A Genetic Algorithm for Task Offloading problem in Vehicular Edge Computing

Han Li
School of Information Science and Engineering
Yunnan University
Kunming,China
lihan@mail.ynu.edu.cn

Weidong Li
School of Mathematics and Statistics
Yunnan University
Kunming,China
weidongmath@126.com

Xuejie Zhang*
School of Information Science and Engineering
Yunnan University
Kunming,China
xjzhang@ynu.edu.cn

Abstract—In this paper, we consider task offloading in vehicular edge computing systems. The limited battery limits the mileage of electric vehicles. For a longer driving distance, the task of the vehicle can not only be performed locally, but also offload task to save energy . In this way,the energy consumption of the system is reduced and the energy utilization rate of the whole system is improved. The task offloading involves optimization variables which are where to offload tasks and the frequency, And The optimization variables are coupled. So it hard to decide how to allocate tasks. We propose that we can solve this problem by using genetic algorithm. Specifically, our experiment shows between 20% and 25% energy savings compared to the baseline which the tasks executes workload locally.

Index Terms—Vehicular Edge Computing; Task Offloading; Genetic Algorithm

I. INTRODUCTION

Clean energy is an important trend in the future. Electric energy is the representative of clean energy, so more and more people choose electric vehicles. Electric intelligent vehicles will be an integral part of urban transportation in the future. However, before major changes in battery technology, battery capacity is the shortcoming of electric intelligent vehicles[1]. Limited battery capacity will cause "range anxiety". Many methods have been proposed to solve the mileage problem, such as considering complete vehicle energy management[2]. The vehicular coasts is part of driving mode, thereby reducing energy consumption in[3].

Because of the large amount of data processed in the driving, the central processing unit(CPU) will consume considerable energy, which have a great impact on the driving mileage. Some researchers proposed that send the task to mobile edge computing servers to save energy. But if all data is uploaded to cloud server for processing, the response time might be too long, and the requirements for low delay will not be met[4]. In paper[5], Liu et al.propose a system to guarantee task low delay performance. In addition, the current bandwidth and storage simply cannot satisfy the requirements for transmitting all data.

To solve this problem, Yu et al.[6] focus on virtual machine resource allocation in vehicular edge clouds. Liu et al.[7] consider dynamic requirements and resource constraints, and classify all tasks into four types of pending lists. Then the tasks in each list will be offloaded to different nodes according to their features.

The recently popular edge computing technology is introduced. We can not only assign tasks to the nearby edge server, but also to resource rich users nearby. We propose to assign tasks to nearby vehicles to minimize energy consumption of the system. This offloading process forms a three-tier architecture, including vehicular cloud, edge servers which is roadside units, and central cloud processors. The edge servers are the controller of vehicular cloud. And the central cloud is responsible for global scheduling.

The problem is a non-linear integer programming. We propose a genetic algorithm to solve the task offloading problem. The experimental results show that when the number of vehicles is less than 30, the energy saving is obvious.

The remainder of this paper is organized as follows. Section II introduces the system model and problem formulation. Section III describes the genetic algorithm. Section IV describe our experiment and analysis the result. Section V is conclusion and future work.

II. SYSTEM MODEL AND PROBLEM FORMULATION

We first describe the our system, including network artechiture and computation models in this section.

A. Network Artechiture

Similar to[6], as shown in Fig.1, the hierarchical architecture consisting vehicular cloud, edge cloud, and central cloud.

The central cloud is responsible for global scheduling, and the tasks it performs include complicated computation and global decision.

The edge cloud is the controller of vehicular cloud, and is responsible for creating, maintaining, and deleting vehicular cloud.

The vehicular cloud consists of vehicles which share their computation resources. Each vehicle can access the edge cloud and convey the information to it. The vehicles

Fig. 1: Vehicular networks architecture

TABLE I: Notations in This Paper

Parameter	Interpretation
T	The resource sharing time
C_{it}	The CPU capacity
r_i	The workload of vehicle i
r'_{jt}	The resource requester j need
M_{it}	Millions Instructions Per Second(MIPS)
θ_i	Estimated parameters
E_{it}^{blnc}	Energy consumed by the vehicle i
E_i^{rec}, E_i^{send}	Energy received or send by the vehicle i

can save the energy through offloading task to other vehicles. It can improve the overall resource utilization.

