Received: 2 March 2021 | *Revised: 26 March 2021* | *Accepted: 28 March 2021*
DOI: 10.37917/ijeee.17.1.8 *Vol. 17 | Issue 1 | June 2021*

Open Access

Iraqi Journal for Electrical and Electronic Engineering
Original Article

IJ EEE University of Basrh College of Engineering

A Light Weight Multi-Objective Task Offloading Optimization for Vehicular Fog Computing

Sura Khairy Abdullah*, Adnan Jumaa Jabir
Department of Computer Science, College of Science, University of Baghdad, Baghdad, Iraq

Correspondence
*Sura Khairy Abdullah
Department of Computer Science, College of Science,
University of Baghdad, Baghdad, Iraq
Email: sura.kh.abdullah@gmail.com

Abstract
Most Internet of Vehicles (IoV) applications are delay-sensitive and require resources for data storage and tasks processing, which is very difficult to afford by vehicles. Such tasks are often offloaded to more powerful entities, like cloud and fog servers. Fog computing is decentralized infrastructure located between data source and cloud, supplies several benefits that make it a non-frivolous extension of the cloud. The high volume data which is generated by vehicles' sensors and also the limited computation capabilities of vehicles have imposed several challenges on VANETs systems. Therefore, VANETs is integrated with fog computing to form a paradigm namely Vehicular Fog Computing (VFC) which provide low-latency services to mobile vehicles. Several studies have tackled the task offloading problem in the VFC field. However, recent studies have not carefully addressed the transmission path to the destination node and did not consider the energy consumption of vehicles.
This paper aims to optimize the task offloading process in the VFC system in terms of latency and energy objectives under deadline constraint by adopting a Multi-Objective Evolutionary Algorithm (MOEA). Road Side Units (RSUs) x-Vehicles Mutli-Objective Computation offloading method (RxV-MOC) is proposed, where an elite of vehicles are utilized as fog nodes for tasks execution and all vehicles in the system are utilized for tasks transmission. The well-known Dijkstra's algorithm is adopted to find the minimum path between each two nodes. The simulation results show that the RxV-MOC has reduced significantly the energy consumption and latency for the VFC system in comparison with First-Fit algorithm, Best-Fit algorithm, and the MOC method.

KEYWORDS: Cloud Computing, Fog Computing, Internet of Vehicle, Intelligent Transportation System, Multi-Objective Evolutionary Algorithm.

I. INTRODUCTION

The current era is witnessing an increase in the number of intelligent objects that communicate with each other through the Internet of things (IoT), to provide many various services in various life aspects [1]. When these objects are smart cars and vehicles, then IoT will be called the Internet of vehicles (IoV) [2]. IoV is the fundamental platform of the intelligent transportation system (ITS), data are collected from the environment, stored, and processed through IoV [3]. Recently, the number of smart vehicles and IoV applications in ITSs has increased. This increase has led to an increase in the volume of data produced by the sensors of smart vehicles, where traditional databases cannot process this massive amount of data. The cloud computing is an operating model based on the information and communications technology that can be employed for vehicular network applications. Although cloud computing has massive resources, in which cloud servers can effectively address any task, it does not suit all IoV applications especially those that are delay-sensitive [4]. Due to two reasons; first, the traffic congestion caused by processing large amounts of data, second, the geographical distance between cloud servers and the vehicular network could cause a response-delay [5]. These issues have imposed challenges on utilizing the far servers of cloud computing environment for processing and storing such huge data. So, rather than moving data to the cloud, it may be more practical to process tasks near IoT devices or smart vehicles [6].

To conquer these drawbacks, a fog computing paradigm has emerged to provide computation and storage facilities close to data sources and users to reduce network congestion and the response time. Fog computing is also known as a cloud at the edge, fogging, or edge computing. The fog computing infrastructure is decentralized and located between the data source and the cloud [7]. This makes it suitable for handling delay-sensitive applications tasks. Fog computing supplies several benefits that make it a non-frivolous extension of the cloud, such as, supporting low-latency applications and real-time interactions, location awareness, and mobility support [8].

Recently, researchers interested in studying the employment of a large number of parked and slow-moving vehicles, especially in urban areas to improve system performance and reduce response time, by applying the vehicular fog computing (VFC) architecture [9]. VFC is a promising model aims to reduce response time to a minimum by exploiting limited resources of vehicles that are parked or moving at slow or medium speed [10]. Vehicles in the VFC system provide its resources such as computing and storage capacity to serve its neighbors [11]. In other words, vehicles within VFC system operate in a collaborative way.

According to the task specifications, the simple tasks can be executed by vehicles while the delay-sensitive ones are offloaded to the high capabilities Road Side Units (RSUs) servers. In addition, when a task requires intensive computation and cannot be executed by fog nodes within a deadline time, it is better to be offloaded to the unlimited capabilities cloud servers [12]. Although the use of high-specification servers in the cloud and fog improves the user experience and reduces latency, at the same time, this increases the amount of energy consumption.

The task offloading in the VFC environment has been considered as an NP-hard optimization problem, where determining the best task offloading decision requires an efficient algorithm that can deal with such high complexity and a large size problem. In recent literature, there have been several studies focused on utilizing heuristic and meta-heuristic algorithm to find the best task offloading solution by making the best tradeoff between different conflicting optimization objectives like latency and energy consumption under deadline and budget constraints. However, these studies have mostly considered the task execution in the cloud and fog, not considered the energy consumption of vehicles. Neither did they carefully address the transmission path to the destination node. The selection of the place where a task is processed along with a proper path for task transmission has a very high impact on offloading performance.

