CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2015; **27**:5780–5792
Published online 7 October 2015 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.3642

SPECIAL ISSUE PAPER

Migrate or not? Exploring virtual machine migration in roadside cloudlet-based vehicular cloud

Hong Yao, Changmin Bai, Deze Zeng*,†, Qingzhong Liang and Yuanyuan Fan

School of Computer Science, China University of Geosciences, Wuhan, China

SUMMARY

Vehicle Ad-Hoc Networks (VANET) enable all components in intelligent transportation systems to be connected so as to improve transport safety, relieve traffic congestion, reduce air pollution, and enhance driving comfort. The vision of all vehicles connected poses a significant challenge to the collection, storage, and analysis of big traffic-related data. Vehicular cloud computing, which incorporates cloud computing into vehicular networks, emerges as a promising solution. Different from conventional cloud computing platform, the vehicle mobility poses new challenges to the allocation and management of cloud resources in roadside cloudlet. In this paper, we study a virtual machine (VM) migration problem in roadside cloudlet-based vehicular network and unfold that (1) *whether* a VM shall be migrated or not along with the vehicle moving and (2) *where* a VM shall be migrated, in order to minimize the overall network cost for both VM migration and normal data traffic. We first treat the problem as a static off-line VM placement problem and formulate it into a mixed-integer quadratic programming problem. A heuristic algorithm with polynomial time is then proposed to tackle the complexity of solving mixed-integer quadratic programming. Extensive simulation results show that it produces near-optimal performance and outperforms other related algorithms significantly. Copyright © 2015 John Wiley & Sons, Ltd.

Received 18 July 2015; Accepted 25 July 2015

KEY WORDS: vehicular networks; vehicular cloud computing; virtual machine migration; cloudlet

1. INTRODUCTION

Vehicular networks are in the progress of merging with the Internet to constitute a fundamental information platform that is an indispensable part of the intelligent transport system [1]. With the development of cloud computing, dedicated short-range communication techniques, and emerging mobile applications, vehicular networks with the capabilities of decision-making and autonomous control can be upgraded to cloud-assisted vehicular cyber physical systems [2], which can be regarded as an evolution of vehicular networks by combining mobile sensing, wireless networking, and mobile/cloud computing and communications and network technologies to improve transport safety, relieve traffic congestion, reduce air pollution, and enhance the comfort of driving [2–4]. Meanwhile, many emerging applications, including in-vehicle multimedia entertainment, vehicular social networking, and location-based services, demand complex computation and larger storage. However, vehicles are normally constrained for resources of computation, storage, and radio spectrum bandwidth because of the requirement of small-size and low-cost hardware systems. In order to efficiently support these applications, a promising solution is to incorporate cloud computing into vehicular networks to build vehicular cloud computing (VCC) [5]. In VCC, we can fully or partially

*Correspondence to: Deze Zeng, School of Computer Science, China University of Geosciences, No.388, Lumo Road, Wuhan, 430074, China.
†E-mail: deze@cug.edu.cn

offload a vehicular application to a remote cloud to reduce the burden on the vehicle. However, such paradigm suffers low response time and high communication cost. To tackle this issue, roadside cloudlet (RSC) is introduced as auxiliary nearby cloud resources for users [6, 7]. Moreover, an emerging paradigm called fog computing is proposed to extend the cloud computing to the network edge such that users can enjoy cloud services locally with low service response time [8–11].

Resource virtualization is one of the key enabling technology for cloud resource allocation and management. It allows simultaneous execution of diverse tenant's tasks over a shared hardware platform in the form of a virtual machine (VM). To facilitate load balancing, fault management, server maintenance, and so on, live VM migration is advocated in traditional cloud data center [12] [13]. However, the key differences of RSC-based VCC and traditional cloud computing platform are that (1) the cloud resources are highly distributed and (2) the users are highly dynamic. This makes VM migration in RSC-based VCC more challenging. Suppose each vehicle has one corresponding VM in VCC to support it. In the consideration of network cost, one critical question is that whether the VM shall be migrated along with the vehicle moving. If so, we shall further consider where the VM shall be migrated. Obviously, if a VM always locates in an RSC, higher network cost may be introduced due to long distance communication resulting from normal traffic data between a vehicle and its corresponding VM. We define such cost as *execution cost* in this paper. On the contrary, if we greedily migrate the VM along with the moving vehicle, higher network cost may be also introduced because of VM migration. We consequently call this cost as *migration cost*. In this paper, we are motivated to study the minimum network cost in VM migration problem. To the best of our knowledge, existing works usually focus on VM migration issues to meet some other objectives, such as task completed time and dynamic resource management [14–16]. We are the first to consider whether VM migration or not to meet the minimum network cost in VCC.

The main contributions of this paper are as follows:

- We study VM migration problem for minimizing network cost in RSC-based VCC and formulate it as a mixed-integer quadratic programming (MIQP) problem.
- Because the formulated MIQP is computation-prohibitive when large numbers of vehicles are involved, to tackle the computation complexity, we further propose a polynomial time *two-phase* heuristic algorithm.
- Via extensive simulations, the performance results show the high efficiency of our algorithm by the fact that it much approaches the optimal one and substantially outperforms other algorithms.

The remainder of this paper is structured as follows. We next review some related work in VCC and VM migration in Section 2. Section 3 introduces the system model and problem statement. Section 4 formally proposes the VM migration formulation. Section 5 presents our *two-phase* heuristic algorithm. Section 6 gives the performance evaluation results. Finally, Section 7 concludes this paper.

