A virtual machine migration policy for multi-tier application in cloud computing based on game theory

Hung Cong Tran
Posts and Telecommunications Institute of Technology
Khiet Thanh Bui (khietbt@tdmu.edu.vn)
Thu Dau Mot University https://orcid.org/0000-0002-1686-5055
Hung Dac Ho
Thu Dau Mot University
Vu Tran Vu
Ho Chi Minh City University of Technology

Research Article

Keywords: VM migration, Game theory, Cloud computing, Q-Learning

Posted Date: June 2nd, 2021

DOI: https://doi.org/10.21203/rs.3.rs-261767/v1

RESEARCH

A virtual machine migration policy for multi-tier application in cloud computing based on game theory

Cong Hung Tran[1], Thanh Khiet Bui[2,3,4*], Dac Hung Ho[4] and Tran Vu Pham[2,3]

Abstract

Cloud computing technology provides shared computing which can be accessed over the Internet. When cloud data centers are flooded by end-users, how to efficiently manage virtual machines to balance both economical cost and ensure QoS becomes a mandatory work to service providers. Virtual machine migration feature brings a plenty of benefits to stakeholders such as cost, energy, performance, stability, availability. However, stakeholder's objectives are usually conflicted with each other. Also, the optimal resource allocation problem in cloud infrastructure is usually NP-Hard or NP-Complete class. In this paper, the virtual migration problem is formulated by applying game theory to ensure both load balance and resource utilization. The virtual machine migration algorithm, named V2PQL, is proposed based on Markov Decision Process and Q-learning algorithm. The results of the simulation demonstrate the efficiency of our proposal which are divided into training phase and extraction phase. The proposed V2PQL policy has been benchmarked to the Round-Robin policy in order to highlight their strength and feasibility in policy extraction phase.

Keywords: VM migration; Game theory; Cloud computing; Q-Learning

1 Introduction

Cloud computing technology has made computing resources more and more powerful, abundant and cheaper based on the rapid development of processing power and storage, where multiple users can access to computing resources over the Internet in on-demand fashion [1]. Cloud computing resources can be adjusted based on user-on-demand mechanism to ensure quality of service (QoS) as well as profits. By using virtual technology, the physical machines (PMs) are accessed by customers in a multi-tenant manner. Virtualization technology allows to create multiple virtual machines (VMs) on a physical server (PM), and each VM is also allocated hardware resources like real machines with RAM, CPU, network card, hard drive, operating system, and other own applications. Virtualization resources are flexibly organized for the benefit of applications and software. To take advantage of distributed computing, cloud-based deployment applications are often developed based on service-oriented architecture which is deployed based on a cluster of unique services and communicated with each other by a flexible mechanism [2]. For example, multi-tier applications based on web services such as three-tier web applications include web server tier, application server tier, and database server tier; NoSQL applications are deployed based on Cassandra, Hbase, Infinispan and HDFS technologies; Unlike monolithic applications that incorporate tightly integrated modules, applications based on service architecture are well suited for cloud infrastructures. Therefore, resource mismanagement can lead a lot of problems to customers, their end-users and service level agreements (SLA) violations, energy wastage, increased costs, revenue loss, and so on.

VM migration is one of major advantages of virtualization to manage cloud computing resources. It allows migrating VMs to from one location to another which makes VMs free from the underlying hardware. A VM can be migrated from one PM to another PM while the VM is still running during migration [3]. It is a major advantage of Cloud computing which increases shared-resource utilization. Cloud system can archive load balance by migrating VMs from over-loaded PM to light-loaded ones. VMs running on light-load PM

*Correspondence: khietbt@tdmu.edu.vn
[1]Training and Science Technology Department, Posts and Telecommunications Institute of Technology, 11 Nguyen Dinh Chieu, Ho Chi Minh, Vietnam
Full list of author information is available at the end of the article

can be consolidated in another PM to minimize power consumption by reducing the amount of running PMs. Proactive fault tolerance model can be implemented by migrating VMs to another PMs to avoid expected faults before their occurrence in PMs [4]. Besides, the performance of cloud-based application can be increased by migrating some VMs from their limited resource PMs to rich resource ones. However, VM migration aims at different purposes of stakeholder's objectives including service providers, customers, end-users. The optimal resource utilization is essential in the efficient use of resources in large-scale of a cloud environment, the optimization problem of this type is usually of the NP-Hard or NP-Complete class [5]. The type of VM migration algorithms can be derived from distributed and parallel computing such as scheduling work for multiple processors, bin packing, graph partitioning algorithms. Many resource coordination algorithms have been developed, but none is suitable for all applications [6][7][8]. To solve these problems, usually, the exhaustive algorithms, deterministic algorithms or meta-heuristic algorithms [9][10][11] are applied by specific characteristics. In experiments, the deterministic algorithms are better than the exhaustive algorithms. However, the deterministic algorithms are inefficient in in large-scale environment [12]. Meanwhile, cloud services need to response customer as soon as possible to ensure QoS. In addition, not only cloud system but also cloud-hosted application become more complex in run-time because of elasticity characteristic and resource sharing paradigm. Therefore, it is rarely feasible to have the detailed prior knowledge on cloud system, cloud-hosted application and their interactive dynamics for managing resources effectively. Besides, the majority of physical resources in the cloud computing environment are not homogeneous as well as customers' resource demand. Heterogeneous resources can cause fragmentation of resources, resulting in a waste of resources. This issue requires new methods to coordinate resources in stochastic, complex, and heterogeneous systems with limited prior knowledge. These methods should be able to automatically produce effective resource management policies in run-time.

This paper focuses on VM migration solutions which aim the purposes of stakeholders including cloud service providers and customers. From perspective of cloud service providers, resource management helps maximize system utilization and reach high profit. To customers, resource coordination helps to ensure service level agreement(SLA). However, the maximum exploitation of resources will lead to the performance and QoS provided to customers will be difficult to satisfy. In the meantime, customers want to minimize using costs thereby leading to minimizing service time by requiring more resources. From there, it can be seen that the target relationship of cloud service providers and the customers may conflict with each other. Motivated by stakeholder goals, the VM migration problem considered in our work is based on maximizing the resource utilization while ensuring customers SLAs by balancing the load among PMs to avoid concurrency and congestion. The problem is modeled as non-cooperate game based on game theory. To deal with VM migration in run-time, a new approach of continuous learning in interaction which refers to as Reinforcement Learning (RL) should be applied to the dynamic cloud environment. With no prior knowledge about characteristics of cloud system, the cloud controller agent takes migration actions and learns on-the-fly about their efficiency through the observed feedback from the cloud infrastructure.

In view of this challenge, the VM migration algorithm is proposed by applying reinforcement learning method which ensures to balance the goals of stakeholders including service providers and customers in this paper. The VM migration problem for multi-tier applications is formalized in non-cooperative game theory for ensuring the goal of stakeholders. Based on Markov Decision Process (MDP)[13][14][15], the V2PQL algorithm is developed to trade-off the load balance and resource utilization in cloud infrastructure. The optimal policy of VM migration is searched in the training phase. The agents perform actions impacting the environment in order to maximize the total reward as a result of actions. At discrete moment of time, the agents observe the state of system and choose an action from the set of action impacting the environment. After the training phase, the optimal policy with Q-Value is used to migrate VMs to other PMs. Our main contributions of the study are as following.