In this paper, a multi-objective evolutionary algorithm (MOEA/D) for task offloading optimization of VFC system is adopted to reduce both energy consumption and latency by considering the task transmission time and energy under deadline constraint. The vehicles' abilities for task execution and transmission are exploited, so the well-known Dijkstra's algorithm is adopted to find the shortest route for task transmission over vehicles. The major contributions of this paper are summarized below:

- A three layers architecture is proposed, consisting of the vehicular layer, the RSUs layer which contains several RSUs distributed along the road, and the top layer where the Macro Base station (MBS) that has high capabilities and located in the center of the road where this station works to find the optimum solutions for offloading [13].
- The RxV-MOC model is proposed. In this model, RSUs and elite vehicles are used for task execution. The path to transfer the task is chosen according to the place where the task is executed. This means, if the destination is an RSU, this task is transmitted over RSUs, while if the destination is a vehicle, this task is transmitted over vehicles.
- The MOEA/D algorithm is adopted to achieve task offloading optimization in fog computing by minimizing energy consumption and latency for the VFC system by utilizing both RSUs and vehicles for task computation and transmission.
- In order to find the best path for task transmission in the vehicles layer taking into consideration both energy consumption and latency, the well-known Dijkstra algorithm is adopted.
- Evaluation the performance of the proposed method by comparing to multi-objective computation offloading (MOC) method, First-Fit algorithm, and Best-Fit algorithm.

The rest parts of this paper are as follows: Section II states the literature works related to task offloading optimization. Section III describes the system architecture, exhibits the proposed task offloading method. Section IV described the problem formulation details and MOEA/D algorithm. Section V discusses the results obtained by the simulation; the paper is concluded in section VI.

II. Related Works

Task offloading is considered as one of the most critical issues in the VFC system, due to its significance in making decisions concerning where to process the vehicle tasks and how to allocate the resources for computation. Kumar et al. [14] presented a survey of literature related to task offloading. They discussed different types of algorithms used to distribute and offload programs to save energy or improve performance and described why computation offloading is important for limited resources devices. Zhu et al. [15] discussed the importance of determining whether a task offloading is useful or not and introduced a fog computing model and an offloading policy.

Chang et al. [16] investigated the problem of energy-efficient optimization. They worked to optimize both the offloading process and transmission power for the mobile devices in a fog computing system for the sake of decreasing the energy consumption with a delay constraint. Ning et al. [17] constructed an energy-efficient scheduling framework for balancing the offloading among RSUs. They focused on reducing the total energy consumption of RSUs with delay constraints, but they ignored the energy consumption of vehicles. A model of fog computing and an offloading policy was proposed by Zhu et al. [15]. This proposed offloading policy considered the completion time and energy consumption with the constraint of the charges of execution data in the cloud. However, only the energy consumption of task offloading and feedback result was considered, while the energy consumption during task transmission and execution was neglected The idea of utilizing parked vehicles was proposed by Liu et al. [18] to improve the performance of VANET. Then, a VFC paradigm which is based on utilizing parked and moving vehicles as fog infrastructures was proposed in [9] to improve the capability of communication and computation. Similarly, Wang et al. [19] designed an offloading algorithm for average response time reduction in fog-based IoV systems, where they used moving and parked vehicles as fog nodes. However, these researchers did not address the energy aspects.

Recently, Xu et al. [20] proposed a multi-objective computation offloading method (MOC) to minimize the energy consumption and the task execution time with the constraints of the load balancing and the ensuring of IoV data's trustworthiness. This model utilized vehicles for task transmission only and all the computation is performed by RSUs. Although the proposed method achieved improvement in energy consumption, the downsides of MOC are threefold; First, it did not involve the vehicles in the task processing. Second, the shortest path was not clearly addressed. Third, transmission energy was not considered when computing the total energy consumption. Although the current studies achieved acceptable performance in the field of task offloading optimization, the best transmission path, which ensures low time and energy consumption, still requires further investigation. This paper targets reducing the energy consumption and latency to complete the generated tasks by offloading and balancing the tasks among RSUs and vehicles.

III. System Model

In this section, the three layers of VFC system are described.

1. MBS layer: the management layer which is exemplified by MBS in the center of the road. The coverage area of MBS is sufficiently broad to reach all vehicles. MBS hosts the proposed MOEA algorithm, also hosts two data bases The first database contains information about RSUs (Id, coverage area, computational capacity, ready time, channel state, etc.). The second database provides vehicle details (Id, location, velocity, computational capability, wireless communication range, etc.). These data bases are updated periodically, so that MBS is fully aware of the system status.
2. RSUs layer: consists of a number of RSUs distributed along the unidirectional road. The set of RSUs is denoted as $\mathcal{R} = \{RSU_1, \dots, RSU_n, \dots, RSU_N\}$, and the set of indices is denoted as $\mathcal{N} = \{1, \dots, n, \dots, N\}$. The road is divided into N segments with the same size according to the coverage radius of RSUs. A vehicle can communicate wirelessly with RSU_n only when it is located in segment n.
3. Vehicular network layer: consist of a number of vehicles travelling in the same direction. The set of vehicles is denoted as $\mathcal{V} = \{v_1, \dots, v_m, \dots, v_M\}$ and the set of indices is denoted as $\mathcal{M} = \{1, \dots, m, \dots, M\}$. Each vehicle will generate one task. The set of tasks is denoted as $\mathcal{T} = \{t_1, \dots, t_m, \dots, t_M\}$. A task t_m is characterized by two features, the data size which is denoted by d_m and maximum tolerable delay which is denoted by t_m. Fig. 1 shows the system architecture, Fig. 1 shows the system architecture, where fog computing is represented by the management layer and RSUs layer.

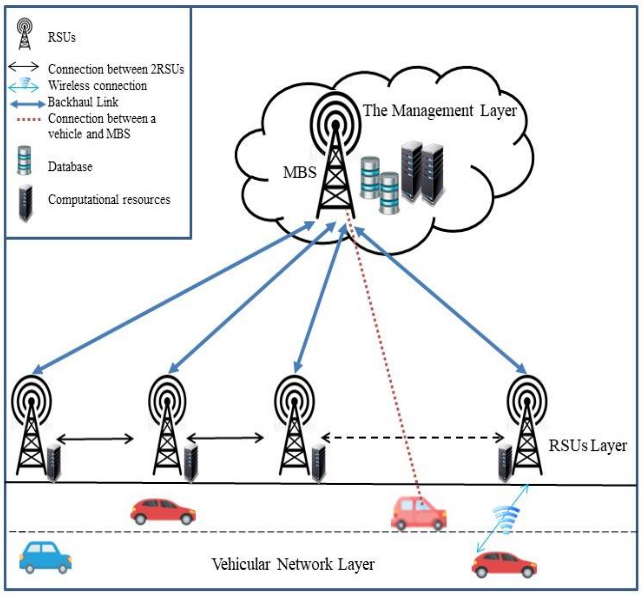

Fig. 1: System Architecture.