2. RELATED WORK

2.1. Vehicular cloud computing

Mobile cloud computing (MCC) [17], with its advance to meet the urgent need for richer applications and services of resource-constrained mobile devices, is emerging as a new computing paradigm and has recently attracted significant attention. Sanaei *et al.* [18] give the comprehensive definition of MCC as a rich mobile computing technology that leverages unified elastic resources of varied clouds and network technologies toward unrestricted functionality, storage, and mobility to serve a multitude of mobile devices anywhere, anytime, through the channel of Ethernet or Internet. Cloudlet, envisioned as a new MCC scheme, brings the cloud to the mobile user [19]. Satyanarayanan *et al.* [7] propose to offload computation workload on a mobile device to a nearby cloud system by dynamic VM synthesis. They also explore the survivability of mobile computing in hostile environments such as military applications and disaster recovery. Fesehaye *et al.* [20] focus on file editing, video streaming, and collaborative chatting, which are representative enterprise application scenarios to study the impact of cloudlets in interactive mobile cloud applications.

DOI: 10.1002/cpe

Li *et al.* [6] study further the fundamental mobile cloudlet properties that unfold whether and when a mobile cloudlet can provide mobile application service in mobile ad-hoc network. In our problem, we adopt the cloudlet-based roadside clouds to provide services/resources to the mobile vehicles. Same with cloudlets that bring the cloud to the mobile users, fog computing [8] extends the cloud to the edge of network to reduce network delay and task execution time for computation-sensitive and delay-sensitive applications.

Recently, many efforts are devoted to study the combination of cloud computing and vehicular networks. The concept of VCC was first proposed by Olariu *et al.* [21]. They propose the concept of autonomous vehicular clouds by exploiting the under-utilized resources in VANETs to offer different VANET applications to users. They also briefly discuss the research challenge in vehicular clouds. A platform-as-a-service model is designed in [22] to support cloud services for mobile vehicles. Then, Hussain *et al.* [23] take a step further and put forward the taxonomy of VANET-based clouds into three large architectural frameworks, that is, vehicular clouds, vehicles using clouds, and hybrid clouds. Vehicles act as cloud service providers and clients in vehicular clouds, vehicles using clouds, and hybrid clouds, respectively. Yu *et al.* [5] invent the hierarchical cloud architecture for vehicular networks. The proposed architecture includes a vehicular cloud, a roadside cloud, and a central cloud. On this basis, they further propose a resource reservation scheme for the resource re-relocation during VM migration. However, none of them discuss the VM migration along with vehicle moving, which we think is critical to the performance and cost of RSC-based VCC. We are motivated to discuss this issue.

2.2. VM migration problem

Virtualization is a key technique to improve QoS in could computing [24]; it provides a virtualized view of resources used to instantiate VMs. One important feature of virtualization is live VM migration, which can be employed to facilitate load balancing, fault management, and server maintenance [12, 13, 25]. Many problems about VM migration have been studied in traditional cloud systems. For instance, optimizing the impact of migration on VM communication cost [13, 26–28], efficient scheduling for real time jobs to minimize the completed time [14, 25, 29]. Firstly, Zhang *et al.* [13] present an optimization model that focuses on jointly maximizing resource utilization and minimizing migration cost in the data center. Another, Zhang *et al.* [26] introduce an online migration algorithm that can maximize the number of overcommitted VMs, while minimizing the impact on the communication cost among VMs. Similarly, Liu *et al.* [12] present a design and implementation of a novel slowdown scheduling algorithm for Xen live VM migration platform. In work [27], the authors consider the interdependencies and inter-communication patterns among VMs and propose an application-aware VM migration policy aimed to minimize the data center network traffic while satisfying all of the server-sides constrains. Li *et al.* [28] further consider the utilization of physical machines and jointly optimize the network traffics and physical machine cost to find the placement scheme of VMs. Besides, jobs scheduling to minimize the completed time for cloud computing also attracts many attentions. For instance, in order to study the heterogeneity of in cloud data center, Li *et al.* [29] propose off-line and on-line VM placement methods, respectively, to minimize the completed time of tasks. Zhang *et al.* [25] further theoretically analyze how much bandwidth is required to guarantee the total migration time and downtime of live VM migration. The works in the previous text all focus on traditional cloud computing. However, VM migration in mobile cloud computing, especially in VCC, needs to be addressed due to the mobility of the mobile nodes (vehicles) and the topology of a vehicular cloud data center network changes more rapidly. Refaat *et al.* [15] study the VM migration in vehicular cloud in which the cloud nodes are self-organized by vehicles. Besides, Gkatzikis *et al.* [14] investigate the potential task of VM migration to reduce contention for the shared resources so as to minimize execution time of mobile tasks. They propose mobile cloud computing architecture. The most related work to ours is [5], in which authors propose the hierarchical cloud architecture and discuss the scenario of VM migration and resource re-relocation. Different from existing work on mobile cloud, we focus on the network cost in VCC with the consideration of tradeoff between *execution cost* and *migration cost*.

DOI: 10.1002/cpe

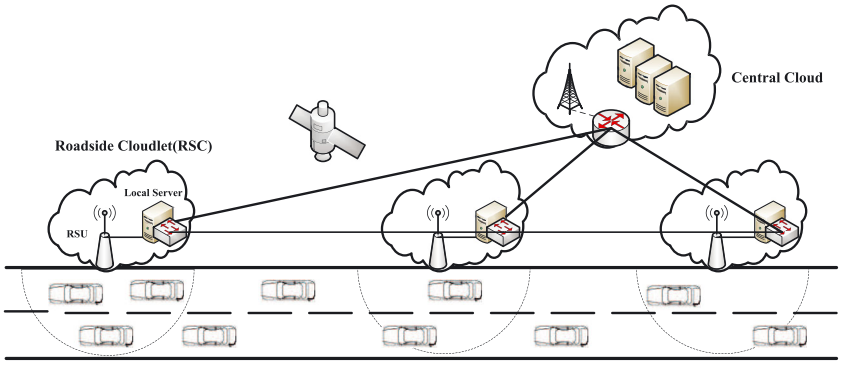


Figure 1. Cloudlet-based vehicular networks architecture.

3. SYSTEM MODEL AND PROBLEM STATEMENT

In this section, we briefly describe the VM migration scenarios and propose our system model.