- The VM migration for multi-tier application is modeled by using non-cooperative game theory to describe the conflict among cloud service provider and customers. In this game, the PMs are considered players of the game which take into the self-fish feature in the case of scarce resources [16][17]. Each player tries to maximize their own utility by changing their strategies which trade-offs the load balance and resource utilization.
- The VM migration algorithm, named V2PQL, is proposed by applying Q-Learning algorithm to solve VM migration game. Without any prior knowledge, the V2PQL algorithm tries to find an optimal VM migration policy based on interacting the agents and the environment. The optimal policy is described as Q-Table which includes states, actions, and q-values. The Q-Table is updated overtime by reinforcement learning mechanism.

- The heterogeneous data center which deploys one hundred multi-tier application is simulated. The optimal VM migration policy is investigated by the V2PQL algorithm in the training phase. And then,in VM migration game, the utility of V2PQL policy is benchmarked with Round-Robin policy in the extraction phase.

The outline of the paper is as follows. The related work is discussed in section 2. Section 3 presents the The VM migration game approach . The VM migration algorithm based on Q-Learning is described in section 4. Section 5 presents the evaluation of the proposed method and a discussion on the results. Finally, Section 6 concludes the work.

2 Related work

According to stakeholder requirements, the decision making process of selecting available resource's PM to deploy VM have different objectives [18][19].

PMs running with under-loaded state leads to waste of energy while overloaded state results shorten lifespan of PMs, and then reduces QoS. By migrating VMs, the loads can be balanced between among PMs in data-center to ensure QoS [20][21][22]. Massimo Ficco et al.[23] proposed a meta-heuristic approach for the allocation of cloud resources based on the model of biological-inspired coral ecology optimization. Based on the game theory, the optimize resource allocation strategies are searched to ensure the objectives of the service providers as well as the requirements of customers. The evolutionary algorithm is proposed based on observing the structure of coral reefs and spawning corals. It also exploits the dynamism of competition among users and service providers to satisfy the benefits of stakeholders. Experiments show that the combined method based on biological emotions and game theory not only achieves a satisfactory solution of adaptability and elasticity but can also lead to significant performance improvement. In [6], Bai et al. proposed a method to evaluate the performance of applications on cloud computing. By the analysis of the QoS metrics including average response time, average waiting time, the flow density (usage) of each PM is evaluated in a heterogeneous data center. A complex queue model of serial and parallel queuing systems is modeled to evaluate the performance of heterogeneous data centers.

Considering environmental and economic aspects, the energy-aware in cloud computing becomes a hot topic. In PM consolidation, VMs are migrated for using as fewer PMs as possible. The power cost of VM migration models are considered through the metric networks between the source PM and the destination PM. [24] considers load balancing of PMs which consists of a set of VMs described as a multi-dimensional vector. VMs are assigned to the smallest amount of PM within the power limit to achieve optimum load. VM allocation is modeled based on non-cooperative games. Moreover, the distribution of resource utilization is resolved with machine learning algorithm to achieve efficiency [25][26][27][28]. Dhanoa et al.[29] analyzed energy consumption during VM migration on VM size and network bandwidth. In [30], Rybina et al. introduced prediction of energy cost due to migration based on resource utilization of PM. By reaching more accurate prediction model, the better migration decisions are taken in data center power management. Therefore, VM migration with less time will help in minimizing power cost during the migration process. Algorithms to adjudicate which VMs to be migrated from each PM should be considered the phase of prediction of VM resource demand based on machine learning [31][32][33][34] to support the decision making.

The resource management in cloud environment is considered as a automatic control problem using the the reinforcement learning approach. Reinforcement learning is one of the machine learning methods in which the agents take actions impacting the environment to minimize the total amount of penalties from result of each action[35]. In [36], the authors proposed a unified reinforcement learning approach to the VM and application configuration processes. VM resource needs the changing workload which is adapted to provide service quality assurance. However, the proposed approach does not account the need of VM migrations. Farahnakian et al. [37] proposed a dynamic consolidation method based on reinforcement learning to minimize the number of active hosts according to the current resource requirements. The host power mode can be determined by an agent. A decision about host power mode from collected data is taken by the agent learns and is improved itself as the workload changes dynamically. However, the proposed algorithm focuses on only CPU performance and does not discuss other host resources. In [38], the authors analyze the possibility of application to cloud data center resource management based on reinforcement learning method. The proposed method deals with the power consumption and the number of SLA violations by using Q-Learning approach.

3 Virtual machine migration game approach

In this section, the VM migration problem is modeled by non-cooperative game with PMs as players. Each player tries to maximize the utility which trades off

the load balance and resource utilization. The VM migration game approach is proofed that exist the Nash equilibrium.

3.1 VM migration modeling

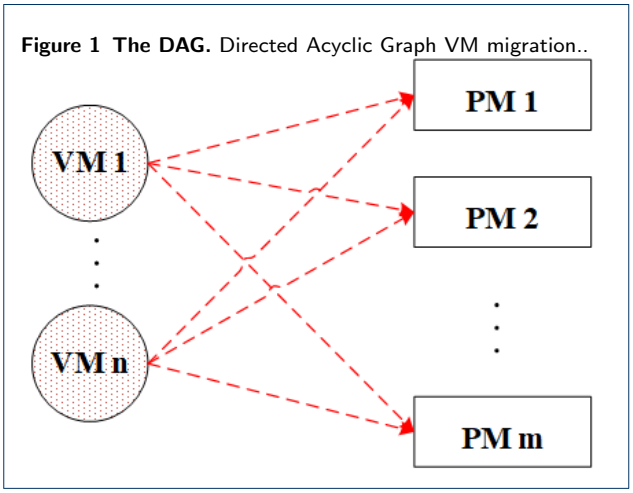


Figure 1 The DAG. Directed Acyclic Graph VM migration..

Supposing the cloud infrastructure has a large scaled heterogeneous physical machine and provides the computational resource as on demand model. There are a lot of physical machines deploying VMs based on virtualization technology. Cloud providers offer a group of VM types to remove the complexities of selection for customers, and each type is specifically determined as the number of CPU, the memory size, the storage size. Depending on the needs of the users, the resource allocation decision of providers have to adjust dynamically. The multi-tier applications are hosted in VMs cluster.

As shown in Fig. 1, it is possible to model the VM migration problem on the cloud in the form of a directed acyclic graph (DAG)[39][40] G(V, E) where V is the set of vertices representing the work, E is a set of edges that show the dependency relationship between vertices. VM migration process is trigged when the deteriorating PM is detected. At the moment, cloud infrastructure has n VMs which need to be migrated to m safe optimal PMs.

Definition 1 *(migration decision). A possible resource allocation for migrating n VMs to m PMs can be described as a binary matrix* $\mathbf{X}(n \times m)$*:*

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{pmatrix} \tag{1}$$

where $x_{nm} = \{0, 1\}$*, the migration VM n to PM m is described* $x_{nm} = 1$*, otherwise* $x_{nm} = 0$*.*

Definition 2 *(allocation decision). Based on definition 1, a possible for k kind of resource allocation in the PM* i^{th} *can be described as allocation matrix* $\mathcal{M}^{(i)}(n \times k)$*:*

$$\mathcal{M}^{(i)} = \begin{pmatrix} v_{11}^{(i)} & v_{12^{(i)}} & \cdots & v_{1k}^{(i)} \\ v_{21}^{(i)} & v_{22^{(i)}} & \cdots & v_{2k}^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n1}^{(i)} & v_{n2}^{(i)} & \cdots & v_{nk}^{(i)} \end{pmatrix} \tag{2}$$

where $v_{nk}^{(i)} \in \mathbb{Z}_+$ *shows the amount of resource type k of* i^{th} *PM provided to VM n.*

A vector $\mathcal{M} = \{\mathcal{M}^{(1)}, \mathcal{M}^{(2)}, \ldots, \mathcal{M}^{(m)}\}$ defines a possible resource allocation strategy for all PMs. An optimal VM migration problem can be described as following the trade-off between load balance and resource utilization based on non-cooperate game theory.