The MBS selects an elite of vehicles which is located in the middle of the road and along it. Employs this elite as a fog nodes and exploitation their resources for task computation. The set of elite of vehicles is denoted as $\mathcal{E} = \{ev_1, \dots, ev_x, \dots, ev_X\}$ and the set of indices is denoted as $\chi = \{1, \dots, x, \dots, X\}$. The set of fog nodes which is used for tasks computation is denoted as $\mathcal{FN} = \{fn_1, \dots, fn_f, \dots, fn_F\}$ where $F = N + E$ and is denoted as $\mathcal{F} = \{1, \dots, f, \dots, F\}$. In this work, when a vehicle v_m generates a task t_m, the latter asks the MBS for the best execution node and the best transmission path to the destination node by sending the computational requirement of the task (d_m and t_m). Then, the MBS assigns the task appropriate fog node fn_f so the task is executed with minimum energy consumption and latency. If fn_f is a vehicle, t_m will be transferred to fn_f over the vehicles network. If fn_f is an RSU, then t_m will be transferred over RSUs. Afterwards, the MBS will inform the vehicle v_m with the decision of task offloading by sending the ID of fn_f and the path for delivering the task to fn_f, as well as the path for delivering the result to concerned vehicle v_n. This work aim to reduce latency and the energy consumption for both RSUs and vehicles by offloading and balancing the tasks among $\mathcal{R}$ and $\mathcal{V}$. Table 1 summaries the mathematical variables used in this work.

TABLE 1.
Important Notations.

Notation	Variable
Individual	Ɲ
Initial population	Þ
Crossover Probability	Ç
Mutation Probability	ɱ
Probability of choosing the fog node for processing	Ŕ
Individual length	IL
Population size	Z
Number of RSU	$\mathcal{N}$
Number of vehicles, Number of tasks	$\mathcal{M}, \mathcal{T}$
Number of vehicles selected for tasks execution	χ
Bandwidth of an RSU	B^{RSU}
Bandwidth of an vehicle	B^{V}
Computational capacity of RSUs	C_n^{RSU}
Computational capacity of vehicles	C_{goal}^{V}
Data size of a task	d_m
Data size of a result	d'_m
Execution energy of RSUs	EE_n^{RSU}
Execution energy of vehicles	EE_m^{V}
Maximum tolerable delay of a task	t_m
Transmission energy of RSUs	TE^{RSU}
Transmission energy of vehicles	TE^{V}
Velocity of vehicles	$V_m^{velocity}$
Wireless communication of RSUs	W^{RSU}
Wireless communication of vehicles	W^{V}

IV. Task Offloading Problem Formulation and MOEA/D Algorithm

Generally speaking, real-world problems require satisfying multiple objectives at the same time. However, the optimization of one objective predominantly degrades at least another objective. The introduced problem of task offloading optimization in this thesis combines two antithetic objective functions to provide the VFC system with a set of solutions. Each of these solutions can perfectly map the tasks to the proper vehicles and RSU nodes, such that the required objectives are satisfied. This section presents how the MOEA/D algorithm is adopted for the task offloading problem.

The MOEA/D algorithm is utilized for the task offloading problem, such that, each individual in the meta-algorithm adopted in this study is represented as a vector with a length equals to the number of generated tasks. For the sake of simplicity, it has been assumed that each vehicle generates one task at a time, thus the total number of generated tasks equals the total number of vehicles and they are used interchangeably in the coming sections. Each gene represents a task generated by a vehicle, while its content identifies where this task will be executed.

A. The Initial Population

In this sub-section, the initial population Þ is described where the formation of the initial population can be derived as:

$$Þ_{p,g} = \text{rand}_{[1,F]} \quad (1)$$

where $p \in \{1,2,\dots,Z\}$, $g \in \{1,2,\dots,IL\}$, and Þ $=\{Ɲ^{1}, Ɲ^{2},\dots, Ɲ^{Z}\}$

$Ɲ^{z}$ represents the zth chromosome in the initial population. Algorithm 1 illustrates generation steps of the initial population for the proposed method.

Algorithm1 Generation of Initial Population

```
Input: $\mathcal{R}$, $\mathcal{V}$, Z, IL, Ŕ
Output: Initial population Þ
Begin
Fog-Node = $\emptyset$;
for P ← 1 to Z do
    for $m$ ← 1 to IL do
        generate a random value $r$;
        if $r \leq$ Ŕ then
            Fog-Node = an Id selected from $\mathcal{R}$ randomly;
        else
            Fog-Node = an Id selected from $\mathcal{E}$ randomly;
        endif
        $Þ_{P,n}$ = Fog-Node;
    endfor
endfor
Return Þ
END
```

B. *Fitness Evaluation*

In general, the meta-heuristic algorithms satisfy single or multi-objective requirements based on the case at hand. As previously indicated, the target of the proposed method is to minimize both the total energy and latency. Algorithm 2 illustrates the calculation of both latency and energy in fitness evaluation for the proposed method.

The objective function gauges the quality of each individual as follows:

1) Latency Measurement

The latency can be defined as the total amount of time required for the task completion which constitutes task transmission, task waiting time in the queue, task execution, and task result
feeding back. Before deriving these terms, it is necessary to define the following identifiers:

- *vsrc*: is the source vehicle that generates the task.
- *vdst*: is the vehicle where the task is executed.
- *rdst*: is the RSU where the task is executed.
- *snrst*: is the nearest RSU to *vsrc*.

