Computers, Materials & Continua
DOI:10.32604/cmc.2021.015026
Article

Tech Science Press

A Resource Management Algorithm for Virtual Machine Migration in Vehicular Cloud Computing

Sohan Kumar Pande[1], Sanjaya Kumar Panda[2], Satyabrata Das[1], Kshira Sagar Sahoo[3], Ashish Kr. Luhach[4], N. Z. Jhanjhi[5,*], Roobaea Alroobaea[6] and Sivakumar Sivanesan[5]

[1]Veer Surendra Sai University of Technology Burla, Burla, 768018, India
[2]National Institute of Technology Warangal, Warangal, 506004, India
[3]VNR Vignana Jyothi Institute of Engineering and Technology, Hyderabad, 500090, India
[4]The Papua New Guinea University of Technology, Lae-41, Papua, New Guinea
[5]School of Computer Science and Engineering, Taylor's University, Subang Jaya, 47500, Malaysia
[6]Department of Computer Science, College of Computers and Information Technology, Taif University, Taif, 21944, Saudi Arabia
[*]Corresponding Author: N. Z. Jhanjhi. Email: noorzaman.jhanjhi@taylors.edu.my
Received: 02 November 2020; Accepted: 19 December 2020

Abstract: In recent years, vehicular cloud computing (VCC) has gained vast attention for providing a variety of services by creating virtual machines (VMs). These VMs use the resources that are present in modern smart vehicles. Many studies reported that some of these VMs hosted on the vehicles are overloaded, whereas others are underloaded. As a circumstance, the energy consumption of overloaded vehicles is drastically increased. On the other hand, underloaded vehicles are also drawing considerable energy in the under-utilized situation. Therefore, minimizing the energy consumption of the VMs that are hosted by both overloaded and underloaded is a challenging issue in the VCC environment. The proper and efficient utilization of the vehicle's resources can reduce energy consumption significantly. One of the solutions is to improve the resource utilization of underloaded vehicles by migrating the over-utilized VMs of overloaded vehicles. On the other hand, a large number of VM migrations can lead to wastage of energy and time, which ultimately degrades the performance of the VMs. This paper addresses the issues mentioned above by introducing a resource management algorithm, called resource utilization-aware VM migration (RU-VMM) algorithm, to distribute the loads among the overloaded and underloaded vehicles, such that energy consumption is minimized. RU-VMM monitors the trend of resource utilization to select the source and destination vehicles within a predetermined threshold for the process of VM migration. It ensures that any vehicles' resource utilization should not exceed the threshold before or after the migration. RU-VMM also tries to avoid unnecessary VM migrations between the vehicles. RU-VMM is extensively simulated and tested using nine datasets. The results are carried out using three performance metrics, namely number of final source vehicles (*nfsv*), percentage of successful VM migrations (*psvmm*) and percentage of dropped VM migrations (*pdvmm*), and compared with threshold-based algorithm (i.e., threshold) and cumulative sum (CUSUM) algorithm. The comparisons show that the RU-VMM algorithm performs better than the existing algorithms. RU-VMM algorithm improves 16.91% than the CUSUM algorithm and 71.59% than the threshold algorithm in terms of *nfsv*, and 20.62% and 275.34% than the CUSUM and threshold algorithms in terms of *psvmm*.

Keywords: Resource management; virtual machine migration; vehicular cloud computing; resource utilization; source vehicle; destination vehicle

1 Introduction

In the era of technology, vehicular cloud computing (VCC), Internet of vehicles (IoV) and cloud of things (CoT) are some of the promising technologies to provide better services to the users [1–6]. Many modern smart vehicles are connected to the cloud in VCC to offer various services, such as information, storage, cooperation, computation, and infotainment as a service [7,8]. Each modern smart vehicle is equipped with an on-board unit (OBU). In general, OBU contains various components, such as storage unit, processor, sensors, global positioning system (GPS), cameras, communication systems and many more, and OBU is underutilized for a considerable amount of time [3,9]. Moreover, VCC provides a platform for these underutilized vehicles' resources, and these resources can be intelligently utilized by offering services to the users on a pay-per-use basis [10]. For instance, car drivers and passengers can benefit by obtaining services like traffic management, parking availability, emergency assistance, smooth navigation, multimedia content sharing, etc., from the VCC [1,11].

A significant number of vehicles is spending considerable amount of time at parking lot and moving within a given area (hereafter, referred as grid). These vehicles can be participated in VCC by connecting to nearest fixed roadside unit (RSU), which is further connected to cloud service provider (CSP) in order to provide backbone infrastructure to the cloud [1,10–12]. On the other hand, CSP can assign the loads to the RSUs in order to execute the same in the roadside infrastructure. Upon receiving loads from CSP, RSU can identify suitable vehicle and create a VM on that vehicle to perform load sharing as reported in [13]. However, the resources of the vehicle are not fully utilized as the created VM is handling all the resources of the vehicle. Many studies [13–22] reported that some of these VMs are overloaded and others are underutilized. Moreover, energy consumption of overloaded and under loaded vehicles are varying with respect to their utilization. Note that high utilization leads to drastic increase in energy consumption and low utilization or idle situation leads to wastage of energy. Alternatively, energy consumption increases exponentially on or above 70% of resource utilization [14,19]. Therefore, it is challenging task to manage overloaded/source and under loaded/destination vehicles and share their load to minimize the energy [15,18,23,24]. One of the solutions is to migrate/consolidate the over utilized VMs of overloaded vehicles to under loaded vehicles in order to improve resource utilization. This phenomenon motivated us to identify the overloaded vehicles and their corresponding target vehicles by incorporating threshold value and checking the suitability of the vehicles.

In this paper, we propose a resource management algorithm, called resource utilization-aware VM migration (RUVMM) that selects the appropriate source and destination vehicles, and suitable VMs hosted on the source vehicle for the process of VM migration. For this, it monitors the trend of resource utilization and selects the source and destination vehicles within a pre-determined threshold limit. It ensures that the resource utilization of any vehicles should not exceed the threshold before or after the migration process. In this way, it solves the challenges associated with VM migration in VCC. RU-VMM is simulated using MATLAB and tested using nine randomly generated datasets. The simulation results are carried out for RU-VMM algorithm and compared with two existing algorithms, namely threshold-based algorithm [14,18] and cumulative sum (CUSUM) [13] in terms of three performance metrics, namely number of source vehicles, percentage of successful VM migrations and percentage of dropped VM migrations. For the sake of comparison, we use threshold and CUSUM algorithms as other algorithms are not directly comparable to our algorithm. The comparison results show that RU-VMM algorithm performs better than other two algorithms in terms of above performance metrics. Our major contributions are as follows.