3.1. Virtual machine migration in VCC

In this paper, we adopt a hierarchical VCC architecture shown in Figure 1, which is first proposed in [5]. There are two layers: central cloud and RSCs. A RSC is composed of two main parts: dedicated local server and roadside unit (RSU). The dedicated local server virtualizes physical resources and acts as a potential cloudlet. An RSU provides radio interfaces for vehicles to access the cloud (i.e., Vehicle-to-RSU (V2R) communication). It is flexible and compatible for vehicles to use different communication technologies (e.g., Dedicated Short Range Communication (DSRC), 3G, 4G, and LTE-advanced) to access cloud services. To utilize RSC resources, a vehicle can select any RSC and customize a transient cloud on it. Here, we call the customized cloud as *transient cloud* because the cloud can only serve vehicles for a while. When a vehicle customizes a transient cloud from an RSC, it is offered by virtual resources in the form of VM. During service period, as the vehicle moves, it may switch between different RSCs. Under such condition, for service continuity, the customized VM may be transferred across the RSCs. This process is referred as VM migration. To migrate a VM, its memory image has to be shifted from the source to the destination RSC [16, 29]. Consider a VM migration toy example in Figure 2. When vehicle-A within the communication range of RSU-1 wants to use cloud services, it customs VM-A. If it moves into RSU-2, there are two ways for it to continuously use cloud services. One is VM migration, which will introduce migration cost (corresponding to the VM's image size) but may significantly reduce the normal traffic data cost, which is defined as execution cost in this paper. Another one is persistently keeping VM-A in RSC-1 and letting vehicle-A communicate with RSC-1 to acquire the services on VM-A. Obviously, the latter method has zero migration cost but may introduce much higher execution cost due to longer communication distance (or hops). Therefore, it is significant to find an appropriate VM migration policy such that the overall network cost can be minimized.

3.2. System model

In this paper, we consider an urban distributed network graph $G = (S, E)$ consisting of an RSC set S and a edge set E represents the communication links among these RSC nodes. A set H of moving vehicles want to use cloud services hosted by RSCs. A vehicle will establish a transient VM when it moves into the range of a RSC. We assume that a VM can only provide service for its corresponding vehicle. The corresponding VM set of vehicle set H is denoted as V. The communication between a vehicle and its corresponding VM is defined as a session. The session set for all vehicles and their VMs is denoted as I. Obviously, V, H and I are with the same cardinality. A RSC may host a certain number of VMs, constrained by its available resources. There are many different types of resources (e.g., computation, storage, memory, and Input/Output). The type set is denoted as R. Let

DOI: 10.1002/cpe

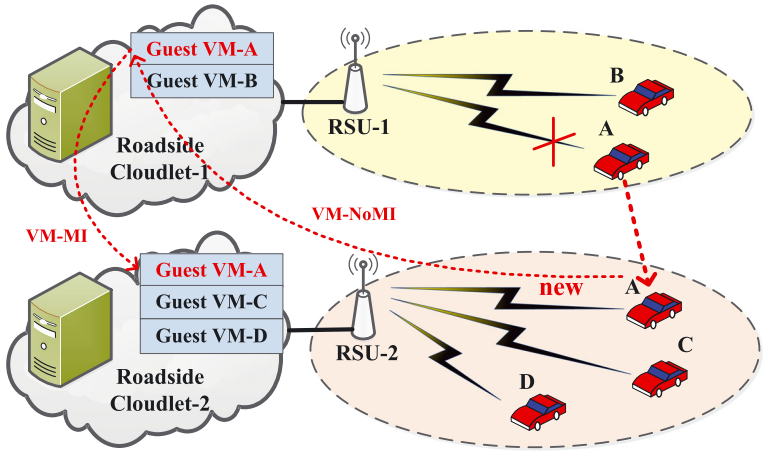


Figure 2. Virtual machine (VM) migration case study [5].

Table I. Notations.

Notation	Description
S	The RSCs set
V, H, I	The VM set, vehicle set, and session set
R	The resource set
A_{rs}	The capacity of resource $r \in R$ on RCS $s \in S$
Q_{ri}	The demand of resource $r \in R$ from VM of session $i \in I$
M_{uv}	The transmission capacity of link$(u, v) \in E$
C_i	Image size of VM i (migration cost)
T_i	Traffic rate of VM i (execution cost)
L_i	The candidate paths of session i
y_i^l	A binary variable indicating whether path l is selected for session i
x_u^{il}	A binary variable indicating whether VM i is placed on RSC u on path l
$h_{u,v}^i$	A binary variable indicating whether link (u, v) is adopted for VM migration of session i

A_{rs} be the capacity of resource $r \in R$ on RSC $s \in S$. Similarly, we define Q_{ri} as the demand of resource $r \in R$ from VM of session $i \in I$. Without loss of generality, we assume that all links between RSCs are also capacity-limited. Let $M_{uv} \geqslant 0$ represent the transmission capacity of link $(u, v) \in E$.

When a vehicle moves into another RSC, a cloud service provider shall make a decision on whether its corresponding VM shall be migrated or not. Obviously, to migrate a VM, its memory image must be transferred from the source to the destination, incurring migration cost, while the execution cost is inevitable and varies on the communication distance between the vehicle and its VM, depending on the VM placement decision. We denote VM image size and traffic rate in session $i \in I$ as C_i and T_i, respectively, which correspond to the migration cost and execution cost. All symbols and variables used in the paper are summarized in Table I for the convenience of readers.

Virtual machine migration is a dynamic process where the vehicle locations vary over the time. In this paper, for tractability, we consider a discrete time slot model that VM migration happens only at the end of one time slot, that is, decision moment, and the placement of all VMs in last time slot is known. In this case, whether the new placement decision of VM changes or not is equivalent to whether it shall be migrated or not. Suppose two time slots t_1 and t_2, where t_2 is the decision moment. The information, such as the placement decision of all VMs at slot t_1 and the locations of all vehicles (by mobility trace) at slot t_2, is known as a priori to the cloud service provider. As shown in Figure 3, for session $i \in I$, its initial location $s_i \in S$ at t_1 and the vehicle location $d_i \in S$ at t_2 are known. There exist multiple different paths between s_i and d_i. We define the path candidate set for session $i \in I$ as L_i. The placement location of VM will belong to one of the path from L_i (i.e., the path shown in Figure 3).