Using these identifiers, the main terms of the latency objective are derived below:

The offloading time ($Toffl$) is the time required to offload the task t_m from the source vehicle (*vsrc*) to the nearest RSU (*snrst*), and can be expressed as:

$$Toffl^{tm} = \frac{I_m}{B^{V2I}} \quad (2)$$

If the destination node (*rdst* or *vdst*) is far from *vsrc*, the task should be transmitted, over RSUs or vehicles, according to the place where the task is executed. The transmission time ($Ttrans$) can be generally expressed as:

$$Ttrans^{tm}_{src2dst} = \sum_{1}^{H} \frac{I_m}{BW} \quad (3)$$

where *src* and *dst* are either RSUs or vehicles, H represents the total hop count between *src* and *dst* nodes, and BW is B^{RSU} or B^{V}.

The required time for the task completion encompasses the waiting time ($Twait$), which is the time that a task must wait until the resources get ready, plus the time consumption for the task execution, which are expressed as:

$$Texec^{tm}_{dst} = \frac{I_m}{C_{dst}} \quad (4)$$

$$Twait^{tm}_{dst} = \sum_{i=1}^{td} Texec^{ti}_{dst} \quad (5)$$

where $\boldsymbol{C_{dst}}$ represents the computation capability of the destination node which is either an RSU ($\boldsymbol{C^{RSU}_{rdst}}$) or a vehicle ($\boldsymbol{C^{V}_{vdst}}$), and $\boldsymbol{td}$ represents the number of tasks in the queue of the destination node that are waiting for their turn to be executed.

The task result $\boldsymbol{t'_n}$ should be transmitted to *vsrc* over RSUs or vehicles. Before that, the new location of vehicle after $\boldsymbol{t_n}$ is finished should be found to determine the transmission path of $\boldsymbol{t'_n}$ which may be the same path of $\boldsymbol{t_n}$ transmitting or may change. The vehicle location is determined depending on the vehicle location when it offloaded the task, the task transmission time, the task waiting time, the task execution time, the task result feeding back time, and the vehicle velocity.

If t_n is executed by an RSU, t'_n should be transmitted to the nearest RSU for the new location of *vsrc*. The downloading time to the *vsrc* can be expressed as shown in Eq. 6. If t_n is executed by a vehicle, t'_n is transmitted normally to *vsrc* without any need for downloading process.

$$Tdown^{t'm} = \frac{I'_m}{B^{I2V}} \quad (6)$$

Accordingly, the total time for the proposed model can be obtained as follow:

In this model, if the task t_n is executed by an RSU, the total time consists of offloading t_n to *snrst*, transmitting t_n to *rdst*, waiting for resources to get ready, executing t_n, transmitting t'_n to *snrst*, downloading t'_n to *vsrc*. This can be expressed as:

$$Ttotal^{tm}_{rdst} = Toffl^{tm} + Ttrans^{tm}_{snrst2rdst} + Twait^{tm}_{rdst} + Texec^{tm}_{rdst} + Ttrans^{t'm}_{rdst2snrst} + Tdown^{t'm}. \quad (7)$$

While if t_n is executed by a vehicle the total time consists of transmitting t_n to *vdst*, waiting for resources to get ready, executing t_n, and transmitting t'_n back to *vsrc* as follow:

$$Ttotal^{tm}_{vdst} = Ttrans^{tm}_{vsrc2vdst} + Twait^{tm}_{vdst} + Texec^{tm}_{vdst}\ Ttrans^{t'm}_{vdst2snrst} \quad (8)$$

To get the accurate minimum latency for one task, the total time consumption for each fog node must be calculated whether it was an RSU or a vehicle. the following objectives should be satisfied:

$$min \sum_{i=1}^{M} \sum_{j=1}^{F} Ttotal^{ti}_{j}, \quad (9)$$

$$s.t. \begin{cases} Eq.(7), Eq.(8), \\ Ttotal^{ti}_{j} \leq t_i. \end{cases} \quad (10)$$

2) *Energy Measurement*

The total energy consumption is the total amount of energy that is consumed for a task completion, which constitutes transmission energy, execution energy and result feedback energy. These terms can be generally expressed as follows:

The offloading energy ($Eoffl$) is the energy consumption for offloading t_n to the nearest RSU, and can be expressed as:

$$Eoffl^{tm} = E^{offload} \times Toffl^{tm}. \quad (11)$$

The transmission energy ($Etrans$) is the energy consumption for transmitting t_n to the destination node over RSUs and vehicles, can be expressed as:

$$Etrans^{tm}_{src2dst} = TE \times Ttrans^{tm}_{src2dst} \quad (12)$$

where *src* and *dst* are the source and destination nodes which can be either an RSU or a vehicle, and TE is the energy consumption of either RSUs (TE^{RSU}) or vehicles (TE^{V}) for transmission.

The execution energy ($Eexec$) is the energy consumption for executing t_n by RSUs or vehicles, can be expressed as:

$$Eexec^{tm}_{dst} = EE \times Texec^{tm}_{dst} \quad (13)$$

where dst is the node where the task gets execution, and EE is the energy consumption of either RSU (EE^{RSU}_{rdst}) or vehicle (EE^{V}_{vdst}) for processing.

The downloading energy can be expressed as shown in Eq. 12. If t_n is executed by a vehicle, t'_n is transmitted to *vsrc*.

$$Edown^{t'm} = E^{download} \times Tdown^{t'm} \quad (14)$$

The total energy consumption for the proposed models can be obtained as follow:

In this model, the total amount of energy consumption for $\boldsymbol{t_n}$ execution and transmission by RSUs is calculated as:

$$Etotal^{tm}_{rdst} = offl^{tm} + Etrans^{tm}_{snrst2rdst} + Eexec^{tm}_{rdst} + Etrans^{t'm}_{rdst2snrst} + Edown^{t'm}. \quad (15)$$

While, the total amount of energy consumption for $\boldsymbol{t_n}$ execution and transmission by vehicle is calculated as:

$$Etotal^{tm}_{vdst} = Etrans^{tm}_{vsrc2vdst} + Eexec^{tm}_{vdst} + Etrans^{t'm}_{vdst2snrst}. \quad (16)$$