- We develop a resource management algorithm, RUVMM to migrate the over utilized VMs that are hosted on the overloaded vehicles to the under loaded vehicles.
- The overloaded vehicles are identified by determining a threshold value and checking the suitability of vehicles.
- The migration process determines the source vehicles and destination vehicles by avoiding unnecessary migrations and ensures that the resource utilization of any vehicles at any point of time should not exceed the threshold.
- We simulate the existing algorithms, threshold and CUSUM, and compare their results to show the efficacy of the proposed algorithm.

The rest of the paper is organized as follows. Section 2 discusses various studies carried out in the field of VM consolidation and migration. Section 3 presents the vehicular cloud model and problem statement followed by the proposed algorithm and performance metrics in Section 4 and Section 5, respectively. Simulation results are discussed in Section 6. We conclude in Section 7 with some notable remarks and future works.

2 Related Work

In the recent past, many researchers [7,9,25] have explored the advancement and application of VCC for better traffic management, entertainment services, assistance in emergencies to drivers, finding parking areas, smooth navigation, managing data centers, parking lot and many more. With an increase in popularity, VCC also faces various notable problems, such as resource management, energy consumption, bandwidth, latency and many more [17,23,26]. Researchers have explored different dimensions of VCC to address the above-mentioned problems, which are briefly discussed in this section.

Hamdi et al. [18] have stated that high energy consumption makes an adverse effect on cost and environment. They have highlighted VM consolidation techniques in various dimension and discussed some important characteristics of VM consolidation techniques like varying nature of resource requirement, ideal power consumption of physical machines etc. Wu et al. [23] have proposed a power-aware scheduling algorithm, which uses the utilization threshold strategy to select host physical machines and minimum utilization gap to select VMs for the process of VM migration. Here, they used power-aware best fit decreasing to select destination vehicles. Hsieh et al. [17] have considered both current and future utilization of resources, and predicted future utilization of resources more accurately. They have determined the host overload and under load detection to perform this prediction. Li et al. [26] have proposed an energy-efficient and quality-aware VM consolidation technique, which uses four algorithms, namely multi-resource host overload detection, quality of service-aware VM selection, discrete differential evolution-based VM placement and under-loaded host detection to improve resource utilization and reduced energy consumption. Mekala et al. [16] have focused on the resource requirement rate for task classification, resource balance ranking and processing element cost to evaluate overloaded host, under loaded host and VMs for migration. Sharma et al. [27] have proposed a failure prediction technique, which is based on exponential smoothing to trigger two fault tolerance mechanisms, namely VM migration and VM check pointing. Many other studies on power and resource-aware VM placement and VM migration techniques are discussed in [3,10,11,28,29].

Andreolini et al. [14] have presented a dynamic load management approach using cumulative sum (CUSUM) algorithm to take a decision on VM reallocation. For this, it determines the abrupt change in the CPU utilization without considering any fixed thresholds. If the change is persistent, then it selects the corresponding host as a sender host and finds a suitable guest host to initiate the migration process. This algorithm achieves better performance and low overhead as it minimizes the number of migrations. However, the number of source vehicles is increased to a greater extent due to unknown thresholds over time. Alternatively, they have considered the load behavior of resources without relying on average measures. Khanna et al. [15] have monitored the performance of resources. If the performance exceeds a threshold, then it initiates the process of migration. The threshold value is relying on the service level agreement. Hence, it may vary with respect to the resources. Hsu et al. [19] have considered 70% as a threshold to perform the task consolidation. If the load exceeds 70%, then it redirects the load to other resources. However, in the case of the unavailability of resources, it keeps the load by violating the threshold. Some other studies on threshold-based algorithms are discussed in [20,21]. Most of the above studies have not considered the trend behavior of resource utilization of the vehicles while selecting source and destination vehicles in the process of VM migration. In this paper, we propose an algorithm, RU-VMM, which focuses on the trend and change in resource utilization to select source and destination vehicles and continuously monitors the utilization before and after the process of migration, such that unnecessary migrations are avoided.

3 Vehicular Cloud

3.1 Vehicular Cloud Model

We consider a heterogeneous VCC environment, as shown in Fig. 1, where a large number of vehicles provide Intra and Inter grid transport facilities. This environment has a significant difference while comparing with traditional datacenters, and they are moving network pool, autonomous cloud formation and federation, mobility, and many more. In VCC, the vehicles are equipped with various resources, which are generally underutilized. These underutilized resources can host virtual machines (VMs) in order to help the cloud service provider (CSP) to render various services and improve their utilization. Whenever a vehicle enters in a grid, it broadcasts a beacon beam that consists of various information about that vehicle. The beacon beam is received by the roadside unit (RSU) of the respective grid. Moreover, vehicles are connected to the CSP through the RSU that are present in the grid. Here, CSP is a centralized unit to distribute the loads among the RSUs. RSUs keep track of the vehicles, and collect and monitor the current load, utilization of the VMs of the vehicles (i.e., resource utilization) along with the information like path, speed, parking time of the vehicles, etc. Note that RSU is a centralized component to monitor the vehicles that are present in a grid. Here, we assume that the resource utilization of vehicles is identifiable and collected periodically by the corresponding RSU. RSUs receive the loads from the CSP and distribute the same to the vehicles of the grid. Each vehicle can host one or more VMs to execute the assigned load and each VM contains its own operating system and required hardware resources using virtualization technology. To meet better resource utilization and load management, migration of overloaded VMs to under loaded VMs is required. However, the challenging task is the selection of the source vehicles, destination vehicle, and VMs in the process of migration. It is important to mention that VM migration is associated with various parameters, such as bandwidth, delay, and cost. Here, unnecessary migrations can significantly increase these parameters. Therefore, sufficient attention should be taken in the selection of source vehicles, destination vehicles, and VMs in order to overcome unnecessary migrations.