3.1.1 Load balance

Let $\sigma^{(i)}$ denotes the resource usage of the i^{th} PM which is measured as following:

$$\sigma^{(i)} = \sum_{j=1}^{k} \lambda_j \frac{u_j^{(i)}}{c_j^{(i)}} \tag{3}$$

where λ_j coefficient shows the influence o resource type j with $\sum_{j=1}^{k} \lambda_j = 1$, $u_j^{(i)}$ denotes the usage of resource type j in i^{th} PM, and $c_j^{(i)}$ is the capacity of resource type j in i^{th} PM. The load balance of the system is calculated by the following formula:

$$\mathcal{L} = \frac{\sum_{i=1}^{m} (\sigma_i - \bar{\sigma})^2}{m - 1} \tag{4}$$

where $\bar{\sigma}$ is the average value of the performance of PMs in the cloud computing infrastructure.

3.1.2 Resource utilization

For service providers, in order to achieve high profits, the resource of PMs need to avoid wastage resources of PMs. The concept of skewness in [41] is applied to quantify the unevenness of the utilization of different resources on PM p which is calculated as following:

$$\mathcal{H}^{(i)} = \sqrt{\sum_{j}^{k} \left(\frac{u_j^{(i)}}{\bar{u}^{(i)}} - 1 \right)^2} \tag{5}$$

where $\bar{u}^{(i)}$ is the average utilization of all resource for i^{th} PM.

3.2 VM migration game approach

In this section, a game theory approach to VM migration is presented, aiming at keeping a load balance as well as avoiding wastage resources of PMs. Game theory is a mathematical study of strategy in which the interactions among all game players are ensured their best outcomes [16]. The VM migration problem is modeled on non-cooperative game in which the safe PMs are as players.

Definition 3 *(VM migration game model). A VM migration game is described as a three-tuple vector* $G = (\mathcal{P}, (\mathcal{M}^{(i)})_{i\in\mathcal{P}}, (f^{(i)})_{i\in\mathcal{P}})$.

(i) $\mathcal{P}$ *is the finite set of players in the game ,i.e,* $\mathcal{P} = \{1, 2, ..., m\}$.

(ii) $\mathcal{M}^{(i)}$ *is the set of available strategies for player* i.

(iii) $f^{(i)} = \mathcal{M} \to \mathbb{R}$ *is the utility function for player* i .

In this study, one global objective of this migration game is to balance th load and each individual player tries to minimize their resource wastage. To exploit load balance and also get the maximization of resource utilization, the utility function of i^{th} player is designed as following:

$$f^{(i)}(\boldsymbol{\mathcal{M}}) = \frac{1}{\mathcal{H}^{(i)} + \mathcal{L}} \tag{6}$$

The game's utility function has an important influence on a player's strategic decision and the game's outcome. Each player tries to maximize their own utility by adjusting their strategies, which is described as following:

$$\max_{\mathcal{M}} f^{(i)} \tag{7}$$

$$\text{subject to } \sum_{x=1}^{m}\sum_{y=1}^{k} v_{xy} \le c_y^{(i)} \tag{8}$$

$$\mathbf{X}^T\mathbf{1} = \mathbf{1} \tag{9}$$

where the constraint (8) ensures the provided resources of i^{th} PM not to exceed its capacity, and the constraint (9) ensures a VM migrated to one and only one PM.

The Nash equilibrium of the game is a state in which no player can increase its utility by changing its strategy while the other players have fixed their strategies. In other words, the Nash equilibrium is considered as a set of strategies where players have no motivation to change their actions. For any player i, every element $\beta^{(i)} \in \mathcal{M}^{(i)}$ is the strategy for player i, $\boldsymbol{\beta}^{(-i)} = [\beta^{(j)}]_{j\in\mathcal{P}, j\neq i}$ describes the strategies of all player except i, and $\beta = (\beta^{(i)}, \boldsymbol{\beta}^{(-i)})$ is referred to as a strategy profile.

Definition 4 *(Nash equilibrium). A profile* β^* *is a Nash equilibrium of* G *if and only if every player's strategy is a best response to the other players' strategies, that is,*

$$\beta^{(i)*} \in br^{(i)}(\boldsymbol{\beta}^{(-i)*}) \text{ for every player } i \tag{10}$$

where $\boldsymbol{\beta}^{(-i)}$ *is the strategies of all player except* i, $br^{(i)}$ *is the best response of player* i, *and* $br^{(i)}(\boldsymbol{\beta}^{(-i)*}) = \beta^{(i)*} \in \boldsymbol{\beta} | f^{(i)}(\beta^{(i)*}, \boldsymbol{\beta}^{(-i)}) \ge f^{(i)}(\beta^{(i)}, \boldsymbol{\beta}^{(-i)})$.

By defining the set value function $br : \boldsymbol{\beta} \to \boldsymbol{\beta}$ by $br(\beta^{(i)}) = \times_{i\in\mathcal{P}} br(\boldsymbol{\beta}^{(-i)})$, the Eq. (10) can be rewritten in vector form as $\beta^* \in br(\beta^*)$. The existence of β^* for which $\beta^* \in br(\beta^*)$ is proved by using Fixed point theorems.

Lemma 1 *(Kakutani's fixed point theorem) Let* $\boldsymbol{\beta}$ *be a compact convex subset of* $\mathbb{R}^n$ *and* $br : \boldsymbol{\beta} \to \boldsymbol{\beta}$ *be a set-value function such that for all* $\beta \in \boldsymbol{\beta}$ *the set* $br(\beta)$ *is nonempty and convex, and graph of* br *is closed. Then there exists* $\beta^* \in \boldsymbol{\beta}$ *such that* $\beta^* \in br(\beta^*)$.

Theorem 1 *The VM migration game* G *always has at least one strategy Nash equilibrium.*

Proof Using Lemma 1 $\forall i \in \mathcal{P}$, we have

i) $\boldsymbol{\beta}$ is compact, convex, and non-empty.

We have $\beta^{(i)} = \{v_{nk}^{(i)} \in \mathbb{Z}_+ \mid 0 \le v_{nk}^{(i)} \le c_j^{(i)}\}$ is closed, bounded, thus compact convex. Their product set $\boldsymbol{\beta}$ is also compact.

ii) $br(\beta)$ is non-empty.

By definition $br^{(i)}(\boldsymbol{\beta}^{(-i)}) = \underset{\beta^{(i)}}{\text{argmax}} f^{(i)}(\beta^{(i)}, \boldsymbol{\beta}^{(-i)})$ where $\beta^{(i)}$ is non-empty and compact, and $f^{(i)}(\beta^{(i)}, \boldsymbol{\beta}^{(-i)})$ is linear in $\beta^{(i)}$. Hence, $f^{(i)}(\beta^{(i)}, \boldsymbol{\beta}^{(-i)})$ is a continuous function in β, and by Weirstrsaa's theorem $br(\beta)$ is non-empty.

iii) $br(\beta)$ is a convex-valued correspondence.

Equivalently, $br(\beta) \subset \boldsymbol{\beta}$ is convex if only if $br^{(i)}(\boldsymbol{\beta}^{(-i)})$ is convex for all i.