The task offloading process is as follows: first, the real-time traffic information is transmitted to the cloud server for analysis. The real-time information includes destination, current location, time, vehicle speed, and so on. After summarizing the data, the cloud server performs big data analysis to obtain the road congestion, and then infers the location information of the vehicles in a certain time period T in the future, and returns the result to the edge cloud which is near the vehicle. These vehicles will participate in resource sharing. Then, the edge cloud assigns tasks according to these information and divides the vehicles into providers and requesters, and they share resources at time period T.

B. Computation Model

We define resource sharing time as $\mathcal{T} = \{1, \dots, T\}$. And the number of vehicles is defined as N. We use a tuple $J_{it} = \{r_{it}, r'_{it}, d_{it}\}$ to represent the task and bandwith of vehicle i at time slot t. d_{it} is the task's data size.

Tasks can be divided into tasks that can be offloaded and tasks that cannot be offloaded and must be executed locally. At time slot t, r_{it} is task required to complete the local by the vehicle i, r'_{it} is task which can offload to others in the vehicle i,and their values are expressed in the number of instructions.

To facilitate the description of the model, we denote $\mathbf{X}_t = \{x_{ij}\} \in \{0,1\}^{N\times N}$ allocation matrix. If $x_{ij} = 1$, it represents that vehicle i performs the task of offloading vehicle j. Note that if $x_{ii} = 1$, all tasks of vehicle i are executed locally.

We characterize the computing capacity of vehicle i by C_{it} at time t. When a vehicle is selected as a provider (P) at time t, the computing resources of all requester should be less than its computing capacity.

$$\sum_{j=1}^{N} x_{ij}^{t} \cdot r'_{jt} \le C_{it}, i = 1, \dots, N \tag{1}$$

A crucial step is how to quantify the computing resources. The computing capacity is defined based on the number of instructions executed per unit time (MIPS, M_{it}).

$$C_{it} = M_{it} \cdot \Delta T - r_{it} \tag{2}$$

where ΔT is the duration of resource sharing, and is a smaller time scale than time of establishing vehicular network. After ΔT seconds, the offloading matrix changes.

For a single core CPU, the number of instructions executed per unit time (m_{it}) and CPU frequency (f_{it}) have the following relationship:

$$M_{it} = v_i \cdot f_{it} + \theta_i \tag{3}$$

Where v_i and θ_i are parameters to be estimated.

As a result, the CPU capacity C_{it} in Equation (2) can be calculated following:

$$C_{it} = (v_i \cdot f_{it} + \theta_i) \times \Delta T - r_{it} \tag{4}$$

C. Energy Consumption Model

When we consider the energy consumption, we divide it into two parts: (1)the energy required for calculation, (2)the energy required for transmission.

According to [8][9][10][11], the energy consumption is computed as follows:

$$E = \lambda_i \cdot f_{it}^3 \cdot \Delta T \tag{5}$$

where f_{it} is the frequency of the CPU in vehicle i at time slot t. If a vehicle is selected as the requester(R), its frequency will decrease to f'_{it}. Because its tasks are assigned, so the energy saved is:

$$E_i^{save} = (\lambda_i \cdot f_{it}^3 - \lambda_i \cdot {f'}_i^3) \times \Delta T, \forall i \in R \tag{6}$$

The transmission energy consumption is linear with the transmission time which depends on the ratio between the data size and the data transmission rate(b_{ij}).

$$E = P_0 \cdot \frac{d_{it}}{b_{ij}} \tag{7}$$

where P_0 is the transmission power.

Maximum data transmission rate(b) is given by Shannon theorem.

$$b_{ij} = W \log(1 + \mathrm{SNR}) \tag{8}$$

where SNR is Signal-to-noise Ratio, and W is the channel bandwidth. Because they are related to each actual channel, we regard it as a constant in this paper.