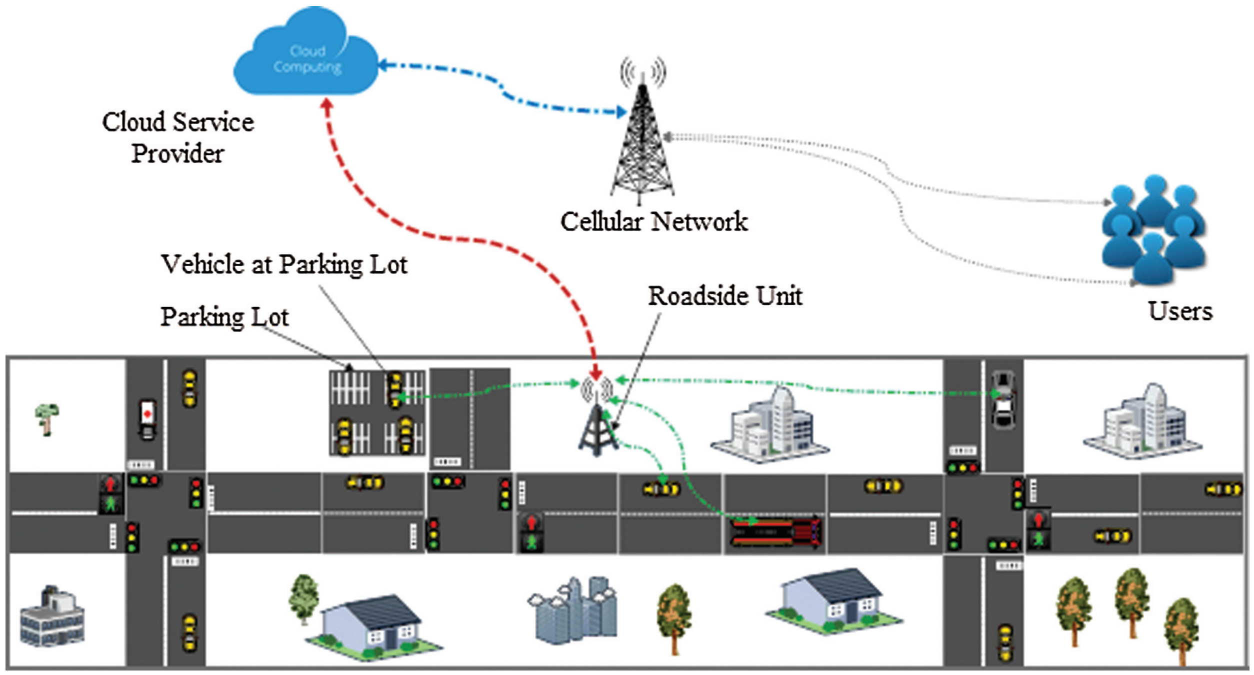

Figure 1: A system architecture

3.2 Problem Statement

Let us consider a grid G, in which a set of n vehicles, $V = \{V_i \mid i = 1, 2, 3, \ldots, n\}$ is presented and connected to the RSU. Each vehicle V_i, $1 \le i \le n$ can host M_i, $1 \le i \le n$ VMs. $VMU[i, j, k]$ is a 3-D matrix which represents the utilization of VM_j, $1 \le j \le M_i$ at time instance k, $1 \le k \le t$, that is hosted on vehicle V_i, $1 \le i \le n$ (Eq. (1)). Here t represents the time window in which utilization of the vehicles are periodically collected.

$$VMUijk = \begin{array}{c} \\ VMi1 \\ VMi2 \\ \vdots \\ VMiMi \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & \ldots & t \end{array} \\ \left\{ \begin{array}{cccc} VMUi11 & VMUi12 & \cdots & VMUi1t \\ VMUi21 & VMUi22 & \cdots & VMUi2t \\ \vdots & \vdots & \ldots & \vdots \\ VMUiMi1 & VMUiMi2 & \cdots & VMUiMit \end{array} \right\} \end{array} \tag{1}$$

It is noteworthy to mention that, the utilization of the vehicle V_i, $1 \leq i \leq n$ at time instance k, $1 \leq k \leq t$ is the summation of the utilization of the hosted VMs at that time. Here, the problem is to find the source vehicles, destination vehicles and VMs for the process of VM migration such that the number of final source vehicles and the percentage of dropped VM migration are minimized, and the percentage of successful VM migration is maximized. Alternatively, the main aspect of the problem is to reduce the number of final source vehicles in order to minimize the unnecessary number of VM migration(s).

4 Proposed Algorithm

The proposed resource utilization-aware VM migration (RU-VMM) algorithm is a load balancing algorithm. It is implemented in the centralized cloud in which RSUs of the grids act as intermediary between vehicles and cloud. The objective of this algorithm is to migrate the load from the VMs that are hosted on overloaded vehicles to the VMs that are hosted on under loaded vehicles. Alternatively, it balances the load of the VMs and improves the resource utilization of the VMs. This algorithm works in three phases as follows. In the first phase, it finds the set of probable source vehicles. In the second phase, it finds the VMs that are hosted on the source vehicles to initiate the process of migration. In the last phase, it finds the destination vehicles to migrate the VMs and performs the actual process of migration. The pseudo-code of these phases is shown in Algorithm 1 and Procedures 1–4. Tab. 1 shows the notations and their definition used in the pseudo-code of the proposed algorithm. An overview of the proposed algorithm is shown in Fig. 2.

Table 1: Notations used in the pseudo-code and their definition

Notation	Definition	Notation	Definition
VU_{ik}	Resource utilization of vehicle V_i at time instance k	$nfsv$	Number of final source vehicles
VM_{mig}	Set of VMs that are suitable for migration	V_{dest}	Set of destination vehicles
$npsv$	Number of probable source vehicles	V_{src}	Set of source vehicles
$SPSV$	Set of probable source vehicles	STD	Standard deviation function
$ncvmm$	Number if candidate VMs suitable for migration	$POWER$	Function to calculate exponentiation
$SORT$	Function to sort in ascending/descending order	AVG	Average
$npdv$	Number of probable destination vehicles	$AVGUTI$	Average utilization
$nvmm$	Number of successful VM migrations		

RU-VMM algorithm first calculates the utilization of the vehicles for a given time window as shown in Algorithm 1 (Line 1–5). Then it calls the three procedures for selecting probable source vehicles (Procedure 1), checking their suitability (Procedure 2, which is called from Procedure 1, Procedure 3 and Procedure 4), finding VMs (Procedure 3) and destination vehicles (Procedure 4), respectively.

