Cooperative Edge Caching With Location-Based and Popular Contents for Vehicular Networks. *IEEE Trans. Veh. Technol.* **2020**, *69*, 10291–10305. [CrossRef]
13. Zaman, S.K.U.; Jehangiri, A.I.; Maqsood, T.; Haq, N.U.; Umar, A.I.; Shuja, J.; Ahmad, Z.; Ben Dhaou, I.; Alsharekh, M.F. LiMPO: Lightweight mobility prediction and offloading framework using machine learning for mobile edge computing. *Clust. Comput.* **2022**, 1–19. [CrossRef]
14. Xiao, K.; Liu, K.; Xu, X.; Feng, L.; Wu, Z.; Zhao, Q. Cooperative coding and caching scheduling via binary particle swarm optimization in software-defined vehicular networks. *Neural Comput. Appl.* **2020**, *33*, 1467–1478. [CrossRef]
15. Khelifi, H.; Luo, S.; Nour, B.; Sellami, A.; Moungla, H.; Nait-Abdesselam, F. An Optimized Proactive Caching Scheme Based on Mobility Prediction for Vehicular Networks. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; IEEE: New York, NY, USA, 2018; pp. 1–6. [CrossRef]
16. Yang, X.-S. *Nature-Inspired Optimization Algorithms*; Academic Press: Cambridge, MA, USA, 2020.
17. Wang, G. A Comparative Study of Cuckoo Algorithm and Ant Colony Algorithm in Optimal Path Problems. *MATEC Web Conf.* **2018**, *232*, 03003. [CrossRef]
18. Guo, H.; Rui, L.-L.; Gao, Z.-P. A zone-based content pre-caching strategy in vehicular edge networks. *Futur. Gener. Comput. Syst.* **2020**, *106*, 22–33. [CrossRef]
19. Su, Z.; Hui, Y.; Xu, Q.; Yang, T.; Liu, J.; Jia, Y. An Edge Caching Scheme to Distribute Content in Vehicular Networks. *IEEE Trans. Veh. Technol.* **2018**, *67*, 5346–5356. [CrossRef]
20. Kumar, N.; Lee, J.-H. Peer-to-Peer Cooperative Caching for Data Dissemination in Urban Vehicular Communications. *IEEE Syst. J.* **2013**, *8*, 1136–1144. [CrossRef]
21. Jehangiri, A.I.; Maqsood, T.; Ahmad, Z.; Umar, A.I.; Shuja, J.; Alanazi, E.; Alasmary, W. Mobility-aware computational offloading in mobile edge networks: A survey. *Clust. Comput.* **2021**, *24*, 2735–2756.
22. Ma, J.; Wang, J.; Liu, G.; Fan, P. Low Latency Caching Placement Policy for Cloud-Based VANET with Both Vehicle Caches and RSU Caches. In Proceedings of the 2017 IEEE Globecom Workshops (GC Wkshps), Singapore, 4–8 December 2017; IEEE: New York, NY, USA, 2017; pp. 1–6. [CrossRef]
23. Mustafa, S.; Sattar, K.; Shuja, J.; Sarwar, S.; Maqsood, T.; Madani, S.A.; Guizani, S. SLA-Aware Best Fit Decreasing Techniques for Workload Consolidation in Clouds. *IEEE Access* **2019**, *7*, 135256–135267. [CrossRef]
24. Ding, R.; Wang, T.; Song, L.; Han, Z.; Wu, J. Roadside-unit caching in vehicular ad hoc networks for efficient popular content delivery. In Proceedings of the 2015 IEEE Wireless Communications and Networking Conference (WCNC), New Orleans, LA, USA, 9–12 March 2015; IEEE: New York, NY, USA, 2015; pp. 1207–1212. [CrossRef]
25. Husnain, G.; Anwar, S. An intelligent cluster optimization algorithm based on Whale Optimization Algorithm for VANETs (WOACNET). *PLoS ONE* **2021**, *16*, e0250271. [CrossRef]
26. Zaman, S.K.U.; Khan, A.U.R.; Malik, S.U.R.; Khan, A.N.; Maqsood, T.; Madani, S.A. Formal verification and performance evaluation of task scheduling heuristics for makespan optimization and workflow distribution in large-scale computing systems. *Comput. Syst. Sci. Eng.* **2017**, *32*, 227–241.