For each vehicle, the energy required for receiving information is:

$$E_i^{rec} = \sum_{j=1,j\neq i}^{N} x_{ij}^t \cdot P_0 \cdot \frac{d_{jt}}{W \log(1+\mathrm{SNR})}, \quad i = 1, \ldots, N \tag{9}$$

For each vehicle, the energy required for sending information is:

$$E_i^{send} = \sum_{i=1,i\neq j}^{N} x_{ij}^t \cdot P_0 \cdot \frac{d_{it}}{W \log(1+SNR)}, \quad j = 1, \ldots, N \tag{10}$$

Define E_t^{blnc} is all the energy consumption of the vehicle in the time slot t .

$$\begin{aligned} E_{it}^{blnc} = &\sum_{j=1,j\neq i}^{N} x_{ij}^t \cdot P_0 \cdot \frac{d_{jt}}{W \log(1+\mathrm{SNR})} \\ &+ \sum_{j=1,,j\neq i}^{N} x_{ji}^t \cdot P_0 \cdot \frac{d_{it}}{W \log(1+\mathrm{SNR})} \\ &+\lambda_i \cdot f_{it}^3 \cdot \Delta T, \quad i = 1, \ldots, N \end{aligned} \tag{11}$$

The objective of our algorithm is to minimize the quadratic power of energy over all vehicles. The goal is to balance and minimize energy consumption, because the square of the minimum drawing energy consumption.

$$\min \sum_{t=1}^{T} \sum_{i=1}^{N} (E_{it}^{blnc})^2 \tag{12}$$

The problem can be formulated as:

$$\min \quad \sum_{t=1}^{T} \sum_{i=1}^{N} (E_{it}^{blnc})^2 \tag{13}$$

$$\begin{aligned} \text{s.t.} \quad E_{it}^{blnc} = &\sum_{j=1,j\neq i}^{N} x_{ij}^t \cdot P_0 \cdot \frac{d_{jt}}{W \log(1+\mathrm{SNR})} \\ &+ \sum_{j=1,j\neq i}^{N} x_{ji}^t \cdot P_0 \cdot \frac{d_{it}}{W \log(1+\mathrm{SNR})} \\ &+ \lambda_i \cdot f_{it}^3 \cdot \Delta T, \quad i = 1, \ldots, N \end{aligned} \tag{14}$$

$$\sum_{j=1}^{N} x_{ij}^t \cdot r'_{jt} \leq C_{it} \tag{15}$$

$$\sum_{i=1}^{N} x_{ij}^t \geq 1 \tag{16}$$

$$f_{it} \geq 0 \tag{17}$$

$$x_{it}^t \in \{0, 1\} \tag{18}$$

As mentioned above, the Equation(15) shows all the amount of requested cannot exceed available capacity of vehicle i. The Equation(16) shows that the task of vehicle i which can be offloaded must be executed whether by the vehicle i or others. f is the frequency and must be positive number.

D. Example

$$\mathbf{X} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The $\mathbf{X}$ is a example of allocation matrix. Vehicular set is {A,B,C,D,E}, the vehicle A, B and D don't assign and load other vehicular tasks, Therefore, they do not generate transmission energy consumption. And vehicle C execute the offloading task from vehicle E. And the sum of each column is greater than or equal to one.

E. Constraints

To ensure the Quality of Service(QoS) between the requester and provider, the following distance constraints need to be met:

$$\min l_{ij}^t < \delta, \quad i = 1, \ldots, N, \forall x_{ij}^t = 1 \tag{19}$$

where l_{ij}^t is the distance between vehicle i and j.

If the constraint is not satisfied, the vehicle will exit resource sharing, and the ith row and the ith column of the allocation matrix are all set to zero while $x_{ii}^t = 1$

III. ALGORITHMS

We design a greedy algorithm to minimize the energy consumption of all time slot. and we use genetic algorithm1 to calculate the minimum value of each moment.

Genetic algorithm is a adaptive heuristic optimization method, which is proposed by John Holland in 1970s[12]. Based on Darwin's theory of natural selection, it simulates the process of natural selection and reproduction to solve optimization problems. Similarly to individuals with favorable variation surviving, the solutions in the algorithm imitate the behavior of chromosomes, such as the mutations of genes and the crossovers of chromosomes.

In recently, the genetic algorithm has been used for real-time path planning[13], maximum coverage deployment in Wireless Sensor Networks[14], and intrusion detection[15].

In genetic algorithm, a population is composed of randomly generated solutions, and all individuals in the population are composed of encoded strings similar to chromosomes[16]. Similar to natural selection, with the iteration, the diversity decreases, but the preserved individuals are excellent adaptive individuals. In other words, the preserved solutions are are locally optimal.