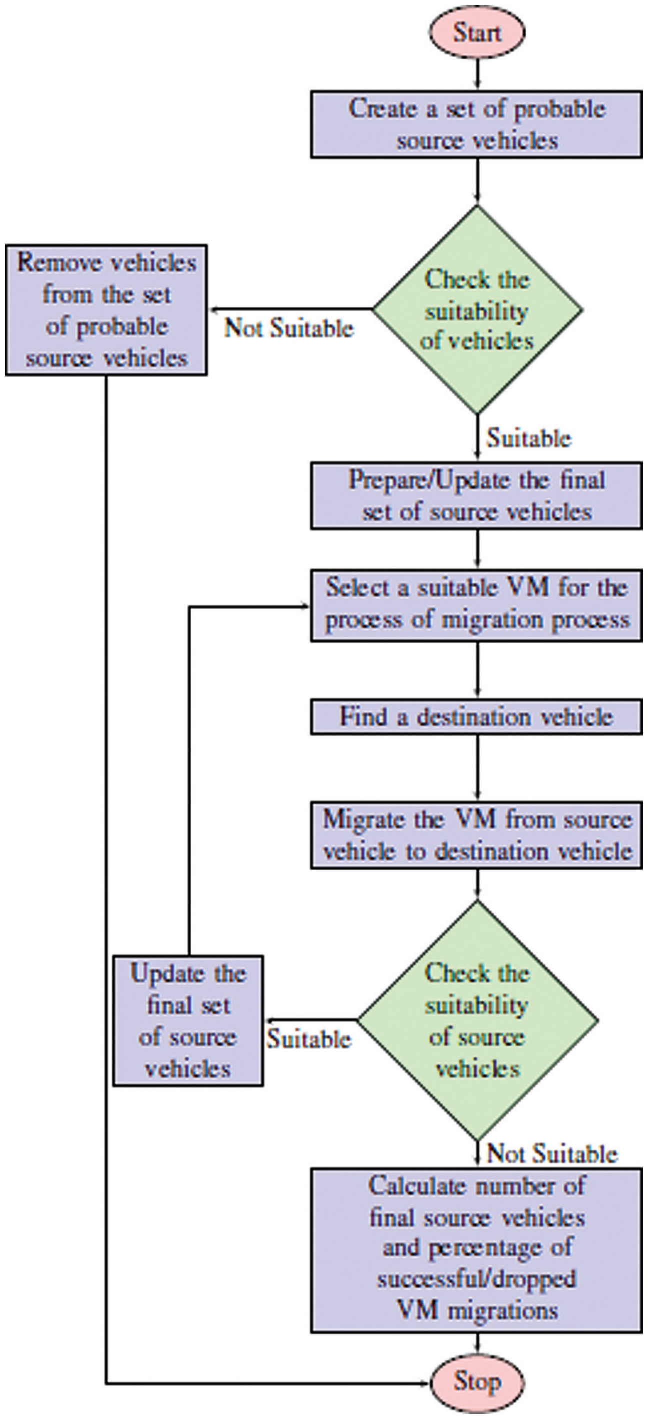
Start
Create a set of probable source vehicles
Check the suitability of vehicles
Not Suitable
Remove vehicles from the set of probable source vehicles
Suitable
Prepare/Update the final set of source vehicles
Select a suitable VM for the process of migration process
Find a destination vehicle
Migrate the VM from source vehicle to destination vehicle
Check the suitability of source vehicles
Suitable
Update the final set of source vehicles
Not Suitable
Calculate number of final source vehicles and percentage of successful/dropped VM migrations
Stop

Figure 2: An overview of the proposed algorithm

Algorithm 1: Pseudo code for RU-VMM

Input: A 3-D matrix VMU, $n =$ number of vehicles, $t =$ length of time window and $M[i] =$ number of VMs in vehicle i

Output: V_{src}, V_{dest}, VM_{mig}

```
1.  for $i = 1, 2, 3, \ldots, n$ do
2.    for $k = 1, 2, 3, \ldots, t$ do
3.        $VU[i, k] = \sum_{j=1}^{M[i]} VMU[i, j, k]$
4.    endfor
5.  endfor
6.  Call $FIND\text{-}SOURCE\text{-}VEHICLES(VU)$
7.  Call $FIND\text{-}VM\text{-}FOR\text{-}MIGRATION(VU, VMU, V_{src}, nfsv)$
8.  Call $FIND\text{-}DESTINATION\text{-}VEHICLES(VU)$
```

The Algorithm calls Procedure 1 to find a set of probable source vehicles. For this, it checks sum of the utilization of VMs that are hosted by vehicles within a time window t. Note that the sum of the utilization of the VMs should not exceed 100%. If the utilization value is above a threshold (say, 70%) at any point of time and it is persistent, then the vehicle is a potential candidate in the set of probable source vehicles (Lines 2–8 of Procedure 1). Note that the threshold is determined based on the drastic change in energy consumption and fixed at 70% in this paper as adopted in [19]. The determination of actual threshold is beyond the scope of this work. It is noteworthy to mention that the vehicles, whose utilization are above 70% for a small interval of time, are also potential candidates in the set of probable source vehicles. In order to avoid this, Procedure 1 calls the Procedure 2 to check the suitability of the source vehicles (Line 11).

Procedure 1: $FIND\text{-}SOURCE\text{-}VEHICLES(VU)$

```
1.  Set $npsv = 0$, $nfsv = 0$, $SPSV[\,] = 0$ and $V_{src} = 0$
2.  for $i = 1, 2, 3, \ldots, n$ do
3.    for $k = 1, 2, 3, \ldots, t$ do
4.      if $VU[i, k] \geq 70$ then
5.        $npsv += 1$; $SPSV[npsv] = i$
6.      endif
7.    endfor
8.  endfor
9.  if $npsv > 0$ then
10.   for $i = 1, 2, 3, \ldots, npsv$ do
11.     $flag =$ Call $CHECK\text{-}SUITABILITY\text{-}OF\text{-}SOURCE\text{-}VEHICLES(SPSV[i], t)$
12.     if $flag == TRUE$ then
13.       $nfsv += 1$; $V_{src}[nfsv] = i$
14.     endif
15.   endfor
16. endif
```

Procedure 2 checks the suitability of source vehicles by determining the rolling change in utilization (Lines 2–8 of Procedure 2). The rolling change is calculated using the equation given in Line 5. Note that it calculates the utilization for every instance of time and every duration of time. Then the rolling change values are averaged (Line 7). Next, this procedure calculates the standard deviation of average values and compares the product of standard deviation and 100 with the product of standard deviation of utilization of vehicle and a system parameter, λ (Line 10). If the later value is more, then the vehicle is suitable for a source vehicle (Line 11 of Procedure 2 and Lines 12–14 of Procedure 1). Note that determination of actual value of the system parameter is beyond the scope of this work. Otherwise, the vehicle is removed from the set of probable source vehicles.