DOI: 10.1002/cpe

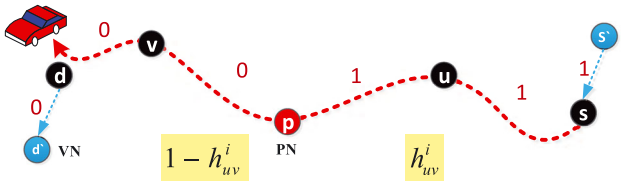


Figure 3. Virtual machine placement.

4. PROBLEM STATEMENT

In this section, we formulate the minimum network cost VM migration problem into a MIQP problem.

Path selection constraints: We first consider the path selection for a given session $i \in I$. We define a binary variable y_i^l to denote whether path $l \in L_i$ is selected for session $i \in I$ as

$$y_i^l = \begin{cases} 1, \text{ if path } l \in L_i \text{ is selected for session } i, \\ 0, \text{ otherwise.} \end{cases} \tag{1}$$

To ensure the communication between each vehicle and the corresponding VM, for each session $i \in I$, one path must be selected, that is,

$$\sum_{\forall l \in L_i} y_i^l = 1, \forall i \in I. \tag{2}$$

Virtual machine placement constraints: We use binary variable x_u^{il} to describe whether VM i is placed on RSC $u \in S$ on path l or not. That is,

$$x_u^{il} = \begin{cases} 1, \text{ if VM } i \text{ is placed on node } u \text{ along the selected path } l, \\ 0, \text{ otherwise.} \end{cases} \tag{3}$$

The VM can be only placed on the selected path. Therefore, we have

$$x_u^{il} \leqslant y_i^l, \forall i \in I, u \in S, l \in L_i. \tag{4}$$

Each VM must be placed onto exactly one RSC on the selected path. This can be expressed as

$$\sum_{l \in L_i} \sum_{\forall u \in l} x_u^{il} = 1, \ \forall i \in I. \tag{5}$$

Resource capacity constraints: An RSC is resources capacity-limited while VM has certain resources requirement. For either resource (e.g., $r \in R$) on an RSC (e.g., $v \in S$), the total resource requirement shall not exceed the provision capacity. In other words, the following constraints must be satisfied:

$$\sum_{i \in I} x_s^{il} \cdot Q_{ir} \leqslant A_{rs}, \forall l \in L_i, s \in S, r \in R. \tag{6}$$

Link capacity constraints: For each session, certain traffic must be transferred between the vehicle and its corresponding VM, in both directions. One thing we shall ensure is that the total traffic rate shall not overflow the available transmission capacity on each link. To this end, we first introduce a binary variable h_{uv}^i indicating whether link $(u, v) \in E$ is adopted to migrate for session $i \in I$ or not as

$$h_{uv}^i = \begin{cases} 1, \text{ if the link } (u, v) \text{ is adopted to migrate for session } i, \\ 0, \text{ otherwise.} \end{cases} \tag{7}$$

As s_i and d_i could be also the placement node. We add two *virtual node* s_i' before s_i and d_i' after d_i, as show in Figure 3, and

$$h_{s's}^i = 1, h_{dd'}^i = 0 \tag{8}$$

DOI: 10.1002/cpe

Note that for any link $(u, v) \notin L_i$, we always have $h^i_{uv} \equiv 0$. Besides, if the path where (u, v) belongs to is not adopted by session i, we also have $h^i_{uv} \equiv 0$. As a result, we have the following relationship:

$$h^i_{uv} \leqslant y^l_i, \forall i \in I, l \in L_i, (u, v) \in l. \tag{9}$$

As shown in Figure 3, we can see that $h^i_{uv} = 1$ and $h^i_{vw} = 0$ if and only if v is selected to host the VM for session i. Consequently, for session $i \in I$, the following expression can be used to describe the relationship between $h^i_{uv}, \forall u, v \in S$, and $x^{il}_v, \forall u \in S$:

$$\begin{aligned} &h^i_{uv} - h^i_{vw} = x^{il}_v, \\ &\forall (u, v), (v, w) \in l, l \in L_i, v \neq s', v \neq d'. \end{aligned} \tag{10}$$

Furthermore, the total traffic for all sessions in each link must be limited within its capacity; that is, based on the previous text, the link capacity constraints can be expressed as

$$\sum_{i \in I} T_i \cdot (1 - h^i_{uv}) \leqslant M_{uv}, \ \forall (u, v) \in E. \tag{11}$$

Network cost: If we migrate the VM to an RSC p_i on $l \in L_i$, the whole path is divided into two parts. The first part is used for VM migration while the second one is for bidirectional data communication between the vehicle and its corresponding VM. For a given path, it is straightforward to derive the cost (e.g., the number of hops) between s_i and p_i and p_i and d_i as $Dis(s_i, p_i)$ and $Dis(p_i, d_i)$, respectively. Therefore, the overall network cost for session $i \in I$ whose VM is placed at RSC p_i can be calculated as

$$Dis(s_i, p_i) \times C_i + Dis(p_i, d_i) \times T_i. \tag{12}$$

The total network cost for all sessions therefore can be calculated as

$$\psi = \sum_{i \in I} \sum_{\forall l \in L_i} y^l_i \left[C_i \sum_{\forall (u,v) \in l} h^i_{u,v} + T_i \sum_{\forall (u,v) \in l} \left(1 - h^i_{u,v}\right) \right] \tag{13}$$

Problem formulation: By summarizing all the previous constraints together, we obtain an MIQP as

$$\begin{aligned} &\min_{L_i, h^i_{uv}} \ : \ \psi \\ &\text{s.t.} : \ (2), (4), (5), (6), (8), (9), (10) \text{ and } (11). \end{aligned} \tag{14}$$

5. HEURISTIC ALGORITHM

It is computation inhibitive to obtain the optimal solution by solving the formulated MIQP when there are many sessions. To address this problem, in this section, we propose a polynomial heuristic algorithm. Our idea comes from the intuition that the shorter the distance of a session between s and d, the less aggregated communication cost would be. In addition, the cost of VM migration and execution directly affects the VM placement on the selected path. Therefore, we design a link-constrained and resource-constrained *two-phase* heuristic, which includes a shortest path-finding algorithm based on link capacity constrained and a cost weighted-based VM placement algorithm, which are summarized in Algorithms 1 and 2, respectively. Our main idea is to find the shortest path without violating the link capacity constraints (11) for each given session $i \in I$ from Algorithm 1. Then, we adopt Algorithm 2 to obtain the best placement node based on resources constrained from the selected path.

