Computing
https://doi.org/10.1007/s00607-021-01047-0

REGULAR PAPER

Virtual machine migration policy for multi-tier application in cloud computing based on Q-learning algorithm

Cong Hung Tran[1] · Thanh Khiet Bui[2,3,4] · Tran Vu Pham[3,4]

Received: 17 July 2021 / Accepted: 13 December 2021

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, stakeholders' objectives are usually conflict with each other. Furthermore, 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 the 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 algorithm has been benchmarked to the Round-Robin, inverse Ant System, Max–Min Ant System, and Ant System algorithms in order to highlight its strength and feasibility in extraction phase.

Keywords VM migration · Game theory · Cloud computing · Q-learning

Mathematics Subject Classification 68Wxx

✉ Thanh Khiet Bui
khietbt@tdmu.edu.vn

1 Training and Science Technology Department, Posts and Telecommunications Institute of Technology, 11 Nguyen Dinh Chieu, Ho Chi Minh, Vietnam

2 Institute of Engineering and Technology, Thu Dau Mot University, 06 Tran Van On Street, Thu Dau Mot, Binh Duong, Vietnam

3 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

4 Vietnam National University Ho Chi Minh City, Linh Trung Ward, Thu Duc District, Ho Chi Minh, Vietnam

Published online: 29 January 2022

1 Introduction

Cloud computing has made computing resources more and more powerful, abundant, and cheaper [36]. The capacity of cloud-based systems and infrastructure characteristics are critical features [2]. Their resources can be adjusted with a user-on-demand mechanism to adapt to dynamic environment and ensure the quality of service (QoS), and the service level agreements (SLA). Users can access resources in a multi-tenant way thanks to virtual technology. It allows users to create multiple virtual machines (VMs) on a physical server (PM) to improve the applications abilities and software deployments. Each VM is also allocated hardware resources like real machines with RAM, CPU, network card, hard drive, operating system, and other applications. Cloud-based applications are commonly designed by service-oriented architecture (SOA) which delivers a cluster of separate services and communicates with each other flexibly [25].

VM migration is one of major virtual technology advantages that makes VMs free from the underlying hardware to flexibly manage resources and increases shared-resource utilization. A VM can be migrated from one PM to another PM while the VM is still running during migration [22]. By moving VMs from over-loaded PMs to light-loaded ones, a cloud system may achieve load balancing. VMs that are operating on a light-load PM can be merged into another PM to save energy by lowering the number of running PMs. Migrating VMs to other PMs may be used to build a proactive fault tolerance strategy that prevents predicted problems from occurring in PMs [3]. In addition, the performance of cloud-based applications can be improved by migrating some VMs from their limited resource PMs to richer ones. However, VM migration aims at different purposes of stakeholders' objectives including service providers, customers, end-users that faces the following challenges of cloud resource management.

- Stakeholders' goals, including cloud service providers and consumers, are controversial to each other. Resource management helps cloud service providers maximize system utilization and get high profits while customers' benefit from resource management can be ensured by SLA. However, maximizing resource utilization would make it harder to make both performance and QoS requirements for consumers. Customers, on the other hand, desire to reduce their expenses, which leads to a reduction in service time by demanding more resources. It can be inferred that the desired relationship of cloud service providers and consumers may be conflict.
- The physical resources in the cloud computing environment, as well as consumer resource requirements, are not homogeneous. They can lead to resource fragmentation and resource waste. This issue requires new methods to coordinate resources in stochastic, complex, and heterogeneous systems with limited prior knowledge.
- The resource problems in large-scale environment have been classified into the NP-Hard or NP-Complete class [15]. Distributed and parallel computing methods, such as task scheduling for many processors, bin packing, and graph partitioning algorithms, can be used to derive VM migration algorithms [1,14,17]. Exhaustive algorithms, deterministic algorithms, or meta-heuristic algorithms are typically

used to address these issues due to particular features [13,21,29]. Exhaustive algorithms perform worse in tests than deterministic algorithms do. Furthermore, in a large-scale setting, deterministic algorithms are inefficient [28]. To achieve QoS, cloud services must respond to customers as quickly as feasible. Furthermore, due of the elasticity and resource sharing concept, not only cloud systems but also cloud-hosted applications become more complex in running time.