Algorithm 1 is the pseudo-code of genetic algorithm in our experiment. The algorithm has as input the vector of

Algorithm 1: Genetic Algorithm

Input: input parameters $T, M, r', rloc$,
Output: $\mathbf{X}, f, E$
1 $t = 0$;
2 Initialize $\mathbf{X}(0), f(0)$;
3 while $t < T$ do
4 Encoder $\mathbf{X}(t), f(t)$;
5 for i = 1 to M do
6 Evaluate fitness $P(ti) = -E_i$;
7 end
8 Eliminate fitness smaller individuals from $P(t)$;
9 Replicate the fitness larger individuals from $P(t)$;
10 Divide the array composed of optimization variables into two. ;
11 for i = 1 to M/2 do
12 Crossover operation to $P(t)$;
13 end
14 for i = 1 to M do
15 Mutation operation ;
16 end
17 Decoder $\mathbf{X}(t), f(t)$;
18 $t \leftarrow t + 1$;
19 end
20 for i = 1 to N do
21 $E \leftarrow E_i + E$
22 end
23 $\mathbf{X} = \mathbf{X}(t), f = f(t)$;
24 return $\mathbf{X}, f, E$

vehicles with their request size, r'_i, and their local workload $rloc$. The output consists of the allocation matrix $\mathbf{X}_t$, the frequency f_t, and the energy consumption E_t in the current time. At the initial stage of the algorithm, it will randomly generate a set of feasible solutions which is the first generation of chromosomes(Line2). After that, $\mathbf{X}(t)$ and $f(t)$is encoded as a string like a chromosome. Then the fitness function is used to calculate the fitness degree for each chromosome(Line5-7), and the probability of each chromosome being selected in the next evolution is calculated according to the fitness degree(Line8).The chromosomes with lower fitness will be eliminated, and only the chromosomes with higher fitness will be retained(Line9). Divide the individual into two parts at random(Line10). A certain position of these two chromosomes is cut off and spliced together to generate a new chromosome. As a result, this new chromosome contains a certain number of genes of both father and mother(Line11-13).

The solution is easier to get the local optimal solution, and there is no way to achieve the global optimal solution. In order to solve this problem, we need to introduce mutation(Line14-16). The next step is to decode the string to the original solution(Line17). Finally we calculate all the energy consumption(Line20-22).

IV. EXPERIMENT

We set the parameters of the experiment and analyzed the result of the experiment in the section.

A. The Setup

As we all known, $U[x, y]$ is the uniform distribution between x and y, and $N(x, y)$ is the normal distribution which mean is x and variance is y. We assume that the time which vehicles share their resource is $T = 10$ seconds and $\Delta T = 1$ second, because of their limited space.

The Cortex-A57 processor is ARM's highest performing processor, designed to further mobile and enterprise computing applications[17]. The Cortex-A57 has the frequency from 700 MHz to 1900 MHz. The frequency set is $\{700, 800, 900, , , 1900\}$, as a result, there is only 13 frequency levels.

As is shown in Equation (3), the number of instructions is $M_{it} = v_i \cdot f_{it} + \theta_i$. And the CPU capacity in Equation (4) is $C_{it} = (v_i \cdot f_{it} + \theta_i) \times \Delta T - r_{it}$,

We estimate the CPU capacity parameters v_i, θ_i is based on the analysis provided in [18], and the value is 7.683 and 4558.52. After calculation, the maximum and minimum M values are respectively 819 and 10039. The required computing resource r_{it} for all tasks to execute workloads is uniformly drawn from $[820, 10000]$, and we assume that the task computing resource value r'_{it} which vehicle i can offload to others is $U[200, r_{it}]$, because some subtask must be execute locally. In our experiment the size of data varies from 1MB to 10 MB. And we set the value of bandwith d is 27Mb/s[19].

We have selected seven kinds of vehicle number to participate in resource sharing, which are$\{10, 15, 20, 25, 30, 35, 40\}$. The population number was set to 40 in algorithm. While the number is set to 150 when the number of vehicles is 35 and 40, because it is too small and will not converge.

Our experiment are implemented in python and executed on an Intel Core i7 with 8 GB RAM.

B. Performance Metrics

The performance of our experiment is the percentage of energy saving,which is define as follows:

$$P = \min(100 \cdot (1 - \frac{\sum_{i=1}^{N} E_{it}^{blnc}}{E_{loc}})), t = 1, \ldots, T \tag{20}$$

where E_{loc} is the energy consumed by local task execution.