In Algorithm 1, we first initialize the output path for a session $i \in I$ as an empty set that includes nodes id from s to d. Some intermediate variables (i.e., the weight for each node to destination node, C, and minimum-weight node id, min_id) are also initialized in line 2. Next, we adopt Dijsktra algorithm based on link capacity constrained (M_{uv}) to find a suitable path. We first set the source RSC node by s as current node in line 3. Then, we update the weight of current node ($C[u]$) to its neighbor nodes by checking the link capacity between u and v (lines 5–12). After finding out all

DOI: 10.1002/cpe

Algorithm 1 Link capacity-shortest path-finding algorithm

Input: network graph $G = (S, E)$ with weighted $\mathbf{W}$ ($w_{uv} = 1$), link capacity M_{uv}, the traffic rate t, and the s_i and d_i for a VM session $i \in I$.
Output: The selected path P of a VM session i

1: $P \leftarrow \{0\}$
2: Initialize the weight (distance) for each node to destination node and minimum-weight node id, i.e., $C \leftarrow \{infinite\}, min\_id \leftarrow 0$
3: $CurRSC \leftarrow get\_rsc\_via\_id(s)$
4: **while** $true$ **do**
5: **for all** $AnyRSC \in CurRSC's\ neighbor$ **do**
6: $u \leftarrow CurRSC'id, v \leftarrow AnyRSC'id$
7: **if** $check\_capacity\_this\_link\_satisfy\_this(t, u, v) < M_{uv}$ **then**
8: **if** $C[u] > C[u] + w_{uv}$ **then**
9: $C[u] = C[u] + w_{uv}, P \leftarrow \{u\}$
10: **end if**
11: **end if**
12: **end for**
13: **for all** $v \in V[G]$ **do**
14: Find minimum value from $C[v]$
15: $min\_id \leftarrow v$
16: **end for**
17: **if** $min\_id == d$ **then**
18: $update\_links\_capacity(t, P)$
19: return P
20: **end if**
21: **end while**

Algorithm 2 Cost weighted-based VM placement algorithm

Input: the selected paths set PS of all VM sessions I, the migration cost set C, the execution cost set T, and the resources set R
Output: the VMs placement set $\mathbf{X}^{VL}$

1: **for all** $p_i \in PS,\ \forall i = 1, 2, \cdots, \alpha$ **do**
2: $\eta = \frac{C_i}{C_i + T_i}, \xi = \frac{T_i}{C_i + T_i}$
3: **if** $\eta \geq \xi$ **then**
4: *find* j *from front of* p_i
5: **else**
6: *find* j *from behind of* p_i
7: **end if**
8: **if** $check\_rsc\_res\_satisfy\_this(Q_{ij}, A_{jr_i})$ **then**
9: $x_{ij}^{VL} \leftarrow 1$
10: *break*
11: **end if**
12: **end for**
13: $update\_rsc\_resource(\mathbf{X^{VL}})$

updated nodes, we select the node with minimum-weight as new current node, and repeat the process until min_id equal d (lines 13–16). We then can finalize P and update link capacity according to P from lines 17 to 20.

By Algorithm 1, we can obtain the suitable path set P for each session $i \in I$. All the suitable paths form path set PS. Based on PS, we next try to obtain the VM placement solution using Algorithm 2. Specially, we define two variables η and ξ to represent the weight of C_i and T_i for

DOI: 10.1002/cpe

each session $i \in I$, respectively in line 2. Then, we perform two policies to find a suitable RSC node with enough capacity to place VMs (lines 3–7). The basic idea is inspired by an intuitional fact that we shall avoid VM migration if the migration cost to be incurred is larger than the actual execution cost (i.e., $\eta \geq \xi$), so we should find the placement node starting from source node for a given session $i \in I$ (line 4), and vice versa (line 6). After finding the suitable node, we need to check if the VM's resource requirement violates the RSC node's resource capacity; if not, we obtain the placement node (lines 8–11).

Remark: The overall time complexity of our *two-phase* algorithm is $O(MN^2)$ if there are M sessions and N RSC nodes. Note that the dominant part of the *two-phase* algorithm to the computation complexity is the shortest path-finding algorithm in Algorithm 1. Up to N iterations are required to find a suitable path set by checking the neighbor nodes between s and d for a session $i \in I$. Then, only a few nodes in a path will be checked in Algorithm 2 to find out an appropriate node with sufficient resources to hold the VMs to be placed. Therefore, in the worst case, the time complexity of the *two-phase* algorithm is $O(MN^2)$.

6. PERFORMANCE EVALUATION

In this section, we conduct simulations to evaluate the performance of the proposed *two-phase* heuristic algorithm in different topologies. The optimal solutions are solved using commercial solver Gurobi optimizer [30] for small-scale network with few RSCs and vehicles. To choose the competitors of our algorithm, we notice that Xu *et al.* [31] recently formulated the VM placement in data center as a pin-packing problem and proposed two VM placement algorithms, that is, *best-fit decreasing* and *first-fit decreasing* algorithms. In order to make the proposed algorithms suitable for VM placement in RSC-based VCC, we modify the previous two algorithms to *best-fit-like* and *first-fit-like*, as competitors, with the following concept. In each time slot, we consider RSC nodes as bins between s_i and d_i for session $i \in I$ and then increasingly sort all bins by the length of path (the number of nodes). The two competitors are briefly summarized as follows.