Let $\beta^{(i)\prime}, \beta^{(i)\prime\prime} \in br^{(i)}(\boldsymbol{\beta}^{(-i)})$. Then, for all $\lambda \in [0, 1]$, we have $f^{(i)}(\beta^{(i)\prime}, \boldsymbol{\beta}^{(-i)}) \ge f^{(i)}(a_i, \boldsymbol{\beta}^{(-i)})$ for all $a_i \in \beta^{(i)}$, $f^{(i)}(\beta^{(i)\prime\prime}, \boldsymbol{\beta}^{(-i)}) \ge f^{(i)}(a_i, \boldsymbol{\beta}^{(-i)})$ for all $a_i \in \beta^{(i)}$. The preceding relations imply that for all $\lambda \in [0, 1]$, we have $\lambda f^{(i)}(\beta^{(i)\prime}, \boldsymbol{\beta}^{(-i)}) + (1-\lambda) f^{(i)}(\beta^{(i)\prime\prime}, \boldsymbol{\beta}^{(-i)}) \ge f^{(i)}(a_i, \boldsymbol{\beta}^{(-i)})$ for all $a_i \in \beta^{(i)}$. By the linearity of $f^{(i)}$, $f^{(i)}(\lambda\beta^{(i)\prime} + (1-\lambda)\beta^{(i)\prime\prime}, \boldsymbol{\beta}^{(-i)}) \ge f^{(i)}(a_i, \boldsymbol{\beta}^{(-i)})$ for all $a_i \in \beta^{(i)}$. Therefore, $\lambda\beta^{(i)\prime} + (1-\lambda)\beta^{(i)\prime\prime} \in br^{(i)}(\boldsymbol{\beta}^{(-i)})$,

showing that $br(\beta)$ is convex-valued.
iv) $br(\beta)$ has a closed graph.
Supposing that $br(\beta)$ does not have a closed graph. Then, there exists a sequence $(\beta^n, \hat{\beta}^n) \to (\beta, \hat{\beta})$ with $\hat{\beta}^n \in br(\beta^n)$ but $\hat{\beta} \notin br(\beta)$, i.e, there some i such that $\hat{\beta}^{(i)} \notin br^{(i)}(\boldsymbol{\beta}^{(-i)})$. This implies that there exists some $\beta^{(i)\prime} \in \beta^{(i)}$ and some $\epsilon > 0$ such that $f^{(i)}(\beta^{(i)\prime}, \boldsymbol{\beta}^{(-i)}) \geq f^{(i)}(\hat{\beta}^{(i)}, \boldsymbol{\beta}^{(-i)}) + 3\epsilon$. By the continuity of $f^{(i)}$ and the fact that $\beta^{(i)n} \to \boldsymbol{\beta}^{(-i)}$, we have for sufficiently large n, $f^{(i)}(\beta^{(i)\prime}, \boldsymbol{\beta}^{(-i)n}) \geq f^{(i)}(\beta^{(i)\prime}, \boldsymbol{\beta}^{(-i)}) - \epsilon$. Combining the preceding two relations, we obtain $f^{(i)}(\beta^{(i)\prime}, \boldsymbol{\beta}^{(-i)n}) > f^{(i)}(\hat{\beta}^{(i)}, \boldsymbol{\beta}^{(-i)}) + 2\epsilon \geq f^{(i)}(\hat{\beta}^{(i)n}, \boldsymbol{\beta}^{(-i)n}) + \epsilon$, where the second relation follows from the continuity of $f^{(i)}$. This contradicts the assumption that $\hat{\beta}^{(i)n} \in br^{(i)}(\boldsymbol{\beta}^{(-i)n})$, and completes the proof. □

4 VM migration algorithm based on Q-Learning

In this section, the MDP model includes the state and action spaces, transition probabilities, and reward structure which are completely specified. However, transition probabilities are often unknown for real-work setting. Also, the state and action spaces are often too large for algorithms to handle [42]. To solve this problem, the V2PQL algorithm is applied to find a Nash equilibrium of migration strategy by influencing the observed system states and rewards. Furthermore, the algorithm dose not require the prior knowledge of model parameters.

4.1 MDP framework for VM migration

A discrete-time MDP model is applied to build the optimal VM migration algorithm. Considering the process of migrating a VM to a safe PM is a stochastic process with assuming VM migration request arrivals independently. At each small separate time step, there is either one arrives exactly or no VM migration request. Also, these migration events occur with some given probability. Furthermore, the probability of VM migration requests a given type following with a predefined distribution. Given a sufficiently small a discrete-time step, a good approximation to the Poisson process is provided by this simple stochastic process.

To narrow down the system state space, fuzzy logic method are applied to the value of load lalancing in Eq.(4), utilization resource in Eq. (6). The Fig. 2 (a) depicts the membership functions of load balance level including three states, i.e., *Good*, *Normal*, and *Bad*, which is calculated as following:

$$\mu_{Good}(x) \begin{cases} 1 & \text{if } x < r \\ (s-x)/(s-r) & \text{if } r \leq x \leq s \\ 0 & \text{if } x > s \end{cases} \quad (11)$$

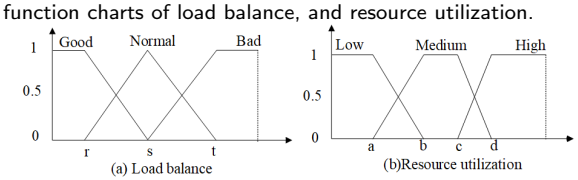


Figure 2 The membership function. The membership function charts of load balance, and resource utilization.

$$\mu_{Normal}(x) \begin{cases} 0 & \text{if } x < r \text{ or } x > t \\ (s-x)/(s-r) & \text{if } r \leq x \leq s \\ 1 & \text{if } x = s \\ (t-x)/(t-r) & \text{if } s \leq x \leq t \end{cases} \quad (12)$$

$$\mu_{Bad}(x) \begin{cases} 1 & \text{if } x > t \\ (t-x)/(t-s) & \text{if } s \leq x \leq t \\ 0 & \text{if } x < s \end{cases} \quad (13)$$

Fig 2(b) shows the membership functions of resource utilization level including three states, i.e., *Low*, *Medium*, and *High*, which is calculated as following:

$$\mu_{SLow}(x) = \begin{cases} 1 & \text{if } x < r \\ (b-x)/(b-a) & \text{if } a \leq x \leq b \\ 0 & \text{if } x > b \end{cases} \quad (14)$$

$$\mu_{Medium}(x) \begin{cases} 0 & \text{if } x < a \text{ or } x > d \\ (b-x)/(b-a) & \text{if } a \leq x < b \\ 1 & \text{if } b < x < c \\ (d-x)/(d-c) & \text{if } c < x \leq d \end{cases} \quad (15)$$

$$\mu_{High}(x) = \begin{cases} 1 & \text{if } x > d \\ (d-x)/(d-c) & \text{if } c < x \leq d \\ 0 & \text{if } x < c \end{cases} \quad (16)$$

Definition 5 *(Transition probabilities). The set of MDP states S at time t as a three-tuple of load balance state, resource utilization state, and the type of migrated VM, i.e., $s_t = (\mathcal{L}[t], \mathcal{H}[t], \vartheta[t])$. The action of migrating VM j to safe PM i corresponds to changing x_{ji} in Eq. (1) from 0 to 1 and adding the type of*

migrated VM to vector $\vartheta[t]$. The transition probability matrix $P(s'|s,a)$ can be analytically derived for a stochastic model.