Algorithm 2 The Fitness Evaluation

```
Input: $Ɲ^z$, $IL$, $\mathcal{R}$, V, $\mathcal{T}$
Output: Latency and Energy of $Ɲ^z$
Begin
 $m = \emptyset$, g = $\emptyset$, $Latency = 0$, $Energy = 0$;
 for $n \leftarrow 1$ to IL do
   $m$ = the content of gene numbered $n$;
   if $m$ is an RSU's Id then
     calculate $Ttotal_m^n$ and $Etotal_m^n$ based on Eq. 7 and Eq. 15;
     $Latency = Latency + Ttotal_m^n$;
     $Energy = Energy + Etotal_m^n$;
   else
     calculate $Ttotal_m^n$ and $Etotal_m^n$ based on Eq. 8 and Eq. 16;
     $Latency = Latency + Ttotal_m^n$;
     $Energy = Energy + Etotal_m^n$;
   endif
 endfor
 Return $Latency$ and $Energy$ of $Ɲ^z$
End
```

To obtain the minimum energy consumption for one task execution. For each fog node (an RSU or a vehicle), the total energy consumption must be calculated, the following objectives should be satisfied:

$$min \sum_{i=1}^{M} \sum_{j=1}^{F} Etotal_j^{t_i}, \tag{17}$$

$$s.t. \begin{cases} Eq.(15), Eq.(16), \\ Ttotal_j^{t_i} \leq t_i. \end{cases} \tag{18}$$

C. Crossover Operation

This operation is responsible for forming a new individual (Child) by integrating the genetic information of two elected solutions.

$$Child_i = \begin{cases} Ind_i^1 & rand_i \leq Ç \\ Ind_i^2 & otherwise \end{cases} \tag{19}$$

where Ç is the crossover probability. Algorithm.3 illustrate the crossover operation.

Algorithm 3 The Crossover Operation

```
Input: $Ɲ^1$, $Ɲ^2$, IL, Ç
Output: new individual ($Child$)
Begin
 $Child = \emptyset$;
 for $n \leftarrow 1$ to IL do
   generate a random value Y;
   if Y ≤ Ç then
     $Child[n]$ = the content of gene t of $Ɲ^1[n]$;
   else
     $Child[n]$ = the content of gene t of $Ɲ^2[n]$;
   endif
   perform the fitness evaluation of $Child$ based on Algorithm 2;
 endfor
 Return $Child$
End
```

D. Mutation Operation

The fundamental function that aids in the exploration of the entire search space and forbidding the population from falling in a local optimal solution. In this operation, a random modification to one or more gene values is done in an attempt to generate a better solution from the original solution that resulted from the crossover operation. In all proposed models, the mutation operation depends on the value of the mutation probability ɱ. For each gene, a random value is generated first. After that, this generated value is compared with ɱ. Gene content will be replaced if and only if the random value is less than or equal to ɱ.

If the fog node which executes the task is an RSU, then the gene content is randomly replaced by another RSU. But if the fog node is a vehicle, then the gene content will be randomly replaced by another vehicle. After that, the generated solution is examined to confirm whether the gene replacement process had a positive effect and produced a better solution than the previous solution or not. If the resulting solution is best than the original solution and satisfies the deadline condition, the replacement will be approved. Otherwise, the original solution will be restored and the replacement process is canceled. The mutation operation is illustrated in Algorithm 4.

Algorithm 4 The Mutation Operation

```
Input: $IL$, $Ɲ^z$, ɱ
Output: Enhanced individual $Ɲ'^z$
Begin
 L = latency of $Ɲ^z$, E = energy of $Ɲ^z$, $Ɲ'^z = Ɲ^z$;
 for $n \leftarrow 1$ to $IL$ do
   generate a random value ŋ;
   if ŋ ≤ ɱ then
     $fn$ = the content of gene numbered $n$;
     if ($fn$ is an RSU) then
       $new\text{-}fn$= Id selected from $\mathcal{M}$ randomly, where $new\text{-}fn \neq fn$;
     else
       $new\text{-}fn$ = Id selected from $\mathcal{E}$ randomly, where $new\text{-}fn \neq fn$;
     endif
     $Ɲ'^z[n] = new\text{-}fn$;
     perform the fitness evaluation for $Ɲ'^z$ based on Algorithm 2.3;
     if (($Ttotal$ of $Ɲ'^z[n] > D_t$) &&( $Ttotal$ of $Ɲ'^z >$ $Ttotal$ of $Ɲ^z$...
        || $Etotal$ of $Ɲ'^z >$ $Etotal$ of $Ɲ^z$)) then
       $Ɲ'^z[n] = Ɲ^z[n]$
     endif
   endif
 endfor
 Return $Ɲ'^z$
End
```

V. Numerical Result

In this section, the performance of the proposed algorithm is evaluated. In addition to MOC method, two basic offloading methods First-Fit and Next-Fit are employed against our proposed algorithm for comparison analysis. Then, the numerical results are introduced. These models and MOC model are implemented in Matlab Ten systems for each model with different entities and task specifications are used, such that, every system is executed ten times. It is necessary to mention that the population size is 100, the system generations is set also to 100, Ç, ɱ, and Ŕ are set to 0.3, 0.3, and 0.5 respectively. The simulation parameters are summarized in Table 2. The comparative methods adopted are briefly described as follows:

- **Multi-objective computing offloading (MOC) method:** this method has been presented by Xu et al. in [20]. They proposed a method for computation offloading, where the tasks are executed using RSUs. While tasks transmission is done through vehicles network. Also, they utilized a simple method to find shortest path between two vehicles.
- **First Fit in VFC system (First-Fit):** the task is offloaded to the nearest RSU. In the case of the nearest RSU cannot meet the required resources of the task, this task is offloaded to nearest RSU for the current RSU. This process is repeating until all tasks are offloaded.
- **Best Fit in VFC system (Best-Fit):** for each task, finding the RSU which can execute this task with minimum execution time taking into consideration the waiting time. This process is repeating until all tasks are offloaded.