- *first-fit-like (FFL)*: sort bins on each path by their index value from s_i to d_i, and select the first available bin that has enough resources capacity to host the VM.
- *best-fit-like (BFL)*: sort the bins on each path by their residual resources capacity increasingly, and select the first available bin that has enough resources capacity to host the VM.

Furthermore, in order to validate comprehensively the performance of VM migration problem, we consider two straightforward algorithms. The first one is that a VM is always migrated with its vehicle, called *VMs-follow-vehicles (VFV)*, and another one is that persistently keeping a VM in its original position, called *VMs-never-migrate* (VNM).

Other settings such as link capacity, RSC resource capacity, VM resource demand, and image size are also randomly generated for a given range. Both the positions of both vehicles and RSCs are also randomly generated following Poisson process [32]. Without loss of generality, we normalize the capacity of all links in network, that is, M_{uv}, into a common value (2000) in all simulations. The main simulation settings are summarized in Table II. We evaluate the performance of our *two-phase* heuristic algorithm under several groups of simulations. The evaluation results are reported as follows.

Table II. A part of the environment configuration.

Settings	Range	Unit
Link capacity	[2000, 2000]	Bytes
RSC resource capacity	[1000, 2000]	MIPS
VM resource demand	[80, 300]	MIPS
VM image size	[50, 200]	Bytes
Session traffic rate	[100, 200]	Bytes

RSC, roadside cloudlet; VM, virtual machine; MIPS, million instructions per second.

DOI: 10.1002/cpe

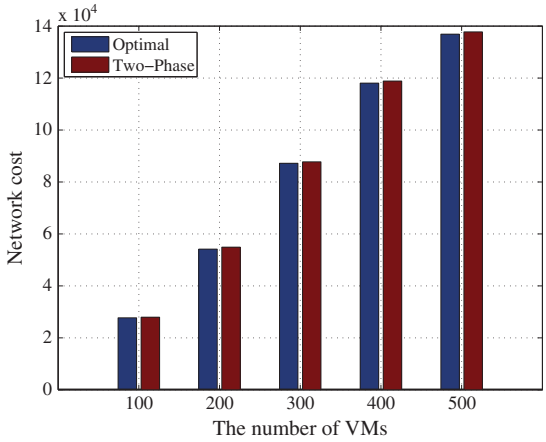
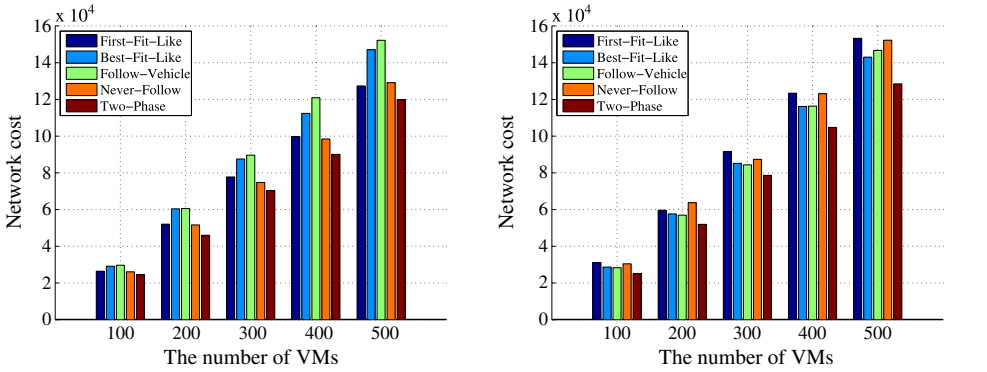


Figure 4. Performance comparison between optimal and two-phase.



(a) Network cost on different number of sessions when $C_i > T_i$ (b) Network cost on different number of sessions when $C_i > T_i$

Figure 5. (a) and (b) Network cost on different number of sessions. VM, virtual machine.

6.1. *On different number of sessions*

Let us first check how our 'two-phase' algorithm compared with the optimal solution obtained by Gurobi. The results are shown in Figure 4, from which we can see that the performance of our proposed algorithm quite approaches the optimal solution on different numbers of sessions. Besides, we also compare our algorithm with other heuristics and show the results in Figure 5. Obviously, for either algorithm, the overall network cost shows as an increasing function of the number of sessions. This is due to the fact that more sessions imply more network traffics and hence the network cost. Furthermore, from Figure 5(a), we can also see that *FFL* performs better than *BFL*, and *VNM* is better than *VFV*. This is because migration cost is greater than execution cost ($C_i > T_i$) for all VMs in current settings. Conversely, we make another experiment that the execution cost is greater than migration cost ($T_i > C_i$) showing in Figure 5(b). We can see that it exhibits different phenomena with Figure 5(a) that *FFL* performs worse than *BFL*, and *VNM* is worse than *VFV*. Nevertheless, our 'two-phase' is always with the lowest network cost in any case. We can conclude that both greedy VM migration along with vehicle moving and persistently keeping VM in its initial location are not efficient. The migration cost and execution cost shall be well balanced via appropriately migrating the VMs.