Definition 6 *(Reward structure). The optimization problem (7)-(9) described the benefit of the current VM migration showing the system state snapshot. The reward $\mathcal{R}(s,a)$ of VM migration MDP can be defined as using the object function (7):*

$$\mathcal{R}(s,a) = \frac{1}{\mathcal{H}^{(i)} + \mathcal{L}} \tag{17}$$

The optimal MDP policy is a mapping from a MDP states S to a set of actions A based on maximizing the average reward of discounted cumulative reward over time. The reward function can serve as a basic element to change the policy. By using the modified reward function architecture, algorithms like Value Iteration (VI) or Policy Iteration (PI) calculate optimal policies. For example, in the Value Iteration algorithm, set $V(s_0)$ as the initialization value and update $V(s)$ iteratively until $V(s_t) \approx V(s_{t+1})$ according to the equation Bellman:

$$V(s_{t+1}) = R(s_t,a) + \alpha \max_k \sum_{s'} P(s'|s,a)V(s_t) \tag{18}$$

where $\alpha < 1$ represents the discounted value and n iterations. The optimal policy is then seen as $\underset{a}{\text{argmax}} V^*(s)$ where $V^*(s)$ are convergent values from equation (18).

Definition 7 *(Policy). Policy $\Pi(s,a)$ is the probability of selecting action a from state s, calculated using the following formula:*

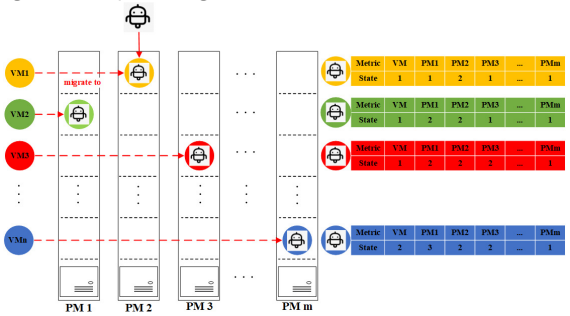
$$Q^{\Pi} = E_{\Pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}\right\} \tag{19}$$

where E_{Π}. is the expected function of policy Π, $\mathcal{R}_t = r_{t+1} + \gamma r_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$, γ is a coefficient that denotes the importance of future reward value.

4.2 VM migration algorithm

To find the VM migration strategies, a model-free version of the learning agent is proposed by applying Q-Learning algorithm [35]. The VM migration decision can be generated close to optimal by interacting with the environment without any prior knowledge. The Q-learning model is presented by a set S of environment states that learning agent can meet perceptual learning, a set A of actions that agent can execute on cloud resources, a reward given to the agent, and the environment state can be changed by the action. The agent's cumulative reward is maximized by interacting its action responding to its observations. The optimal policy can be found according to interactively updating the Q function until convergence. At each step, an action's system is chosen based on the system state S, which is denoted $Q(s,a)$. As shown in Fig. 3,

Figure 3 Steps to migrate VMs.



the process of finding the VM migration strategies is modeled as traveling the graph of robot. At the first time, the start state of robot corresponds to without a VM migrated a PM. After performing the action that goes to PM2, the robot state changes state 1 which corresponds to migrating VM1 to PM2. Each step of the robot, he will select a PM i for hosting VM j. In the model of stochastic state, the probability transition matrix is described by $P(s'|s,a)$. The final state of robot when all VMs are migrated PMs. At each step, the robot will select an action that has a good reward in the past denoted $Q(s,a)$. Before the next interaction of management process, the Q function is defined two-dimensional table of $Q(s,a)$ as follows:

$$\begin{aligned} Q(s_t,a_t) \leftarrow (1-\eta)Q(s_t,a_t) + \eta[\mathcal{R}_{t+1} \\ +\gamma max_a Q(\mathcal{P}(s_{t+1},a)) - Q(s_t,a_t)] \end{aligned} \tag{20}$$

where $Q(s_t,a_t)$ is an expected long-term reward for executing the current action a_t in the current state s_t which denotes the t^{th} estimate of $Q^*, \eta \in [0,1]$ is the learning rate that indicates how fast the data of new states will be taken into account in the next steps, the robot does not learn to improve future actions when $\eta = 0$, if $\eta = 1$ then the data on results of the latest management is used by the robot; $\gamma \in [0,1]$ is discount factor which determines the importance of future rewards, if $\gamma = 1$ then the robot takes into account a long-term maximize reward, in case $\gamma = 0$, the robot aspires only the latest reward; $\mathcal{P}(s_{t+1},a)$ is randomly obtained according to the probabilities defined by P and η is a step-size sequence; $max_a Q(\mathcal{P}(s_{t+1},a))$ is an

estimation of the optimal Q-value in the future; the immediate reward $\mathcal{R}_{t+1} = \mathcal{R}(s_t, a_t)$ is observed at every time step given to the robot by environment and can be obtained through a real-world setting or a simulation engine, not requiring the knowledge of either $\mathcal{P}$ or $\mathcal{R}$. After a sufficiently large number of time steps, an approximate optimization policy, i.e., the mapping from a given state s to the action a^*, is taken from the Q table as follows:

$$\Pi(s_t) = a^* = \underset{a}{\text{argmax}} Q(s, a) \tag{21}$$

The objective of the learning agent is to find the best mapping policy $S \rightarrow A$ that maximize expected long-term reward for executing actions. The learning agent can choose control action as following strategies: (i) the choice of random action can occur at the beginning of the management process; (ii) the choice of action defined by the policy Π. The VM migration algorithm is presented as follows:

Algorithm 1 V2PQL - VMs migrate to PMs Q-Learning Algorithm

Input: ϵ, η, γ
Output: Q^*

1: Initialize Q value
2: $Q[i, j] = 0, 1 < i < S, 1 < j < A$
3: Choose action a for current state i
4: $a = \underset{j}{\text{argmax}} Q[i, j]$ with probability $1 - \epsilon$ in Eq.(21)
5: $a = random\{j | j \in A\}$ with probability ϵ
6: Action taken a and let system goes to next state i'
7: Calculate the reinforcement signal
8: $Q(s_t, a_t) \leftarrow (1 - \eta)Q(s_t, a_t) + \eta[\mathcal{R}_{t+1} + \gamma max_a Q(\mathcal{P}(s_{t+1}, a)) - Q(s_t, a_t)]$ in Eq.(20)
9: Repeat step 3 until Q value converge.

The estimates Q converge with probability 1 (w.p.1) to Q^* as long as $\sum_t \eta_t = \infty$, $\sum_t {\eta_t}^2 < \infty$. Watkins first proposed Q-learning algorithm [35] which is later established convergence w.p.1 by Watkins and Dayan [43]. The V2PQL algorithm starts with controlling the VM migration without prior knowledge. The migration policy can be determined by choosing actions that correspond to the highest Q-value after enough explorations.

5 Evaluation

In this section, the efficiency and effectiveness of proposed VM migration approach are demonstrated through the large scale infrastructure cloud computing simulation. The evaluation of VM migration is done through a prototype implementation of V2PQL algorithm running on cloud infrastructure which has hundreds of the needed VM's multi-tier application migration following CloudSim. It is divided into training phase and extraction phase. Initially, the optimal policies are explored by V2PQL algorithm in training phase. And then, these polices are continuously applied to the real time VM migration process by the line 3 to line 8 of V2PQL algorithm in extraction phase. During execution, the VM migration polices which show the strength of V2PQL reinforcement learning algorithm are continuously updated. In training phase, the cumulative reward and the temporary evolution of Q-value which show the efficiency of exploration/exploitation strategies are studied by changing the ϵ parameter of V2PQL algorithm. In extraction phase, the utility of players, load balancing, resource utilization, and running time which show the efficiency of V2PQL algorithm are benchmarked with Round-Robin algorithm.