27. Muniyandi, R.C.; Hasan, M.K.; Hammoodi, M.R.; Maroosi, A. An Improved Harmony Search Algorithm for Proactive Routing Protocol in VANET. *J. Adv. Transp.* **2021**, *2021*, 6641857. [CrossRef]
28. Glass, S.C. Improving Privacy with Intelligent Cooperative Caching in Vehicular Ad Hoc Networks. Ph.D. Thesis, Florida Atlantic University, Boca Raton, FL, USA, 2017.
29. Zaman, S.K.U.; Maqsood, T.; Ali, M.; Bilal, K.; Madani, S.A.; Khan, A.U.R. A Load Balanced Task Scheduling Heuristic for Large-Scale Computing Systems. *Comput. Syst. Sci. Eng.* **2019**, *34*, 79–90. [CrossRef]
30. Li, L.; Chan, C.A.; Erfani, S.; Leckie, C. Adaptive Edge Caching based on Popularity and Prediction for Mobile Networks. In Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14–19 July 2019; IEEE: New York, NY, USA, 2019; pp. 1–10. [CrossRef]
31. Mukhairez, H.H.; Maghari, A.Y. Performance comparison of simulated annealing, GA and ACO applied to TSP. *Int. J. Intell. Comput. Res.* **2015**, *6*, 647–654. [CrossRef]
32. Ab Wahab, M.N.; Nefti-Meziani, S.; Atyabi, A. A Comprehensive Review of Swarm Optimization Algorithms. *PLoS ONE* **2015**, *10*, e0122827. [CrossRef] [PubMed]
33. Safi, A.; Ahmad, Z.; Jehangiri, A.I.; Latip, R.; Zaman, S.K.U.; Khan, M.A.; Ghoniem, R.M. A Fault Tolerant Surveillance System for Fire Detection and Prevention Using LoRaWAN in Smart Buildings. *Sensors* **2022**, *22*, 8411. [CrossRef]
34. Bello-Salau, H.; Onumanyi, A.J.; Abu-Mahfouz, A.M.; Adejo, A.O.; Mu'Azu, M.B. New discrete cuckoo search optimization algorithms for effective route discovery in IoT-based vehicular ad-hoc networks. *IEEE Access* **2020**, *8*, 145469–145488. [CrossRef]
35. Bhavana, V.; Ramesh, V.; Sivagami, M. Implementing Discrete Cuckoo Search Algorithm for TSP using MPI and Beowulf Cluster. *Int. J. Innov. Technol. Explor. Eng.* **2019**, *8*, 554–560.
36. Zheng, H.; Zhou, Y. A novel cuckoo search optimization algorithm based on Gauss distribution. *J. Comput. Inf. Syst.* **2012**, *8*, 4193–4200.
37. Hajlaoui, R.; Alaya, B.; Mchergui, A. Optimized VANET Routing Protocol Using Cuckoo Search Algorithm. In Proceedings of the 2022 International Wireless Communications and Mobile Computing (IWCMC), Dubrovnik, Croatia, 30 May–3 June 2022; IEEE: New York, NY, USA, 2022; pp. 824–828.
38. Marwah, G.P.K.; Jain, A.; Malik, P.K.; Singh, M.; Tanwar, S.; Safirescu, C.O.; Mihaltan, T.C.; Sharma, R.; Alkhayyat, A. An Improved Machine Learning Model with Hybrid Technique in VANET for Robust Communication. *Mathematics* **2022**, *10*, 4030. [CrossRef]
39. Behura, A.; Srinivas, M.; Kabat, M.R. Giraffe kicking optimization algorithm provides efficient routing mechanism in the field of vehicular ad hoc networks. *J. Ambient Intell. Humaniz. Comput.* **2022**, *13*, 3989–4008. [CrossRef]
40. Lv, B.; Wang, Z.; Huang, T.; Chen, J.; Liu, Y. A hierarchical virtual resource management architecture for network virtualization. In Proceedings of the 2010 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM), Chengdu, China, 23–25 September 2010; IEEE: New York, NY, USA, 2010; pp. 1–4.
41. Harper, F.M.; Konstan, J.A. The movielens datasets: History and context. *Acm Trans. Interact. Intell. Syst. (Tiis)* **2015**, *5*, 1–19. [CrossRef]
42. Jiang, Y.; Ma, M.; Bennis, M.; Zheng, F.-C.; You, X. User Preference Learning-Based Edge Caching for Fog Radio Access Network. *IEEE Trans. Commun.* **2018**, *67*, 1268–1283. [CrossRef]