6.2. *On different requirements of resources and traffic rates*

Besides the performance evaluations on the number of sessions, we are also interested in the performance evaluation on different requirements of resources and traffic rates of the sessions. In this

DOI: 10.1002/cpe

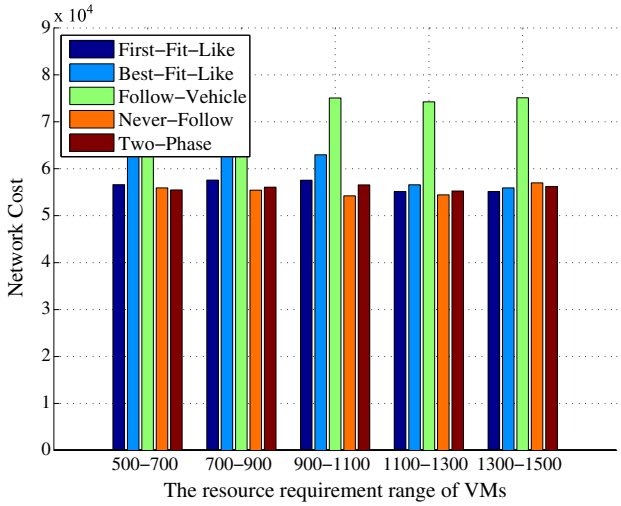


Figure 6. Network cost on different resource requirements of virtual machines (VMs).

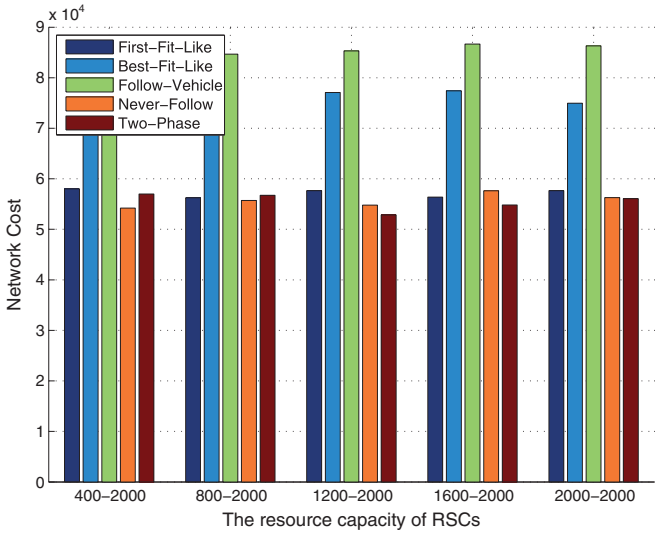


Figure 7. Network cost on different roadside cloudlet (RSC) resource capacities.

group of experiments, we change the number of RSCs to 300 (i.e., $|S| = 300$) and set the migration cost larger than the execution cost (i.e., $C_i > T_i$). Firstly, we compare our algorithm with others on different resource requirements of VMs in Figure 6. We can always see the advantage of our proposed two-phase algorithm in all settings. In addition, when $C_i > T_i$, we can see that *BFL* and *VFV* are relatively worse. We also compare our algorithm with others on different RSC resource capacities in Figure 7. Similar phenomenon can be also observed.

Moreover, we also evaluate the performance of our algorithm on different traffic rates of sessions by comparing it with other algorithms. The performance evaluation results are reported in Figure 8. Because of the increasing demand of session traffic, the quality of service degrades as the capacity-limited communication link cannot satisfy all sessions. Nevertheless, our algorithm always has an obvious advantage over other algorithms, under any session traffic rate.

DOI: 10.1002/cpe

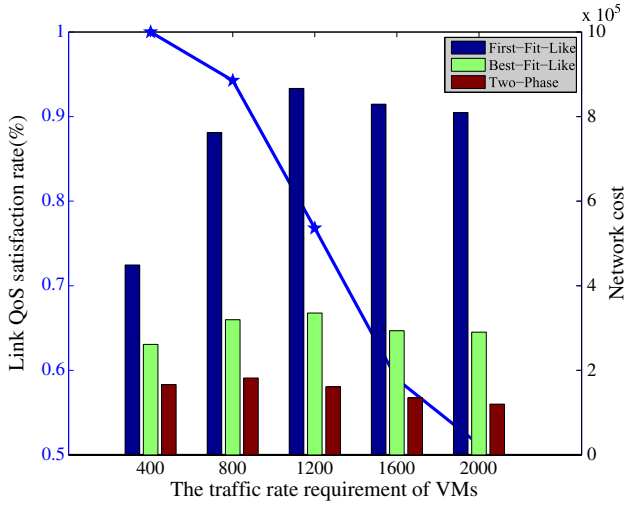


Figure 8. Network cost on different traffic rates. VM, virtual machine.

7. CONCLUSION

Vehicular cloud emerges as a promising platform to promote the driving experience. In this paper, we first present the roadside cloudlet-based vehicular cloud architecture. Then, we study a critical problem of how we shall migrate the VMs along with the vehicles during moving so as to reduce the overall network cost, with the consideration of VM migration cost and normal vehicle-to-VM communication cost. We formulate the problem as an MIQP problem and propose a polynomial heuristic algorithm *two-phase*. Its efficiency is extensively validated in simulation-based studies by the fact that it much approaches the optimal solution while with much advantage over other heuristic algorithms.

ACKNOWLEDGEMENTS

This research was supported by the NSF of China (grant nos. 61305087, 61402425, 61502439, 61272470, 41404076, and 61440060), the China Postdoctoral Science Foundation funded project (grant no. 2014M562086), the Fundamental Research Funds for National University, China University of Geosciences, Wuhan (grant nos. CUG14065 and CUGL150829), and the Provincial Natural Science Foundation of Hubei (grant no. 2015CFA065).

REFERENCES

1. Miche M, Bohnert TM. The Internet of vehicles or the second generation of telematic services. *ERCIM News* 2009; **77**:43–45.
2. Wan J, Zhang D, Zhao S, Yang LT, Lloret J. Context-aware vehicular cyber-physical systems with cloud support: architecture, challenges, and solutions. *Communications Magazine, IEEE* 2014; **52**(8):106–113.
3. Abid H, Phuong LTT, Wang J, Lee S, Qaisar S. V-cloud: vehicular cyber-physical systems and cloud computing. In *Proceedings of the 4th International Symposium on Applied Sciences in Biomedical and Communication Technologies*. ACM: New York, NY, USA, 2011; 165.
4. Yan H, Wan J, Wang Y, Wang Z, Song Z. Vehicular cyber-physical systems with mobile cloud computing support. In *Cloud Computing*. Springer: Berlin, Germany, 2014; 27–35.
5. Yu R, Zhang Y, Gjessing S, Xia W, Yang K. Toward cloud-based vehicular networks with efficient resource management. *IEEE Network* 2013; **27**(5):48–55.
6. Li Y, Wang W. Can mobile cloudlets support mobile applications? In *2014 Proceedings IEEE INFOCOM*. IEEE: New York, NY, USA, 2014; 1060–1068.
7. Satyanarayanan M, Bahl P, Caceres R, Davies N. The case for VM-based cloudlets in mobile computing. *IEEE Pervasive Computing* 2009; **8**(4):14–23.
8. Vaquero LM, Rodero-Merino L. Finding your way in the fog: towards a comprehensive definition of fog computing. *ACM SIGCOMM Computer Communication Review* 2014; **44**(5):27–32.