Table 1 The configuration of PM for generating data.

	CPU Core	RAM (GB)	DISK (GB)
Max	128	256	8192
Min	32	64	512

Table 2 The VM types.

VM type	CPU Core	RAM (GB)	DISK (GB)
Tiny	1	1	5
Small	1	3	15
Medium	2	6	30
Large	4	12	60
X Large	8	24	80

5.1 Environment setup

The simulations to evaluate the performance of VM migration were done on the computer (8GB RAM, Core i5, 256GB SSD). To reduce the complexity of simulations, three kinds of resource are considered in our simulations, i.e., CPU, RAM, Storage of PM, and VM configuration. The heterogeneous datacenter which deploys multi-tier applications is simulated by using the parameters of Table 1. Each multi-tier application is deployed in a cluster VM which the configuration of VM is randomly chosen by Table 2. The VM migration process is trigged when the deteriorating PMs are detected. We set up a data center including 450 PMs, 200 multi-tier applications in which 119 PMs are detected faults and 1543 VMs need to be migrated to safe PMs.

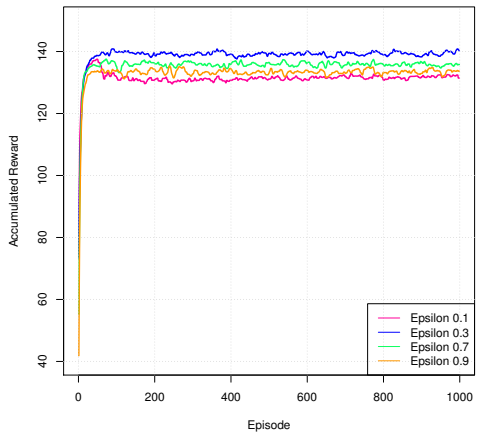
5.2 Training phase

To evaluate the efficiency of V2PQL algorithm, the different investigating learning strategies are considered by a group of simulations. The VM migration policies are depended on the V2PQL parameters including ϵ exploration/exploitation (cf. step 2 in V2PQL algorithm), η learning rate, and γ discount factor of rewards (cf. step 4 in V2PQL algorithm). The exploration/exploitation strategies are invested by changing $\epsilon \in [0.1, 0.9]$ while the learning rate is set to a constant

value $\eta = 0,1$ and the discount factor of reward is set to $\gamma = 0,8$ like [44]. The efficiency of V2PQL algorithm are evaluated in terms of reward, temporal evolution of Q-value.

The cumulative reward over time by following the actions is generated by a policy, starting from an initial state which the robot does not choose a PM for any VM. An episodic task is referred to a complete sequence of interaction, from start to finish. The robot reaches a terminal state when the list of needed VM migration is processed. V2PQL can exploit such knowledge by initializing the Q value (cf. step 1 in V2PQL algorithm) with more meaningful data instead of initializing them with zero as well as can be quicker learning convergence. In $episodic = 1000$, the average rewards as function of the $\epsilon = 0.1, 0.3, 0.7, 0.9$ are described in 4. At $\epsilon = 0.1$ show that the robot seldom focuses on improving future actions, otherwise, $\epsilon = 0.1$ show that the robot focuses on improve future actions. The discounted cumulative reward depicts in 5.

Figure 4 The average reward as function.



The temporary evolution of Q-value refers to each state-action pairs in the learning strategy. The change of Q-values occurs when the system is in state s_t and takes specific action a_i. For instance, in Fig. 6, $q(23, 24)$ shows that the change in the q-value occurs when the system state $s(t)$ is 23 and specific action a_i takes 24. The almost q-value is convergence in $episodic = 1000$.

5.3 Extraction phase

After training phase, the optimal VM migration policies are found out through Q table. The V2PQL policies are benchmarked with Round-Robin policy. The utility of players following Eq.(7) are shown in Fig. 7.

Figure 5 The discounted reward as function.

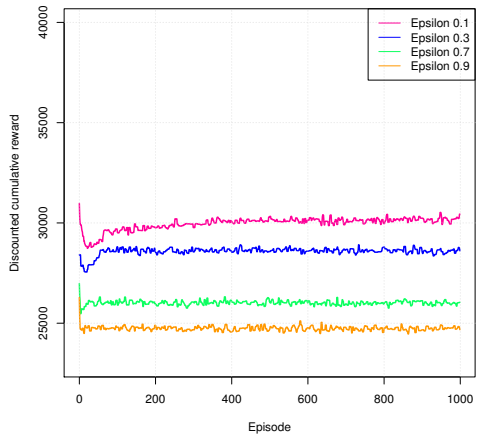
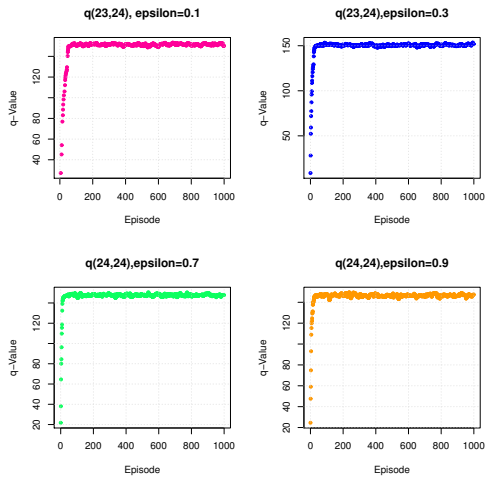


Figure 6 The temporary evolution of Q-value.



The utility of Round-Robin algorithm is distributed to more players than V2PQL algorithm. As shown in Fig 9, the resource utilization of V2PQL algorithm is better than Round-Robin algorithm. However, as shown in Fig. 8, the load balance of Round-Robin algorithm is better V2PQL algorithm. As shown in Fig 10, the running time of V2PQL algorithm with $\epsilon = 0.3$ is better than Round-Robin algorithm in whole VMs while the running time of V2PQL algorithm with $\epsilon = 0.7, 0.9$ is better than Round-Robin algorithm from 500th to 1543th VM. As the result, the V2PQL migration policies have a promising running time.

6 Conclusions

In this paper, the V2PQl algorithm is proposed to solve the VM migration game approach based on MDP. De-

Figure 7 The utility benchmark.

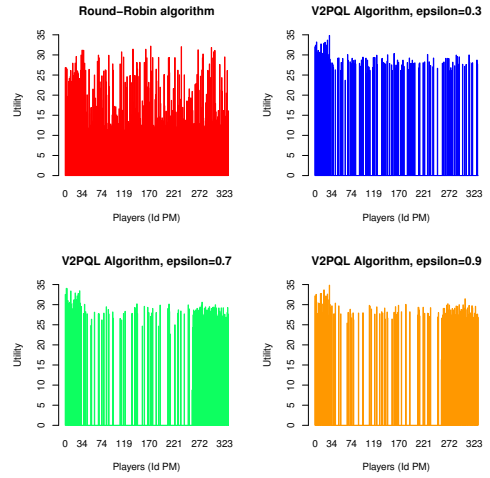


Figure 8 The Load balance.

