A Genetic Algorithm for Task Offloading problem in Vehicular Edge Computing

*Note: Sub-titles are not captured in Xplore and should not be used

1st Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

3rd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***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 save energy by offloading the task. In this way, the energy utilization rate of the whole system is improved and the energy consumption of the system is reduced. 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[3]. Limited battery capacity will cause "range anxiety". Many methods have been proposed to solve the mileage problem, such as considering complete vehicle energy management[15]. The vehicular coasts is part of the driving mode, thereby reducing energy consumption in[2].

Because of the large amount of data processed in the process of driving, the central processing unit(CPU) will consume considerable energy, which will 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[12]. In paper[10], 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.

Identify applicable funding agency here. If none, delete this.

To solve this problem, 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. Yu et al.[20] focus on virtual machine resource allocation in vehicular edge clouds. Liu et al.[11] 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.

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[20], 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.

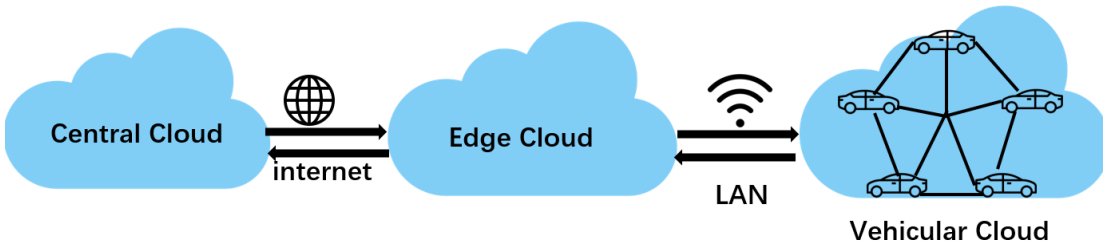


Fig. 1: Vehicular networks architecture

Due to vehicular mobility, the edge cloud is composed of two parts: local server and communication site. And the edge cloud only serve nearby vehicles. The edge cloud is the controller of vehicular cloud, and is responsible for creating, maintaining, and deleting the 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 can save the energy through offloading task to other vehicles. It can improve the overall resource utilization.

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 vehicle in a certain time period T in the future, and returns the information to the edge cloud which is near the vehicle. The information includes the set of vehicles which are closed to the vehicle within the range of time T, 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 the vehicles share resources at time slot T.

B. Constraints

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

$$\min_{j \in S_{it}} l_{ij} < \delta, i \in P \tag{1}$$

where l_{ij} is the distance between vehicle i and j. If the constraint is not satisfied, the vehicle will exit resource sharing.

C. Computation Model

We define resource sharing time as $\mathcal{T} = \{1, \ldots, T\}$. 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 of vehicle i at time slot t. Tasks can be divided into tasks that can be offloaded and tasks that cannot be offloaded and must be executed locally. r_{it} is the number of instructions to complete the local task required by the vehicle i at time slot t, r'_{it} is the number of instructions which the vehicle i task can offload to others at time slot t, and d_{it} is the task's data size.

TABLE I: Notations in This Paper

Parameter	Interpretation
P	The provider
S_{it}	The provider i's requester
V	The set of vehicles
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

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.

$$\sum_{j=1}^{N} x_{ij} \cdot r'_{jt} \leq C_{it}, i = 1, \ldots, N \tag{2}$$

where $r'_{jt}$is the computing resource that the requester needs at time slot t, s_{it} is the set of customers of the provider at time slot t.

A crucial step is how to quantify the computing resources. We regard the number of instructions that can be executed per unit time as the computing resources of a vehicle.The calculation capacity is defined based on the number of instructions executed per unit time (MIPS, M_{it}).

$$C_{it} = M_{it} \cdot T_s - r_{it} \tag{3}$$

where T_s is the duration of resource sharing, and is a smaller time scale than time of establishing vehicular network. After T_s seconds, the offloading method 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:

$$M_{it} = v_i \cdot f_{it} + \theta_i \tag{4}$$

Where v_i and θ_i are parameters to be estimated. As a result, the CPU capacity C_{it} in Equation (3) can be calculated following:

$$C_{it} = (v_i \cdot f_{it} + \theta_i) \times T_s - r_{it} \tag{5}$$

D. Energy Consumption Model

When we consider the energy consumtion, we divide it into two parts: (1) the energy required for calculation, (2) the energy required for send or receive. In this paper[8], the energy consumption of processor contain dynamic energy consumption $E^{dynamic}$ and static energy consumption E^{static}. According to [6], the dynamic energy consumption is computed as follows:

$$E^{dynamic} = \alpha \cdot C \cdot V^2 \cdot f \cdot t \tag{6}$$

where α is a factor, C is the total capacitance load, V is the supply voltage, f is the processor frequency, t is a time period. E is proportional to $E^{dynamic}$, according to [9] [13].