Procedure 2: *CHECK-SUITABILITY-OF-SOURCE-VEHICLES* (*veh, t*)

```
1.  $AVG[t] = 0$
2.  for $k = 1, 2, 3, \ldots, t-1$ do
3.      $temp = 0$
4.      for $k' = 1, 2, 3, \ldots, t-1$ do
5.          $temp = temp + \mathrm{POWER}\left(\left(\frac{VU[veh, k+k']}{VU[veh, k']}\right), \frac{1}{k}\right) - 1$
6.      endfor
7.      $AVG[k] = \frac{temp}{t-k}$
8.  endfor
9.  $stdroll = STD(AVG)$
10. if $(stdroll \times 100) < (STD(VU[veh, 1]), VU[veh, 2], \ldots, VU[veh, t]) \times \lambda)$ then
11.     return $TRUE$
12. else
13.     return $FALSE$
14. endif
```

Next, the algorithm calls Procedure 3 to find a suitable VM from the final set of source vehicles (Line 7 of Algorithm 1). For this, it determines the standard deviation of VM utilization (Lines 4–6 of Procedure 3) and arranging them in descending order of their standard deviation (Line 7). Then it finds the highest standard deviation and make that VM as a potential candidate for VM migration (Lines 10–12). However, we have not used any elastic computing theory concept to find the potential VM for migration. Next, the suitability of the same vehicle is further checked (Lines 13–21). If the vehicle is still suitable, then another potential candidate is determined by following the above process (Lines 9–22). Otherwise, the vehicle is removed from the set of final source vehicles.

Next, the algorithm calls Procedure 4 to find the destination vehicles so that the VM of the source vehicle can be migrated to one of the destination vehicles. For this, it determines the average utilization of all vehicles, excluding the vehicles that are present in the set of final source vehicles and sort them in the ascending order of their average utilization (Lines 2–7 of Procedure 4). Then the vehicle with lowest average utilization is selected as a destination vehicle (Line 9) and the migration process is carried out (Line 11). The suitability of the selected vehicle is checked to ensure that the destination vehicle should not become the source vehicle after the successful migration (Lines 10–23).

Procedure 3: *FIND-VM-FOR-MIGRATION(VU, VMU, V_{src})*

```
1.  $VM_{mig}[\,] = 0,\ ncvmm = 0$
2.  for $i = 1, 2, 3, \ldots, nfsv$ do
3.    $veh = V_{src}[i],\ STDVM[\,] = 0$
4.    for $j = 1, 2, 3, \ldots, M[i]$ do
5.      $STDVM[j] = STD(VMU[i, j, 1],\ VMU[i, j, 2], \ldots, VMU[i, j, t])$
6.    endfor
7.    $ASTDVM = SORT(STDVM)$
8.    $check = 1$
9.    while $check == 1$ do
10.     $ncvmm += 1$
11.     $VM_{mig}[nvmm] = ASTDVM[M[i]]$
12.     $M[i] -= 1$
13.     for $k = 1, 2, 3, \ldots, t$ do
14.      if $VU[veh, k] \geq 70$ then
15.       $flag =$ Call $CHECK\text{-}SUITABILITY\text{-}OF\text{-}SOURCE\text{-}VEHICLES(veh, t)$
16.       break
17.      endif
18.     endfor
19.     if $flag == FALSE$ then
20.      $check = 0$
21.     endif
22.   endwhile
23. endfor
```

Procedure 4: *FIND-DESTINATION-VEHICLES(VU)*

```
1.  $V_{dest}[\,] = 0;\ npdv = 0;\ AVGUTI[\,] = 0;\ nvmm = 0$
2.  for $i = 1, 2, 3, \ldots, n$ do
3.    if $V[i] \notin V_{src}$ then
4.      $AVGUTI[i] = \frac{1}{t}\sum_{k=1}^{t} VU[i, k];\ npdv += 1$
5.    endif
6.  endfor
7.  $V_{dest} = SORT(AVGUTI)$
8.  for $i = 1, 2, 3, \ldots, npdv$ do
9.    $veh = V_{dest}[i]$
10.   for $j = 1, 2, 3, \ldots, ncvmm$ do
11.    $M[i] += 1; nvmm += 1$
12.    $V_{veh}[M[i]] = VM_{mig}[ncvmm]$
13.    for $k = 1, 2, 3, \ldots, t$ do
14.     if $VU[veh, k] \geq 70$ then
```

(Continued)

```
15.          $flag =$ Call $CHECK\text{-}SUITABILITY\text{-}OF\text{-}SOURCE\text{-}VEHICLES(veh, t)$
16.          break
17.        **endif**
18.      **endfor**
19.      **if** $flag == TRUE$ **then**
20.        $M[i] -= 1$
21.        break
22.      **endif**
23.    **endfor**
24.  **endfor**
```

Illustration

We illustrate the proposed algorithm using four vehicles (i.e., V_1, V_2, V_3, and V_4) that is present in a grid. Here, each vehicle can host one or more VMs. The proposed algorithm first calculates the resource utilization of the VMs that are hosted by the vehicles. These values are shown up to time instance $t = 10$ in Tab. 2, and their cumulative values with respect to vehicles and time instances are shown in Fig. 3. Then the proposed algorithm calls Procedure 1 to find the set of probable source vehicles. Here, vehicles V_1, V_2 and V_3 are the set of probable source vehicles as their resource utilization values exceed 70%.

Table 2: Resource utilization of the VMs

	V_1			V_2			V_3			V_4
Time	VM_1	VM_2	VM_3	VM_1	VM_2	VM_3	VM_1	VM_2	VM_3	VM_1
1	16	10	39	04	03	05	03	08	09	01
2	23	18	45	13	22	30	08	07	10	08
3	11	09	36	15	18	43	07	11	14	04
4	04	03	05	21	33	35	09	10	20	07
5	14	13	28	9	11	12	10	9	24	03
6	09	08	27	16	26	27	08	13	48	06
7	11	09	28	19	30	25	12	10	56	03
8	12	07	13	17	29	48	17	15	49	02
9	26	11	21	20	25	42	15	19	54	02
10	05	07	10	14	18	49	16	17	62	03