Fig. 2 shows that RxV-MOC method takes less time than other methods. This mean, utilizing resources of vehicles for the task execution and transmission contribute in minimizing latency. There are four parts of time consumption which are, transmission time, execution time, waiting time, and feedback time. Fig. 3 and 4 show that RxV-MOC method consumes less average transmission time and also less average feedback time than MOC method, First-Fit and Best-Fit algorithm. It is noticeable that First-Fit algorithm outperform the MOC method which was utilized vehicles for task transmission. This lead to a response-delay because the transmission capability of vehicles is less than that of RSUs. First-Fit algorithm outperform Best-Fit algorithm, because the latter focused on execution and waiting time and not considered transmission time.

TABLE 2.
Simulation parameters.

Variable	Value
Length of the road	3000 m
$\mathcal{N}$	6
$\mathcal{M}, \mathcal{T}$	20-100
χ	5
B^{RSU}	27 Mbps
C_n^{RSU}	1 – 4 GHz
$E^{download}$	31.7 dBm
TE^{RSU}	33 dBm
EE_n^{RSU}	43 – 49 dBm
W^{RSU}	250 m
B^{V}	20 Mbps
B^{V2I}	2 Mbps
C_n^{V}	0.5 – 1 GHz
$E^{offload}$	31.7 dBm
EE_m^{V}	36 – 40 dBm
TE^{V}	14 dBm
W^{V}	250 m
$V_m^{velocity}$	30 - 60 Km/h
t_m	10 Sec.
d_m	1 – 256 KB
d'_m	≥ 0.5 KB
B^{V2I}	2 Mbps

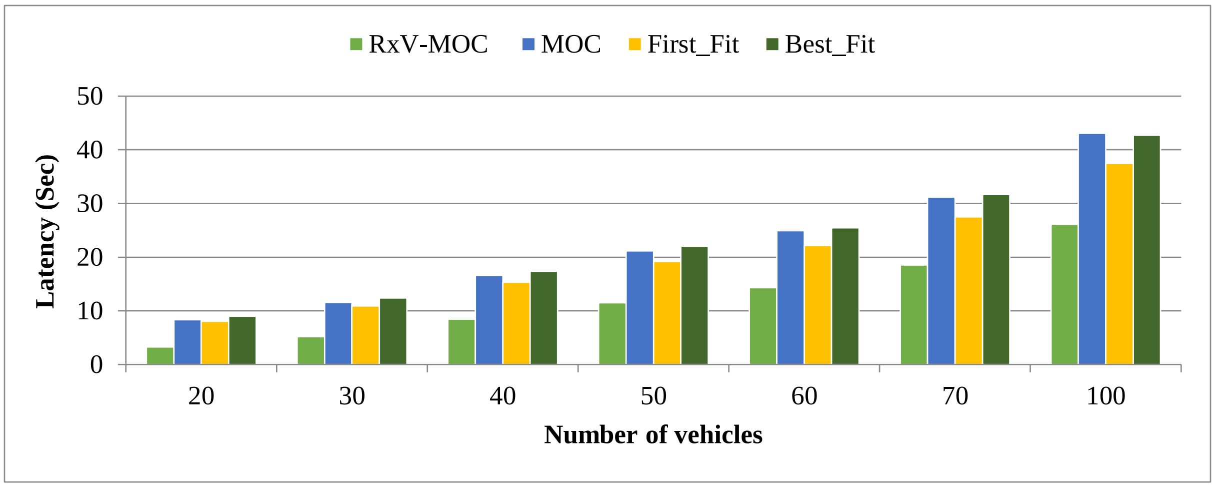

Fig. 2: Comparison of the average latency of RxV-MOC, MOC, First-Fit, and Best-Fit versus the number of vehicles.

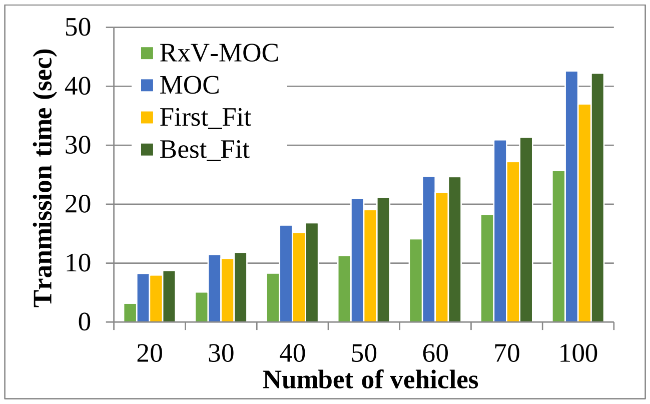

Fig 3: Time consumption of the task transmission.

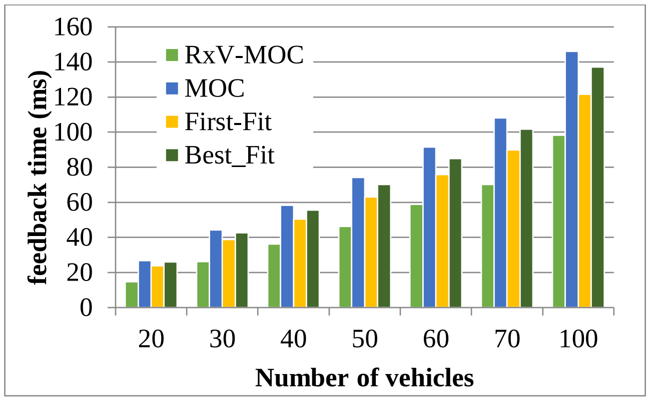

Fig. 4: Time consumption of the result feeding back.

Fig. 5 and 6 show that our proposed method consumed more time for execution and waiting than other methods. Due to the limited computational capacity of vehicles.

Fig. 7 shows the average energy consumption versus the number of vehicles for the proposed method, MOC method, First-Fit algorithm, and Best-Fit algorithm. It is noticeable that the RxV-MOC method consumes less energy than the others. As previously described, the total energy consumption consists of energy consumption of transmission, energy consumption of execution, and energy consumption of result feeding back.