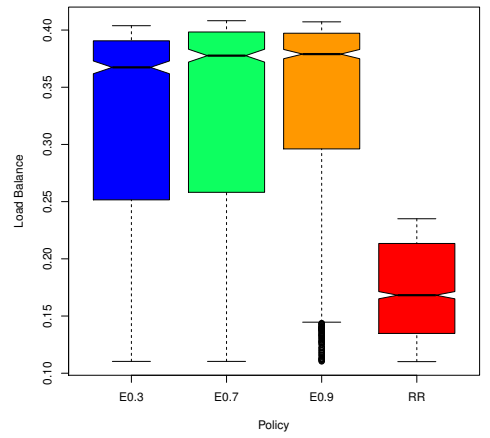


Figure 9 The resource utilization.

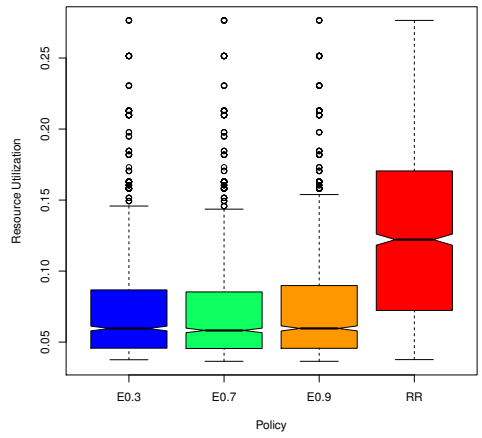
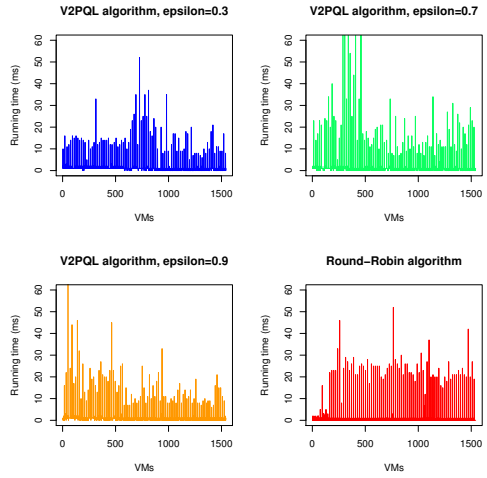


Figure 10 The running time.



pending on the characteristics of each algorithm, the use of strategic construction to migrate VMs for games is also different. The action exploration strategies have been studied by changing ϵ. Therefore, prior knowledge dose not need for VM migration problem if training phase of V2PQL enough. The effectiveness of the algorithm is evaluated by comparing it with the Round-Robin algorithm. In extraction phase, the optimal VM migration policy of V2PQL algorithm is simply applied by choosing the maximum q-value at specify system state. In the future, many other RL algorithms will be developed to compare the evaluation with the proposed algorithm.

Availability of data and materials

The resource code of our algorithm is available as open source Github "https://github.com/buithanhkhiet/v2pql". The repository of V2PQL algorithm contains folders, etc., data, lib, src.

Competing interests

The authors declare that they have no competing interests.

Funding

This research is funded by Thu Dau Mot University under grant number DT.21.1-080.

Authors' contributions

Thanh Khiet is the main author of the manuscript. He has designed the V2PQL algorithm and implemented them in the java programming language. Cong Hung has contributed the "Related work" and "VM migration modeling" and helped Thanh Khiet with fruitful discussions during the algorithm design phase. Dac Hung has contributed the "VM migration game approach" and helped Thanh Khiet with java code implementation in the "Evaluation". Tran Vu has contributed the

"Evaluation" and helped Thanh Khiet with fruitful discussions during the evaluation phase. The author(s) read and approved the final manuscript.

Acknowledgments
We would like to thank Ho Chi Minh City University of Technology (HCMUT) for the support of time and facilities for this study.

Authors' information
Cong Hung Tran received the master of engineering degree in telecommunications engineering course from postgraduate department Hanoi University of technology in Vietnam, 1998. He received Ph.D at Hanoi University of technology in Vietnam, 2004. His main research areas are B – ISDN performance parameters and measuring methods, QoS in high speed networks, MPLS, Wireless Sensor Network, Cloud Computing. He is, currently, Associate Professor PhD. of Faculty of Information Technology II, Posts and Telecommunications Institute of Technology in Ho Chi Minh.
Thanh Khiet Bui received B.Sc. degree on Software Engineering from Ho Chi Minh City University of Technology (HUTECH) in 2010. He acquired his Master's degree from Posts and Telecommunications Institute of Technology in Ho Chi Minh in 2012. He is working at Faculty of Engineering Technology, Thu Dau Mot University as a lecture. At present, he is a Ph.D student at Computer Science, Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT), VNUHCM. His research focuses on Cloud computing.
Dac Hung Ho received B.Sc. degree on Software Engineering from Posts and Telecoms Institute of Technology in Ho Chi Minh in 2014. He acquired his Master's degree from Posts and Telecommunications Institute of Technology in Ho Chi Minh in 2016. He is working at Faculty of Engineering Technology, Thu Dau Mot University as a lecture.
Tran Vu Pham is an associate professor and also the dean of the Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT), VNUHCM, Vietnam. He is interested in developing and applying new and advanced techniques and tools from big data, IoT, and distributed systems to solve real life problems in urban traffic, smart cities, agriculture, etc. Tran Vu Pham received his PhD degree in computing from the University of Leeds, UK.

Author details
[1]Training and Science Technology Department, Posts and Telecommunications Institute of Technology, 11 Nguyen Dinh Chieu, Ho Chi Minh, Vietnam. [2] Faculty of Computer science and Engineering, Ho Chi Minh City University of Technology (HCMUT), 268 Ly Thuong Kiet Street, District 10, Ho Chi Minh, Vietnam. [3]Vietnam National University Ho Chi Minh City, Linh Trung Ward, Thu Duc District, Ho Chi Minh, Vietnam. [4]Faculty of Engineering and Technology, Thu Dau Mot University, 06 Tran Van On Street, Binh Duong, Vietnam.