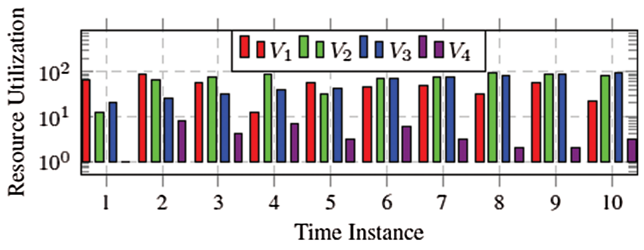

Figure 3: Resource utilization of vehicles

Now, the algorithm calls Procedure 2 to check the suitability of the vehicles as source vehicle(s). For example, let us check the suitability of the vehicle V_3. For this, the algorithm calculates the rolling change (RC) in the resource utilization. For V_3, resource utilizations are 20, 25, 32, 39, 43, 69, 78, 81, 88 and 95 for time instance 1 to time instance 10, respectively. Suppose $k = 1$ and $k' = 1$, RC_3 is calculated as $\left(\frac{25}{20}\right)^{\frac{1}{1}} - 1 = 0.2500$. Note that RC is calculated for each time instance k and these values are $\left(\frac{32}{25}\right)^{\frac{1}{1}} - 1 = 0.2800$, $\left(\frac{39}{32}\right)^{\frac{1}{1}} - 1 = 0.2188$, 0.1026, 0.6047, 0.1304, 0.0385, 0.0864 and 0.0795, respectively. Then these RC values are averaged as 0.1990 for $k = 1$. Similarly, RC values are calculated for $k = 2$ (i.e., $\left(\frac{32}{20}\right)^{\frac{1}{2}} - 1 = 0.2649$, 0.2490, 0.1592, 0.3301, 0.3468, 0.0835, 0.0622 and 0.0830) to $k = 9$ (i.e., $\left(\frac{95}{20}\right)^{\frac{1}{9}} - 1 = 0.1890$), respectively and the average values, for $k = 2$ to $k = 9$, are 0.1973, 0.1981, 0.2048, 0.2178, 0.2037, 0.1954, 0.1925 and 0.1890, respectively.

Next, the algorithm finds standard deviation of all the averaged values as 0.0084 (i.e., SD_{arc}) and standard deviation of the utilization as 14.0500 (i.e., SD_{ru}) of the vehicle V_3. As ($SD_{ru} \times \lambda = 14.0500 \times 0.5000$ (Let) $= 7.0250) > (SD_{arc} \times 100 = 0.0084 \times 100 = 0.8400$), the vehicle V_3 is suitable for a source vehicle. Note that λ is a system parameter and it is set as 0.5 for this illustration.

The vehicle V_3 hosts three VMs (i.e., VM_{31}, VM_{32} and VM_{33}). Now, the algorithm finds the suitable VM(s) that is hosted on vehicle V_3 by taking the standard deviation of VM utilization and arranging them in descending order of their standard deviation. The standard deviation of VM_{31}, VM_{32} and VM_{33} is 4.4500, 3.9800 and 21.0400, respectively. As VM_{33} contains the highest standard deviation, it is the potential candidate for VM migration. Once the successful migration of VM_{33} is over, vehicle V_3 is no more suitable for a source vehicle to carry out further migration.

Now, the algorithm finds a destination vehicle to migrate VM_{33}. For this, it finds the average utilization of all vehicles, excluding the vehicles that are present in the set of final source vehicles and sort those vehicles in the ascending order of their average utilization. In this illustration, V_4 is only available. Note that VM_{33} can be migrated to vehicle V_4 if the utilization of vehicle V_4 after accommodating VM_{33} should not be a potential candidate for the set of probable source vehicles. Here, VM_{33} is migrated to vehicle V_4 and vehicle V_3 is removed from the set of final source vehicles after this successful migration. As a result, the total number of VM candidates is one, i.e., VM_{33}.

From the set of probable source vehicles (refer Fig. 3), it is clearly visible that the change in utilization for vehicle V_3 is stable, but change in utilization for vehicle V_1 and vehicle V_2 is not stable. In this illustration, our proposed algorithm RU-VMM finds vehicle V_3 as a source vehicle. Note that apart from vehicle V_3, other vehicles, that are present in set of probable source vehicles, are not suitable for migration. We run the same illustration for the existing threshold-based algorithm [15,19–21] and CUSUM algorithm [14]. Here, threshold-based algorithm finds VM_1, VM_2 and VM_3 as source vehicles and CUSUM algorithm finds VM_2 and VM_3 as source vehicles. The detailed comparison of proposed and existing algorithms is given in the Tab. 3.

Table 3: Comparison of Comparison of number of source vehicles and percentage of successful/dropped migration for RU-VMM, threshold and CUSUM algorithms

Performance metrics	RU-VMM	Threshold [15,19]	CUSUM [14]
Number of final source vehicles	1	3	2
Percentage of successful VM migration	100.00	33.33	50.00
Percentage of dropped VM migration	00.00	66.66	50.00

5 Performance Metrics

In this section, we present three performance metrics to compare the proposed and existing algorithms. Their definitions are presented in the following subsections.

5.1 Number of Final Source Vehicles

The number of final source vehicles (*nfsv*) is the total number of suitable vehicles that are selected as potential candidates to carry out the process of migration.

5.2 Percentage of Successful VM Migration

The percentage of successful VM migration (*psvmm*) is the ration between the number of VMs that are successfully placed at the destination vehicle and the total number of VM candidates.

5.3 Percentage of Dropped Migration

The percentage of dropped VM migration (*pdvmm*) is the ration between the number of VMs that are not successfully placed and the total number of VM candidates.

6 Simulation Results

We create a virtual environment using MATLAB R2017a (version 9.2), on a computer system with Intel(R) core(TM) i7-4790 CPU@3.60 GHz 3.60 GHz, 8 GB RAM, 64-bit Windows 10 operating system for the simulation of the proposed algorithm RU-VMM. We also use MATLAB R2017a to generate nine datasets by considering three different types of situation, namely low congestion, medium congestion and high congestion of vehicles. Note that these datasets are named using the total number of vehicles in a grid (e.g., 500, 1000, 1500, 5000 etc.). Each dataset further contains three instances of same size. We follow the Monte–Carlo simulation method and uniform distribution to generate the datasets [5,13]. The resource utilization is collected for one hour and the average resource utilization is determined in each five minutes for the simplicity of simulation. The detail of the parameters and their respective values is shown in Tab. 4. In addition, we use simulation of urban mobility (SUMO) traffic simulator 1.6.0 to generate the parameters, such as timestamp, lane, position, speed, vehicle acceleration/deceleration and vehicle length, and network simulator OMNeT++. We also use IEEE 802.11p with ITS band of 5.9 GHz to exchange the data between the vehicles and/or the RSUs [30]. This band enables wireless access in vehicular environments. We use a open source hybrid simulation framework, called Veins, which uses IEEE 802.11p for communication and to integrate two simulators, namely SUMO and OMNeT++ using traffic control interface (TraCI) [31]. In the simulation process, we consider a city map (Bhubaneswar, Odisha) as shown in Fig. 4, which consists of a set of lanes. The other configuration parameters are shown in Tab. 5.