In order to reflect the fairness of our framework, we define the fairness coefficient (FC) based on the standard deviation As with the standard deviation, the smaller the value the greater the fairness. FC is define as follows.

$$FC = \max \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(E_{it}^{blnc} - \bar{E}\right)^2}}{\bar{E}}, t = 1, \ldots, T \tag{21}$$

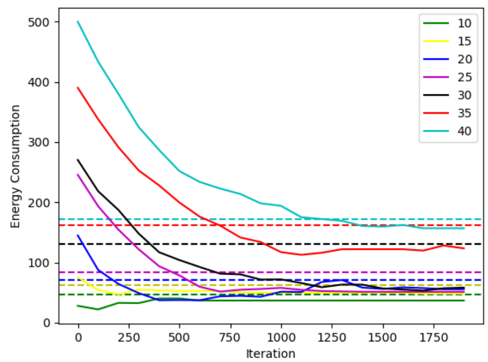


Fig. 2: Energy Consumption

C. Experimental Results

In Fig.2, the dotted line is the energy consumed by local execution. We can see that when the number of vehicles is 10, the algorithm can get good results by running a few times. While the number of vehicles is 20, it takes 350 iterations to get a better result. And when the number of vehicles is 30, the reduction rate of energy consumption is faster before 750 iterations, and will be significantly slower after 750 iterations . Almost no energy saving when the number of vehicles is 40.

We can conclude that with the increase of the number of vehicles, the number of iterations is also increasing. Therefore, when we run the algorithm on the edge server, we need to adjust the number of iterations reasonably as the number of vehicles changes.

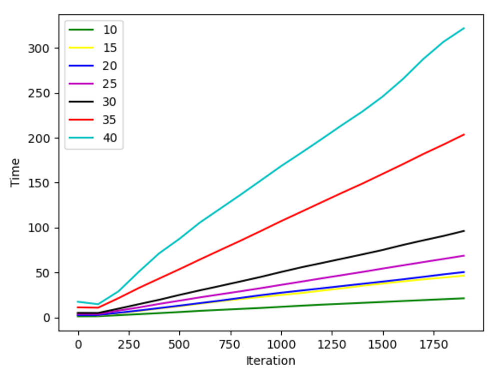
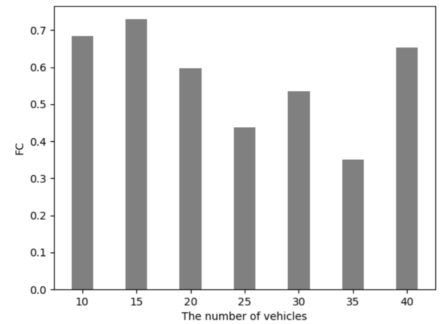


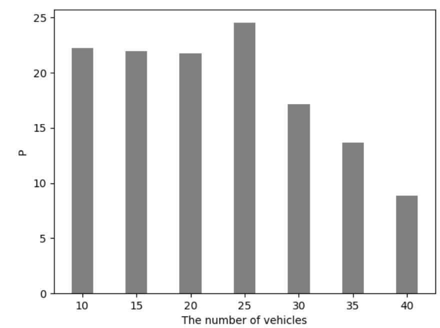
Fig. 3: Experimental running time

In Fig.3, after 100 iterations, the running time and the number of iterations show a linear relationship. However, when the number of iterations is fixed, the relationship between time and vehicle is not linear.

As Fig.4(a) shows, we can see that with the increasing of vehicles, the balance of vehicle energy consumption increases first and then decreases. As Fig.4(b) shows, when the number of vehicles is between 10 and 25, our system can save over 20% the energy consumption, but when the number of vehicles is 30, only 17% can be saved, and when the number of vehicles is 40, almost no savings.



(a) The values of FC



(b) The values of P

Fig. 4: Performance metrics

Considering the performance measurement, energy consumption and time, it is appropriate to choose 25 vehicles to participate in resource sharing.

V. CONCLUSIONS AND FUTURE

In this paper, we proposed task offloading based on genetic algorithm for electric smart vehicle. We evaluate the performance of the algorithm through simulation experiments. Experimental results show that our algorithm can achieve the goal of energy conservation. In the future research, we plan to establish the relationship between the data size and the number of instructions and consider the deadline constraints.

References

[1] Zachary P Cano et al. “Batteries and fuel cells for emerging electric vehicle markets”. In: Nature Energy 3.4 (2018), pp. 279–289.

[2] T. Constantijn J. Romijn et al. “A Distributed Optimization Approach for Complete Vehicle Energy Management”. In: IEEE Transactions on Control Systems Technology 27.3 (2019), pp. 964–980. doi: 10.1109/TCST.2018.2789464.