References

1. Zhang, Q., Cheng, L., Boutaba, R.: Cloud computing: state-of-the-art and research challenges. Journal of internet services and applications **1**(1), 7–18 (2010)
2. Sahni, J., Vidyarthi, D.P.: Heterogeneity-aware adaptive auto-scaling heuristic for improved qos and resource usage in cloud environments. Computing **99**(4), 351–381 (2017)
3. Noshy, M., Ibrahim, A., Ali, H.A.: Optimization of live virtual machine migration in cloud computing: A survey and future directions. Journal of Network and Computer Applications **110**, 1–10 (2018)
4. Bui, K.T., Ho, H.D., Pham, T.V., Tran, H.C.: Virtual machines migration game approach for multi-tier application in infrastructure as a service cloud computing. IET Networks **9**(6), 326–337 (2020)
5. Hartmanis, J.: Computers and intractability: a guide to the theory of np-completeness (michael r. garey and david s. johnson). Siam Review **24**(1), 90 (1982)
6. Bai, W.-H., Xi, J.-Q., Zhu, J.-X., Huang, S.-W.: Performance analysis of heterogeneous data centers in cloud computing using a complex queuing model. Mathematical Problems in Engineering **2015** (2015)
7. Guo, Y., Stolyar, A., Walid, A.: Online vm auto-scaling algorithms for application hosting in a cloud. IEEE Transactions on Cloud Computing (2018)
8. Huang, G., Wang, S., Zhang, M., Li, Y., Qian, Z., Chen, Y., Zhang, S.: Auto scaling virtual machines for web applications with queueing theory. In: 2016 3rd International Conference on Systems and Informatics (ICSAI), pp. 433–438 (2016). IEEE
9. Morton, T., Pentico, D.W.: Heuristic Scheduling Systems: with Applications to Production Systems and Project Management vol. 3. John Wiley & Sons, ??? (1993)
10. Van Laarhoven, P.J., Aarts, E.H., Lenstra, J.K.: Job shop scheduling by simulated annealing. Operations research **40**(1), 113–125 (1992)
11. Ghumman, N.S., Kaur, R.: Dynamic combination of improved max-min and ant colony algorithm for load balancing in cloud system. In: 2015 6th International Conference on Computing, Communication and Networking Technologies (ICCCNT), pp. 1–5 (2015). IEEE
12. Tsai, C.-W., Rodrigues, J.J.: Metaheuristic scheduling for cloud: A survey. IEEE Systems Journal **8**(1), 279–291 (2013)
13. Levin, E., Pieraccini, R., Eckert, W.: Using markov decision process for learning dialogue strategies. In: Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181), vol. 1, pp. 201–204 (1998). IEEE
14. Van Otterlo, M., Wiering, M.: Reinforcement learning and markov decision processes. In: Reinforcement Learning, pp. 3–42. Springer, ??? (2012)
15. Bui, K.T., Nguyen, L.V., Tran, T.V., Pham, T.-V., Tran, H.C.: A load balancing vms migration approach for multi-tier application in cloud computing based on fuzzy set and q-learning algorithm. In: Research in Intelligent and Computing in Engineering, pp. 617–628. Springer, ??? (2021)
16. Xu, X., Yu, H.: A game theory approach to fair and efficient resource allocation in cloud computing. Mathematical Problems in Engineering **2014** (2014)
17. Fujiwara-Greve, T.: Non-cooperative Game Theory vol. 1. Springer, ??? (2015)
18. Gao, Y., Guan, H., Qi, Z., Hou, Y., Liu, L.: A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. Journal of computer and system sciences **79**(8), 1230–1242 (2013)
19. Silva Filho, M.C., Monteiro, C.C., Inácio, P.R., Freire, M.M.: Approaches for optimizing virtual machine placement and migration in cloud environments: A survey. Journal of Parallel and Distributed Computing **111**, 222–250 (2018)
20. Cheng, L., Li, T.: Efficient data redistribution to speedup big data analytics in large systems. In: 2016 IEEE 23rd International Conference on High Performance Computing (HiPC), pp. 91–100 (2016). IEEE
21. Siar, H., Kiani, K., Chronopoulos, A.T.: An effective game theoretic static load balancing applied to distributed computing. Cluster Computing **18**(4), 1609–1623 (2015)
22. Liu, L., Mei, H., Xie, B.: Towards a multi-qos human-centric cloud computing load balance resource allocation method. The Journal of Supercomputing **72**(7), 2488–2501 (2016)
23. Ficco, M., Esposito, C., Palmieri, F., Castiglione, A.: A coral-reefs and game theory-based approach for optimizing elastic cloud resource allocation. Future Generation Computer Systems **78**, 343–352 (2018)
24. Ye, D., Chen, J.: Non-cooperative games on multidimensional resource allocation. Future Generation Computer Systems **29**(6), 1345–1352 (2013)
25. Chang, X., Nie, F., Wang, S., Yang, Y., Zhou, X., Zhang, C.: Compound rank-k projections for bilinear analysis. IEEE transactions on neural networks and learning systems **27**(7), 1502–1513 (2015)
26. Sahoo, S.R., Gupta, B.: Multiple features based approach for automatic fake news detection on social networks using deep learning. Applied Soft Computing **100**, 106983 (2021)
27. Wang, H., Li, Z., Li, Y., Gupta, B., Choi, C.: Visual saliency guided complex image retrieval. Pattern Recognition Letters **130**, 64–72 (2020)
28. Yuan, D., Chang, X., Huang, P.-Y., Liu, Q., He, Z.: Self-supervised deep correlation tracking. IEEE Transactions on Image Processing **30**, 976–985 (2020)
29. Dhanoa, I.S., Khurmi, S.S.: Analyzing energy consumption during vm live migration. In: International Conference on Computing, Communication & Automation, pp. 584–588 (2015). IEEE
30. Rybina, K., Schill, A.: Estimating energy consumption during live migration of virtual machines. In: 2016 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), pp. 1–5 (2016). IEEE
31. Yang, L., Feng, Y., Li, K.: Optimization of virtual resources

provisioning for cloud applications to cope with traffic burst. In: 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC), pp. 80–87 (2017). IEEE
32. Raghunath, B.R., Annappa, B.: Dynamic resource allocation using fuzzy prediction system. In: 2018 3rd International Conference for Convergence in Technology (I2CT), pp. 1–6 (2018). IEEE
33. Hsieh, S.-Y., Liu, C.-S., Buyya, R., Zomaya, A.Y.: Utilization-prediction-aware virtual machine consolidation approach for energy-efficient cloud data centers. Journal of Parallel and Distributed Computing **139**, 99–109 (2020)
34. Zhang, T., Niu, J., Liu, S., Pan, T., Gupta, B.B.: Three-dimensional measurement using structured light based on deep learning. COMPUTER SYSTEMS SCIENCE AND ENGINEERING **36**(1), 271–280 (2021)
35. Watkins, C.J.C.H.: Learning from delayed rewards (1989)
36. Xu, C.-Z., Rao, J., Bu, X.: Url: A unified reinforcement learning approach for autonomic cloud management. Journal of Parallel and Distributed Computing **72**(2), 95–105 (2012)
37. Farahnakian, F., Liljeberg, P., Plosila, J.: Energy-efficient virtual machines consolidation in cloud data centers using reinforcement learning. In: 2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, pp. 500–507 (2014). IEEE
38. Rolik, O., Zharikov, E., Koval, A., Telenyk, S.: Dynamie management of data center resources using reinforcement learning. In: 2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET), pp. 237–244 (2018). IEEE
39. Saovapakhiran, B., Michailidis, G., Devetsikiotis, M.: Aggregated-dag scheduling for job flow maximization in heterogeneous cloud computing. In: 2011 IEEE Global Telecommunications Conference-GLOBECOM 2011, pp. 1–6 (2011). IEEE
40. Bui, K.T., Pham, T.V., Tran, H.C.: A load balancing game approach for vm provision cloud computing based on ant colony optimization. In: International Conference on Context-Aware Systems and Applications, pp. 52–63 (2016). Springer
41. Xiao, Z., Song, W., Chen, Q.: Dynamic resource allocation using virtual machines for cloud computing environment. IEEE transactions on parallel and distributed systems **24**(6), 1107–1117 (2012)
42. Duong, T., Chu, Y.-J., Nguyen, T., Chakareski, J.: Virtual machine placement via q-learning with function approximation. In: 2015 IEEE Global Communications Conference (GLOBECOM), pp. 1–6 (2015). IEEE
43. Watkins, C.J., Dayan, P.: Q-learning. Machine learning **8**(3-4), 279–292 (1992)
44. Jamshidi, P., Sharifloo, A.M., Pahl, C., Metzger, A., Estrada, G.: Self-learning cloud controllers: Fuzzy q-learning for knowledge evolution. In: 2015 International Conference on Cloud and Autonomic Computing, pp. 208–211 (2015). IEEE