Table 4: Parameters and their respective values

Parameter	Values
# of Vehicles	Low: [500, 1000, 1500] Medium: [5000, 10000, 15000] High: [50000, 100000, 500000]
# of VMs	[3∼1] VMs per vehicle
Duration	1 h
Resource utilization of VMs	[1∼99]

Figure 4: A city map

Table 5: Parameters and their respective values in SUMO and OMNeT++

Parameter	Values	Parameter	Values
Area	1500×1500 m^2	Number of lanes	$2 \times$ Direction
Speed of vehicles	25 m/s	Vehicle acceleration	3.0 m/s^2
Vehicle deceleration	6.0 m/s^2	Number of vehicles	500∼5000
Length of vehicles	3.5 and 4.5 m	Simulation time	3600 s
Number of simulations	3	Bitrate	5 Mbps
Communication range of vehicles	250 m		

The simulation results of the proposed algorithm, RU-VMM and two existing algorithms, threshold and CUSUM are compared in terms of number of final source vehicles and percentage of successful/dropped VM migration as shown in Figs. 5 and 6, respectively. Note that we present the average of three instances per dataset for the comparison of the proposed and existing algorithms. The simulation results show the better performance of the proposed algorithm RU-VMM in comparison to the existing algorithms. RU-VMM algorithm improves 16.91% than the CUSUM algorithm and 71.59% than the threshold algorithm in terms of *nfsv*, and 20.62% than the

CUSUM algorithm and 275.34% than the threshold algorithms in terms of *psvmm*. The simulation results are also carried out using SUMO and OMNeT++, and compared in terms of percentage of dropped VM migration as shown in Fig. 7. Here also, the proposed algorithm outperforms the existing algorithms. The rationality behind this better performance is as follows. (1) The proposed algorithm finds the RC of set of probable vehicles to determine continuous increase in utilization. (2) The proposed algorithm uses standard deviation to monitor the abrupt change of VMs. (3) The proposed algorithm restricts the VM candidates by continuously monitoring the utilization after each successful migration.

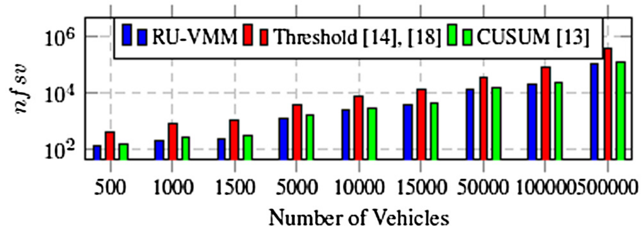

Figure 5: Graphical comparison of *nfsv* for RU-VMM, threshold and CUSUM algorithms

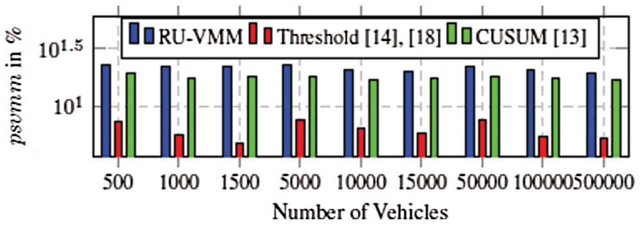

Figure 6: Graphical comparison of *psvmm* for RU-VMM, threshold and CUSUM algorithms

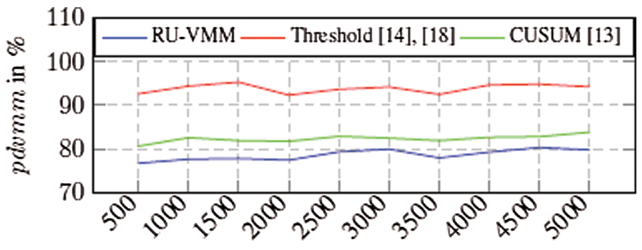

Figure 7: Graphical comparison of *pdvmm* for RU-VMM, threshold and CUSUM algorithms

7 Conclusion

In this paper, we have presented a novel algorithm RU-VMM for VM migration in vehicular cloud computing. The objective of the proposed algorithm is to migrate the load from the overloaded vehicles to the under loaded vehicles. For this, RU-VMM creates a set of probable sources vehicles and monitors the rolling change in resource utilization to select the final source vehicles. Then it finds a destination vehicle to migrate the VM by continuously monitoring the resource utilization of the destination vehicle and the source vehicle. The proposed algorithm has been simulated using MATLAB and compared with two existing algorithms, threshold and CUSUM using twenty-seven instances of nine dataset. The comparison has been shown in terms of three performance metrics, namely number of final source vehicles, percentage of successful migration(s) and percentage of dropped migration(s). The comparison of simulation results has shown that the proposed algorithm RU-VMM outperforms the two existing algorithms in terms of three performance metrics.

The proposed algorithm has not considered the bandwidth and delay associated with VM migration, which can be considered as a future work. Moreover, determining an appropriate threshold to prepare a set of probable source vehicles is very much interesting and beyond the scope of this paper. It can be further investigated to develop more efficient algorithm. We have not considered the parameters associated with driving strategies, such as speed, acceleration, path and many more, which can be considered as a future work. We have also not considered the road accident and emergency scenarios in our proposed model and algorithm, and not tested the proposed algorithm in the actual environment. The congestion scenarios can be generated using SUMO by taking the number of vehicles and speed of the vehicles with respect to the traffic routes. We will extend our work by incorporating these scenarios in our future work.