[3] Dalibor Barta et al. "Possibility of increasing vehicle energy balance using coasting". In: Advances in Science and Technology. Research Journal 12.1 (2018).

[4] Virgilios Passas et al. "V2MEC: Low-Latency MEC for Vehicular Networks in 5G Disaggregated Architectures". In: 2021 IEEE Wireless Communications and Networking Conference (WCNC). IEEE. 2021, pp. 1–6.

[5] Chen-Feng Liu et al. "Dynamic Task Offloading and Resource Allocation for Ultra-Reliable Low-Latency Edge Computing". In: IEEE Transactions on Communications 67.6 (June 2019). Conference Name: IEEE Transactions on Communications, pp. 4132–4150. issn: 1558-0857. doi: 10.1109/TCOMM.2019.2898573.

[6] Rong Yu et al. "Toward cloud-based vehicular networks with efficient resource management". In: Ieee Network 27.5 (2013), pp. 48–55.

[7] Chunhui Liu et al. "Adaptive Offloading for Time-Critical Tasks in Heterogeneous Internet of Vehicles". In: IEEE Internet of Things Journal 7.9 (2020), pp. 7999–8011. doi: 10.1109/JIOT.2020.2997720.

[8] R. Ge, Xizhou Feng, and K.W. Cameron. "Performance-constrained Distributed DVS Scheduling for Scientific Applications on Power-aware Clusters". In: SC '05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing. 2005, pp. 34–34. doi: 10.1109/SC.2005.57.

[9] E. N. (Mootaz) Elnozahy, Michael Kistler, and Ramakrishnan Rajamony. "Energy-Efficient Server Clusters". In: Power-Aware Computer Systems. Ed. by Babak Falsafi and T. N. Vijaykumar. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 179–197. isbn: 978-3-540-36612-6.

[10] J. Li and J.F. Martinez. "Dynamic power-performance adaptation of parallel computation on chip multiprocessors". In: The Twelfth International Symposium on High-Performance Computer Architecture, 2006. 2006, pp. 77–87. doi: 10.1109/HPCA.2006.1598114.

[11] C. Piguet, C. Schuster, and J.-L. Nagel. "Optimizing architecture activity and logic depth for static and dynamic power reduction". In: The 2nd Annual IEEE Northeast Workshop on Circuits and Systems, 2004. NEWCAS 2004. 2004, pp. 41–44. doi: 10.1109/NEWCAS.2004.1359011.

[12] Jeffrey R. Sampson. "Adaptation in Natural and Artificial Systems (John H. Holland)". In: SIAM Review 18.3 (1976), pp. 529–530. doi: 10.1137/1018105. eprint: https://doi.org/10.1137/1018105. url: https://doi.org/10.1137/1018105.

[13] Vincent Roberge, Mohammed Tarbouchi, and Gilles Labonté. "Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning". In: IEEE Transactions on industrial informatics 9.1 (2012), pp. 132–141.

[14] Yourim Yoon and Yong-Hyuk Kim. "An Efficient Genetic Algorithm for Maximum Coverage Deployment in Wireless Sensor Networks". In: IEEE Transactions on Cybernetics 43.5 (2013), pp. 1473–1483. doi: 10.1109/TCYB.2013.2250955.

[15] Anand Kannan et al. "Genetic algorithm based feature selection algorithm for effective intrusion detection in cloud networks". In: 2012 IEEE 12th International Conference on Data Mining Workshops. IEEE. 2012, pp. 416–423.

[16] R Sivaraj and T Ravichandran. "A review of selection methods in genetic algorithm". In: International journal of engineering science and technology 3.5 (2011), pp. 3792–3797.

[17] Cortex-A57. url: https://developer.arm.com/Processors/Cortex-A57 (visited on 07/20/2022).

[18] Tayebeh Bahreini, Marco Brocanelli, and Daniel Grosu. "VECMAN: A Framework for Energy-Aware Resource Management in Vehicular Edge Computing Systems". In: IEEE Transactions on Mobile Computing (2021), pp. 1–1. doi: 10.1109/TMC.2021.3089338.

[19] Zhigang Xu et al. "DSRC versus 4G-LTE for connected vehicle applications: A study on field experiments of vehicular communication performance". In: Journal of advanced transportation 2017 (2017).