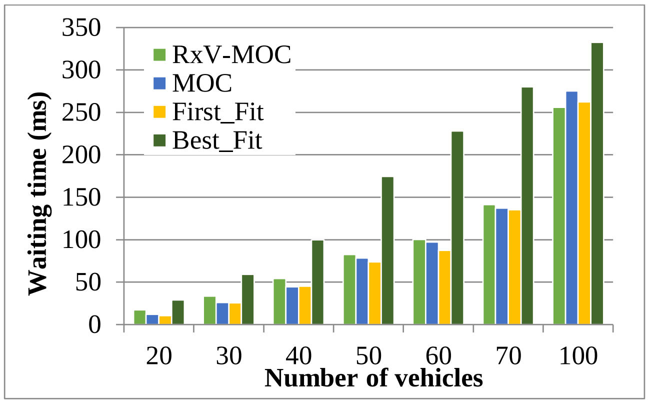

Fig. 5: The amount of waiting time.

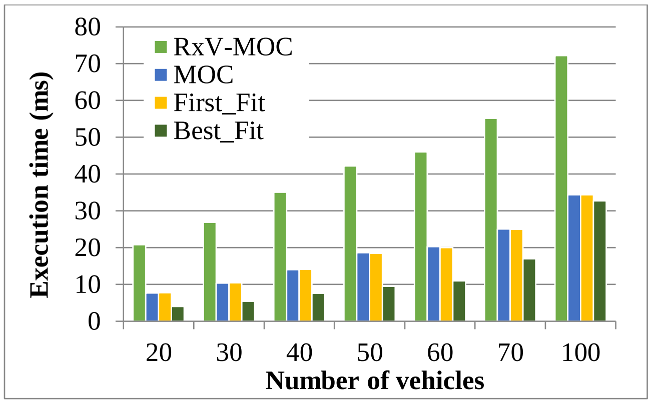

Fig. 6: Time consumption for the task execution.

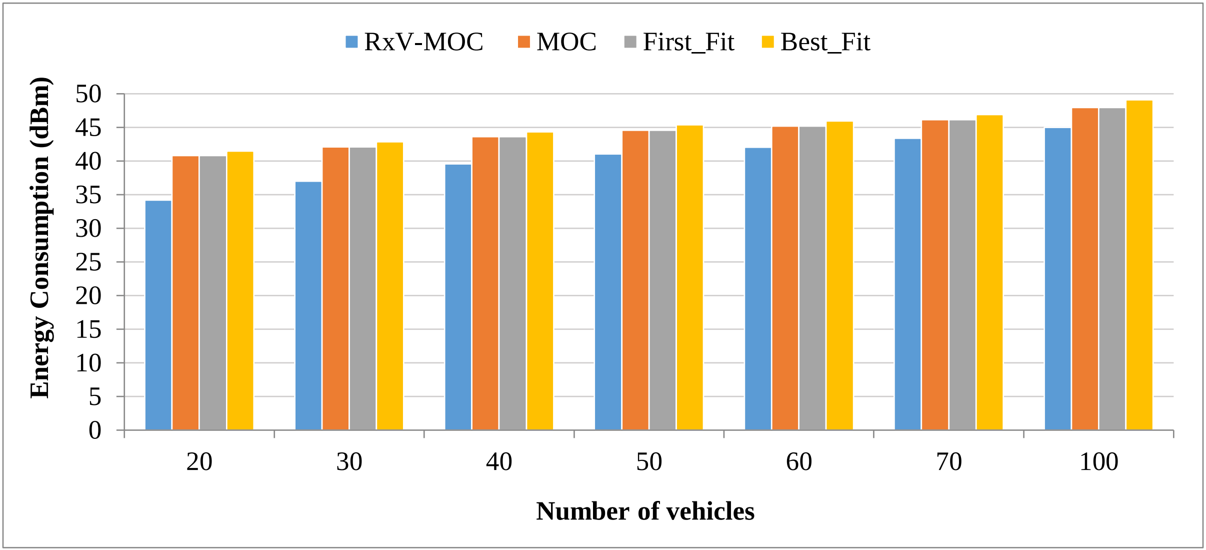

Fig. 7: Comparison of the average energy of RxV-MOC, MOC, First-Fit, and Best-Fit versus the number of vehicles.

Fig. 8, 9, and 10 show that the proposed method consumed less amount of energy for task transmission, result feeding back, and task execution respectively. Because of utilizing vehicles resources and application the theory of transport according to the place of execution, which contributed reducing the energy consumption. The results show that utilization vehicles in the VFC system for task transmission only does not have a positive effect in terms of reducing energy consumption. Our proposed method, MOC method, and First-Fit algorithm outperform Best-Fit algorithm, because the latter do not consider the energy consumption of task transmission and result feeding back.

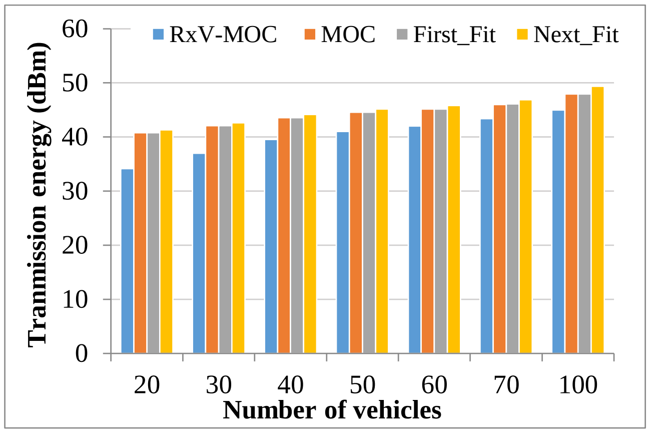

Fig. 8: Transmission energy consumption.

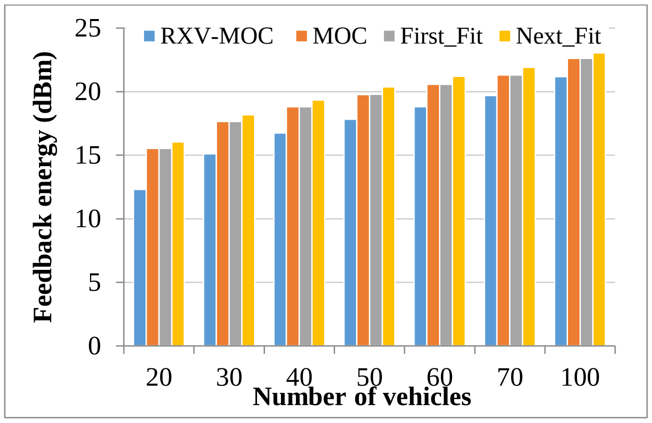

Fig. 9: Feeding back energy consumption.