Funding Statement: Taif University Researchers Supporting Project number (TURSP-2020/36), Taif University, Taif, Saudi Arabia.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] F. Hagenauer, T. Higuchi, O. Altintas and F. Dressler, "Efficient data handling in vehicular micro clouds," *Ad Hoc Networks*, vol. 91, pp. 101871, 2019.
[2] T. Kim, H. Min, E. Choi and J. Jung, "Optimal job partitioning and allocation for vehicular cloud computing," *Future Generation Computer Systems*, vol. 108, pp. 82–96, 2020.
[3] G. J. L. Paulraj, S. A. J. Francis, J. D. Peter and I. J. Jebadurai, "Resource-aware virtual machine migration in IoT cloud," *Future Generation Computer Systems*, vol. 85, pp. 173–183, 2018.
[4] S. K. Panda and P. K. Jana, "Normalization-based task scheduling algorithms for heterogeneous multi-cloud environment," *Information Systems Frontiers*, vol. 20, no. 2, pp. 373–399, 2018.
[5] S. K. Panda, S. K. Pande and S. Das, "Task partitioning scheduling algorithms for heterogeneous multi-cloud environment," *Arabian Journal for Science and Engineering*, vol. 43, no. 2, pp. 913–933, 2018.
[6] S. K. Panda and P. K. Jana, "An energy-efficient task scheduling algorithm for heterogeneous cloud computing systems," *Cluster Computing*, vol. 22, no. 2, pp. 509–527, 2019.
[7] A. Boukerche and E. Robson, "Vehicular cloud computing: Architectures, applications, and mobility," *Computer Networks*, vol. 135, pp. 171–189, 2018.
[8] S. K. Panda and P. K. Jana, "Efficient task scheduling algorithms for heterogeneous multi-cloud environment," *Journal of Supercomputing*, vol. 71, no. 4, pp. 1505–1533, 2015.
[9] M. Whaiduzzaman, M. Sookhak, A. Gani and R. Buyya, "A survey on vehicular cloud computing," *Journal of Network and Computer Applications*, vol. 40, pp. 325–344, 2014.
[10] N. Garg, D. Singh and M. S. Goraya, "Power and resource-aware vm placement in cloud environment," in *IEEE 8th Int. Advance Computing Conf.*, Greater Noida, India, IEEE, pp. 113–118, 2018.
[11] A. N. Asadi, M. A. Azgomi and R. Entezari-Maleki, "Analytical evaluation of resource allocation algorithms and process migration methods in virtualized systems," *Sustainable Computing: Informatics and Systems*, vol. 25, pp. 100370, 2020.

[12] S. Sahoo, A. Pattanayak, K. S. Sahoo, B. Sahoo and A. K. Turuk, "MCSA: A multi-constraint scheduling algorithm for real-time task in virtualized cloud," in *2018 15th IEEE India Council Int. Conf.*, Coimbatore, India, pp. 1–6, 2018.

[13] T. K. Refaat, B. Kantarci and H. T. Mouftah, "Virtual machine migration and management for vehicular clouds," *Vehicular Communications*, vol. 4, pp. 47–56, 2016.

[14] M. Andreolini, S. Casolari, M. Colajanni and M. Messori, "Dynamic load management of virtual machines in cloud architectures," in *Int. Conf. on Cloud Computing*, Beijing, China, Springer, pp. 201–214, 2009.

[15] G. Khanna, K. Beaty, G. Kar and A. Kochut, "Application performance management in virtualized server environments," in *IEEE/IFIP Network Operations and Management Symp.*, Vancouver, BC, Canada, IEEE, pp. 373–381, 2006.

[16] M. S. Mekala and P. Viswanathan, "Energy-efficient virtual machine selection based on resource ranking and utilization factor approach in cloud computing for IoT," *Computers & Electrical Engineering*, vol. 73, pp. 227–244, 2019.

[17] S. Y. Hsieh, C. S. Liu, R. Buyya and A. Y. Zomaya, "Utilization-prediction-aware virtual machine consolidation approach for energy-efficient cloud data centers," *Journal of Parallel and Distributed Computing*, vol. 139, pp. 99–109, 2020.

[18] N. Hamdi and W. Chainbi, "A survey on energy aware vm consolidation strategies," *Sustainable Computing Informatics and Systems*, vol. 23, pp. 80–87, 2019.

[19] C. H. Hsu, K. D. Slagter, S. C. Chen and Y. C. Chung, "Optimizing energy consumption with task consolidation in clouds," *Information Sciences*, vol. 258, pp. 452–462, 2014.

[20] T. Wood, P. J. Shenoy, A. Venkataramani and M. S. Yousif, "Black-box and gray-box strategies for virtual machine migration," *NSDI*, vol. 7, pp. 17, 2007.

[21] N. Bobroff, A. Kochut and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in *2007 10th IFIP/IEEE Int. Symp. on Integrated Network Management*, Munich, Germany, IEEE, pp. 119–128, 2007.

[22] H. Chen, X. Zhu, H. Guo, J. Zhu, X. Qin *et al.,* "Towards energy-efficient scheduling for real-time tasks under uncertain cloud computing environment," *Journal of Systems and Software*, vol. 99, pp. 20–35, 2015.

[23] X. Wu, Y. Zeng and G. Lin, "An energy efficient vm migration algorithm in data centers," in *2017 16th Int. Symp. on Distributed Computing and Applications to Business, Engineering and Science*, AnYang, China, IEEE, pp. 27–30, 2017.

[24] S. Nithya, M. Sangeetha, K. N. Apinaya Prethi, K. S. Sahoo, S. K. Panda *et al.,* "SDCF: A software-defined cyber foraging framework for cloudlet Environment," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2423–2435, 2020.

[25] T. Mekki, I. Jabri, A. Rachedi and M. ben Jemaa , "Vehicular cloud networks: Challenges, architectures, and future directions," *Vehicular Communications*, vol. 9, pp. 268–280, 2017.

[26] Z. Li, X. Yu, L. Yu, S. Guo and V. Chang, "Energy-efficient and quality-aware vm consolidation method," *Future Generation Computer Systems*, vol. 102, pp. 789–809, 2020.

[27] Y. Sharma, W. Si, D. Sun and B. Javadi, "Failure-aware energy-efficient vm consolidation in cloud computing systems," *Future Generation Computer Systems*, vol. 94, pp. 620–633, 2019.

[28] S. K. Mishra, D. Puthal, B. Sahoo, P. P. Jayaraman, S. Jun *et al.,* "Energy-efficient vm-placement in cloud data center," *Sustainable Computing Informatics and Systems*, vol. 20, pp. 48–55, 2018.

[29] I. Mohiuddin and A. Almogren, "Workload aware vm consolidation method in edge/cloud computing for iot applications," *Journal of Parallel and Distributed Computing*, vol. 123, pp. 204–214, 2019.

[30] T. Begin, A. Busson, I. Lassous and A. Boukerche, "Delivering video-ondemand services with ieee 802.11p to major non-urban roads: A stochastic performance analysis," in *Computer Networks*, Elsevier, 2020.

[31] A. F. Acosta, J. E. Espinosa and J. Espinosa, "Traci4matlab: Enabling the integration of the sumo road traffic simulator and matlab$^{®}$ through a software re-engineering process, " in *Modeling Mobility with Open Data*, Berlin, Germany: Springer, pp. 155–170, 2015.