$$E = \beta \cdot E^{dynamic} \tag{7}$$

where β is a factor. Voltage is a linear function of the frequency[5].

$$f = \gamma \cdot V \tag{8}$$

As a result, the energy consumption is the cubic function of frequency.

$$E = \lambda_i \cdot f_{it}^3 \cdot T_s \tag{9}$$

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:

$$E_i^{save} = (\lambda_i \cdot f_{it}^3 - \lambda_i \cdot {f'}_i^3) \times T_s, \forall i \in R \tag{10}$$

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}).

$$E = P_0 \cdot \frac{d_{it}}{b_{ij}} \tag{11}$$

where P_0 is the transmission power.

Data transmission rate(b) is given by Shannon theorem.

$$b = W \log(1 + \text{SNR}) \tag{12}$$

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

$$E_i^{rec} = \sum_{j=1, j\neq i}^{N} x_{ij} \cdot P_0 \cdot \frac{d_{jt}}{W \log(1 + \text{SNR})}, \quad i = 1, \ldots, N \tag{13}$$

SNR is Signal-to-noise Ratio, and W is the channel bandwidth.

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

$$E_i^{send} = \sum_{i=1, i\neq j}^{N} x_{ij} \cdot P_0 \cdot \frac{d_{it}}{W \log(1 + SNR)}, \quad j = 1, \ldots, N \tag{14}$$

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

$$\begin{aligned} E_{it}^{blnc} = & \sum_{j=1, j\neq i}^{N} x_{ij} \cdot P_0 \cdot \frac{d_{jt}}{W \log(1 + \text{SNR})} \\ & + \sum_{j=1, , j\neq i}^{N} x_{ji} \cdot P_0 \cdot \frac{d_{it}}{W \log(1 + SNR)} \\ & + \lambda_i \cdot f_{it}^3 \cdot T_s, \quad i = 1, \ldots, N \end{aligned} \tag{15}$$

The objective of the framework is to minimize the quadratic power of energy over all vehicles.

$$\min \sum_{t=1}^{T} \sum_{i=1}^{N} (E_{it}^{blnc})^2 \tag{16}$$

The problem can be formulated as:

$$\min \quad \sum_{t=1}^{T} \sum_{i=1}^{N} (E_{it}^{blnc})^2 \tag{17}$$

$$\begin{aligned} \text{s.t.} \quad E_{it}^{blnc} = & \sum_{j=1, j\neq i}^{N} x_{ij} \cdot P_0 \cdot \frac{d_{jt}}{W \log(1 + \text{SNR})} \\ & + \sum_{j=1, j\neq i}^{N} x_{ji} \cdot P_0 \cdot \frac{d_{it}}{W \log(1 + SNR)} \\ & + \lambda_i \cdot f_{it}^3 \cdot T_s, \quad i = 1, \ldots, N \end{aligned} \tag{18}$$

$$\sum_{j=1}^{N} x_{ij} \cdot r'_{jt} \leq C_{it} \tag{19}$$

$$\sum_{i=1}^{N} x_{ij} \geq 1 \tag{20}$$

$$f_{it} \geq 0 \tag{21}$$

$$x_{it} \in \{0, 1\} \tag{22}$$

As mentioned above, the Equation(19) shows all the amount of requested cannot exceed available capacity of vehicle i. The Equation(20) 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.

E. Example

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

Algorithm 1: Greedy algorithm

Input: Vehicular real-time information
while *true* **do**
  Update set of vehicles allocated to a vehicular network ;
  for *t =1 to T/T_s* **do**
    for *each edge server* **do**
      Establish vehicular network as controller ;
      Run GA($Iterations, M, r', rloc,$) ;
      Every vehicle take participate in resource sharing according to the allocation matrix ;
    end
  end
end

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.

III. ALGORITHMS

We design a greedy algorithm 1 to minimize the energy consumption of all time slot. For each moment, the central cloud returns the vehicles contained in each edge cloud and the time of participating in resource sharing, and all the edge cloud forms a vehicular cloud, then run genetic algorithm to minimize the energy consumption of a time slot.

Genetic algorithm is a adaptive heuristic optimization method, which is proposed by John Holland in 1970s[16]. 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[14], maximum coverage deployment in Wireless Sensor Networks[19], and intrusion detection[7].

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[17]. 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.