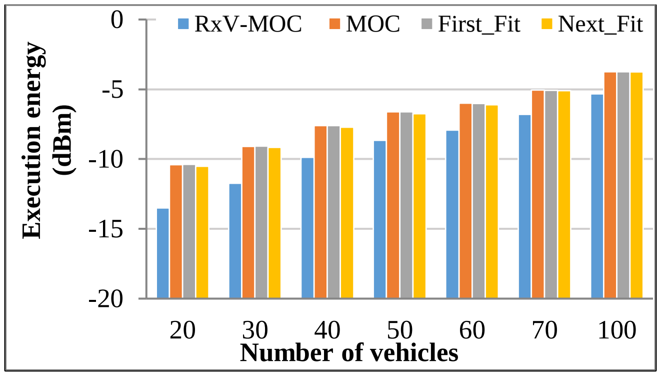

Fig. 10: Execution energy consumption.

Table 3 display a comparison between the proposed method (RxV-MOC) and the MOC method

TABLE 3.
A comparison between RxV-MOC and MOC.

	RxV-MOC	MOC
Reduce latency	✓	×
Reduce energy consumption	✓	×
Exploitation of the vehicle capability for transmission.	✓	✓
Exploitation of the vehicle capability for computation.	✓	×
Considering the transmission energy	✓	×

VI. Conclusions

In this study, the problem of task offloading in VFC is revisited. A RxV-MOC method is proposed in an attempt to optimize two contradictory objectives, namely latency and energy. The well-known multi-objective algorithm (MOEA/D) has been adopted for task offloading optimization. In addition, an elite of vehicles are utilized as fog nodes and made use of their ability to process and transfer tasks, Dijkstra's algorithm is adopted in order to find the minimum path between two vehicles. The results presented in previous section show that our proposed method successfully reduces latency on average of 48% and energy by 9%.

According to the obtained results, RxV-MOC method consumes more execution and waiting time than others. Enhancing the performance of the proposed method in terms of execution and waiting time is worth further investigations. A real dataset can be used to verify the performance of the proposed models and more criteria and constraints like cost and budget can be investigated.

Conflict of Interest

The authors have no conflict of relevant interest to this article.

References

[1] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, and O. Rana, "Fog computing for the internet of things: A Survey," *ACM Transactions on Internet Technology (TOIT)*, vol. 19, pp. 1-41, 2019.

[2] M. N. Sadiku, M. Tembely, and S. M. Musa, "Internet of vehicles: An introduction," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 8, p. 11, 2018.

[3] J. Kang, R. Yu, X. Huang, and Y. Zhang, "Privacy-preserved pseudonym scheme for fog computing supported internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, pp. 2627-2637, 2017.

[4] I. B. Lahmar and K. Boukadi, "Resource Allocation in Fog Computing: A Systematic Mapping Study," in *2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*, 2020, pp. 86-93.

[5] M. N. Abdulredha, A. A. Bara'a, and A. J. Jabir, "Heuristic and Meta-Heuristic Optimization Models for Task Scheduling in Cloud-Fog Systems: A Review," *Iraqi Journal for Electrical And Electronic Engineering*, vol. 16, 2020.

[6] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On reducing IoT service delay via fog offloading," *IEEE Internet of Things Journal*, vol. 5, pp. 998-1010, 2018.

[7] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang, *et al.*, "Fog Computing: Survey of trends, architectures, requirements, and research directions," *IEEE access*, vol. 6, pp. 47980-48009, 2018.

[8] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 2012, pp. 13-16.

[9] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Transactions on Vehicular Technology*, vol. 65, pp. 3860-3873, 2016.

[10] S.-s. Lee and S. Lee, "Resource Allocation for Vehicular Fog Computing using Reinforcement Learning Combined with Heuristic Information," *IEEE Internet of Things Journal*, 2020.

[11] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Computer Communication Review*, vol. 44, pp. 27-32, 2014.

[12] M. N. Abbas, A. A. Bara'a, and N. J. Kadhim, "Evolutionary Based Set Covers Algorithm with Local Refinement for Power Aware Wireless Sensor Networks Design," *Iraqi Journal of Science*, pp. 1959-1966, 2018.

[13] P. Liu, J. Li, and Z. Sun, "Matching-based task offloading for vehicular edge computing," *IEEE Access*, vol. 7, pp. 27628-27640, 2019.

[14] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile networks and Applications*, vol. 18, pp. 129-140, 2013.

[15] Q. Zhu, B. Si, F. Yang, and Y. Ma, "Task offloading decision in fog computing system," *China Communications*, vol. 14, pp. 59-68, 2017.

[16] Z. Chang, Z. Zhou, T. Ristaniemi, and Z. Niu, "Energy efficient optimization for computation offloading in fog computing system," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, 2017, pp. 1-6.

[17] Z. Ning, J. Huang, and X. Wang, "Vehicular fog computing: Enabling real-time traffic management for smart cities," *IEEE Wireless Communications*, vol. 26, pp. 87-93, 2019.

[18] N. Liu, M. Liu, W. Lou, G. Chen, and J. Cao, "PVA in VANETs: Stopped cars are not silent," in *2011 Proceedings IEEE INFOCOM*, 2011, pp. 431-435.

[19] X. Wang, Z. Ning, and L. Wang, "Offloading in Internet of vehicles: A fog-enabled real-time traffic management system," *IEEE Transactions on Industrial Informatics*, vol. 14, pp. 4568-4578, 2018.

[20] X. Xu, R. Gu, F. Dai, L. Qi, and S. Wan, "Multi-objective computation offloading for internet of vehicles in cloud-edge computing," *Wireless Networks*, pp. 1-19, 2019.