DOI: 10.1002/cpe

9. Bonomi F, Milito R, Zhu J, Addepalli S. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*. ACM: New York, NY, USA, 2012; 13–16.
10. Bonomi F, Milito R, Natarajan P, Zhu J. Fog computing: a platform for internet of things and analytics. In *Big Data and Internet of Things: A Roadmap for Smart Environments*. Springer: Berlin, Germany, 2014; 169–186.
11. Kumar VA, Prasad E. Fog computing: characteristics, advantages and security-privacy.
12. Liu Z, Qu W, Liu W, Li K. Xen live migration with slowdown scheduling algorithm. In *2010 International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT),* IEEE: New York, NY, USA, 2010; 215–221.
13. Zhang X, Li K, Zhang Y. Minimum-cost virtual machine migration strategy in datacenter. *Concurrency and Computation: Practice and Experience* 2015. DOI: 10.1002/cpe.3554.
14. Gkatzikis L, Koutsopoulos I. Mobiles on cloud nine: efficient task migration policies for cloud computing systems. In *2014 IEEE 3rd International Conference on Cloud Networking (CLOUDNET),* IEEE: New York, NY, USA, 2014; 204–210.
15. Refaat TK, Kantarci B, Mouftah HT. Dynamic virtual machine migration in a vehicular cloud. In *2014 IEEE Symposium on Computers and communication (ISCC),* IEEE: New York, NY, USA, 2014; 1–6.
16. Mishra M, Das A, Kulkarni P, Sahoo A. Dynamic resource management using virtual machine migrations. *IEEE Communications Magazine* 2012; **50**(9):34–40.
17. Dinh HT, Lee C, Niyato D, Wang P. A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless Communications and Mobile Computing* 2013; **13**(18):1587–1611.
18. Sanaei Z, Abolfazli S, Gani A, Buyya R. Heterogeneity in mobile cloud computing: taxonomy and open challenges. *IEEE Communications Surveys & Tutorials* 2014; **16**(1):369–392.
19. Verbelen T, Simoens P, De Turck F, Dhoedt B. Cloudlets: bringing the cloud to the mobile user. In *Proceedings of the Third ACM Workshop on Mobile Cloud Computing and Services*. ACM: New York, NY, USA, 2012; 29–36.
20. Fesehaye D, Gao Y, Nahrstedt K, Wang G. Impact of cloudlets on interactive mobile cloud applications. In *2012 IEEE 16th International Enterprise Distributed Object Computing Conference (EDOC),* IEEE: New York, NY, USA, 2012; 123–132.
21. Eltoweissy M, Olariu S, Younis M. Towards autonomous vehicular clouds. In *Ad Hoc Networks*. Springer: Berlin, Germany, 2010; 1–16.
22. Bernstein D, Vidovic N, Modi S. A cloud PaaS for high scale, function, and velocity mobile applications—with reference application as the fully connected car. In *Proceedings of the 2010 Fifth International Conference on Systems and Networks Communications*. IEEE Computer Society: New York, NY, USA, 2010; 117–123.
23. Son J, Eun H, Oh H, Kim S, Hussain R. Rethinking vehicular communications: merging VANET with cloud computing. In *Proceedings of the 2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CLOUDCOM)*. IEEE Computer Society: New York, NY, USA, 2012; 606–609.
24. Zhang Qi, Cheng Lu, Boutaba R. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications* 2010; **1**(1):7–18.
25. Zhang J, Ren F, Lin C. Delay guaranteed live migration of virtual machines. In *2014 Proceedings IEEE INFOCOM,* IEEE: New York, NY, USA, 2014; 574–582.
26. Zhang X, Shae ZY, Zheng S, Jamjoom H. Virtual machine migration in an over-committed cloud. In *Network 2012 IEEE Operations and Management Symposium (NOMS),* IEEE: New York, NY, USA, 2012; 196–203.
27. Shrivastava V, Zerfos P, Lee KW, Jamjoom H, Liu YH, Banerjee S. Application-aware virtual machine migration in data centers. In *2011 Proceedings IEEE INFOCOM,* IEEE: New York, NY, USA, 2011; 66–70.
28. Li Xin, Wu Jie, Tang Shaojie, Lu Sanglu. Let's stay together: towards traffic aware virtual machine placement in data centers. In *2014 Proceedings IEEE INFOCOM,* IEEE: New York, NY, USA, 2014; 1842–1850.
29. Li K, Zheng H, Wu J. Migration-based virtual machine placement in cloud systems. In *2013 IEEE 2nd International Conference on Cloud Networking (CLOUDNET),* IEEE: New York, NY, USA, 2013; 83–90.
30. Gurobi Optimization. Gurobi optimizer reference manual. [Online], Available: http://www.gurobi.com, 2012.
31. Xu J, Fortes JA. Multi-objective virtual machine placement in virtualized data center environments. In *Green Computing and Communications (GREENCOM), 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCOM)*. IEEE: New York, NY, USA, 2010; 179–188.
32. Ibrahim M, Nain P, Carreras I. Analysis of relay protocols for throwbox-equipped DTNS. In *In 7th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WIOPT 2009),* IEEE: New York, NY, USA, 2009; 1–9.