We develop a genetic algorithm called GA-Veh. Algorithm 1 is the pseudo-code of GA-Veh in our experiment. The algorithm has as input the vector of vehicles with their request size, r'_i, and their local workload *rloc*. The output

Algorithm 2: 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 $\mathbf{X}, f, E$

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).

Because the solution thus solved is closer to 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).

TABLE II: The values of parameters

Parameters	Values / Distribution
f_i	$[700, 1900]$
v_i	7.683
θ	-4558.52
λ	0.00125
T_s	1
r_{it}	$U[820, 10000]$
r'_{it}	$U[200, r_{it}]$
P_0	0.2
W	10 MHz
SNR	677

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 $T_s = 10$ seconds, because of their limited space.

The Cortex-A57 processor is ARM's highest performing processor, designed to further mobile and enterprise computing applications[4]. 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 (4), the number of instructions is $M_{it} = v_i \cdot f_{it} + \theta_i$. And the CPU capacity in Equation (5) is $C_{it} = (v_i \cdot f_{it} + \theta_i) \times T_s - r_{it}$,

We estimate the CPU capacity parameters v_i, θ_i is based on the analysis provided in [1], and the value is 7.683 and -4558.52. After calculation, the maximum and minimum M values are respectively $819, 10039$. 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[18] We have selected seven kinds of vehicle number to participate in resource sharing, which are$\{10, 15, 20, 25, 30, 35, 40\}$.

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:

$$P = 100 \cdot (1 - \frac{\sum_{t=1}^{T} \sum_{i} E_{it}^{blnc}}{E_{loc}}) \tag{23}$$

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.

$$FC = \frac{\sqrt{\frac{1}{N} \sum_{i=1}^{N} \left(E_{it}^{blnc} - \bar{E}\right)^2}}{\bar{E}} \tag{24}$$

C. Experimental Results

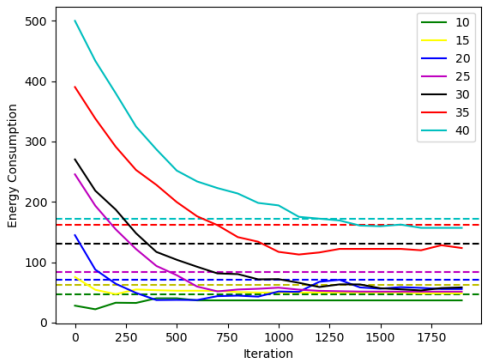


Fig. 2: Energy Consumption

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.

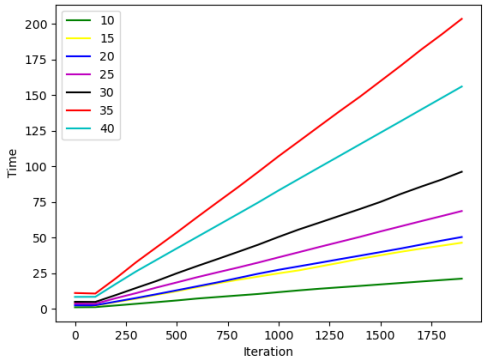
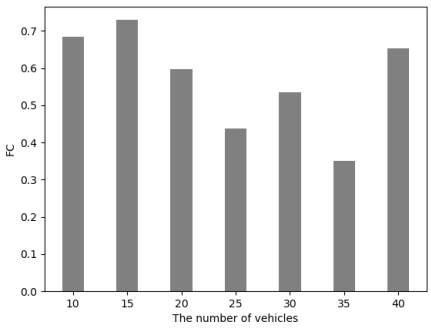


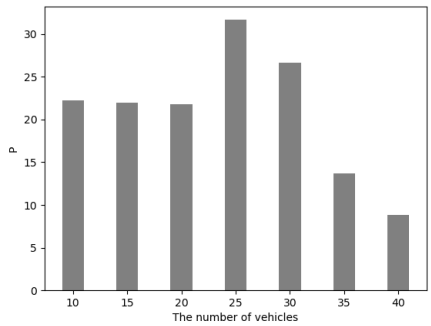
Fig. 3: Experimental running time

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 increase of vehicles, the balance of vehicle energy consumption increases first and then decreases. As Fig.4(b) shows, when the number of vehicles is 10 and 20, our system can save half of the energy consumption, but when the number of vehicles is 25, only 25% 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] 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.

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

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

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

[5] 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.

[6] 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.

[7] 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.

[8] Kyong Hoon Kim, Rajkumar Buyya, and Jong Kim. "Power Aware Scheduling of Bag-of-Tasks Applications with Deadline Constraints on DVS-enabled Clusters". In: *Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid '07)*. 2007, pp. 541–548. DOI: 10.1109/CCGRID.2007.85.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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).

[19] 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.

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