The VM migration challenge explored in this study is motivated by stakeholder goals and focused on maximizing resource usage while guaranteeing customers' SLAs by balancing the load among PMs to minimize concurrency and congestion. The VM migration problem for multi-tier applications is formalized in non-cooperative game theory for ensuring stakeholders' goals. In this study, web-based multi-tier systems focused on three-tier web applications with a web server tier, a business logic tier, and a database server tier. They based on SOA, as opposed to monolithic applications with tightly linked parts, are ideally suited for cloud infrastructures. 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. Based on Markov Decision Process (MDP) [4,19,30], 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.

(i) The VM migration for multi-tier application is modeled by using non-cooperative game theory to describe the conflict among cloud service providers 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 [10,33]. Each player tries to maximize their own utility by changing their strategies which trade-offs the load balance and resource utilization.

(ii) 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.

(iii) The heterogeneous data center which deploys one hundred multi-tier applications is stimulated. The optimal VM migration policy is investigated by the V2PQL algorithm in the training phase. The utility of V2PQL policy is benchmarked with the Round-Robin, inverse Ant System, Max–Min Ant System, and Ant System algorithms Round-Robin policy in the extraction phase.

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

2 Related work

Many researches on VM migration strategies in cloud environment have been proposed in recent years. Each migration approach was created with specific goals such as emphasizing application architecture, resource utility, or service quality. In this section, stakeholders' objectives and machine learning model for migrating VMs are discussed.

Stakeholders' objectives, which include service providers and customers' objectives, take into account selecting available resources PM to deploy VM with different objectives [11]. Service providers concern about minimizing resource consumption and maximizing service level agreement (SLA) compliance. Customers, on the other hand, desire to reduce their cost, which leads to a reduction in service time by demanding more resources. Monitoring data like as CPU load, memory, storage, network status, and configuration settings are frequently used in optimal VM migration algorithms. VM distribution should guarantee efficient usage of PM resources towards service providers or customers or both. Minarolli et al. proposed a CPU allocation model for infrastructure cloud concerning both service quality and operating costs [20]. The model has two layers:(i) the local controller ensures dynamic allocation of shared CPU for VMs to achieve local optimization and (ii) the global controller manages VM migration to maximize resource utility. However, this study only focused on CPU resources not memory, disks and network. In [35], Deshi et al. considered server load balancing game and VM placement game in which system resources are described as multi-dimensional vectors. The non-cooperative game model is applied to formulate the problem. VMs are assigned to the smallest PM amount that workload per PM is within a given power. In [27], PMs running with under-loaded lead to energy waste while overloaded states results shorten PMs lifespan, and then reduces QoS. The problem is solved by game theory method to develop VM migration solutions that meet the stakeholders' needs. By migrating VMs, the loads can be balanced among PMs in data-center to ensure QoS. In [1], Bai et al. proposed a method to evaluate the performance of applications on cloud computing for both cloud providers and cloud service customers. A complex queue model composed of two concatenated queuing systems queuing systems is proposed to evaluate the performance of heterogeneous data centers. The flow density of nodes is evaluated by analyzing the QoS metrics such as average response time, average waiting time in a heterogeneous data center. These metrics are used to manage nodes that are to be removed from a cloud or added to it. In [5], the VM provision approach of cloud computing has been proposed to achieve load balancing. The problem is formulated by non-cooperate game to trade-offs between load balancing and profit maximization. The solution of VM provision is found by Ant Colony Optimization algorithm. Massimo Ficco et al. proposed a meta-heuristic approach for cloud resource allocations based on the model of biological-inspired coral ecology optimization [8]. Based on the game theory, the optimum resource allocation strategies are searched to ensure both service providers' aims and customers' requirements. The evolutionary algorithm is proposed by imitating the structure of coral reefs and spawning corals. It also exploits the dynamism of competition among users and service providers to satisfy stakeholders' benefits. 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 [3], Khiet et al. proposed a proactive fault tolerant based on PM fault prediction and VM migration for infrastructure cloud. The PM fault prediction model is designed by applying Takagi-Sugeno fuzzy system with metrics of PMs. The VM migration process is triggered through iAS algorithm when PM faults occur. The iAS algorithm, which determines the safe PMs for needed VM migration, is improved more effectively than Ant System algorithm.

Furthermore, VM migration algorithms have been considered to be integrated machine learning models to highly support in making decision [7,16,23,24,34]. 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. Farahnakian et al. proposed a dynamic consolidation method based on reinforcement learning to minimize the number of active hosts according to the current resource requirements [7]. The host power mode can be determined by an agent without prior knowledge of the environment and workload. The CPU and memory resources can be dynamically provisioned for a VM according to the current resource requirements. The CPU for a virtual machine can be dynamically provided based on the current resource requirements. However, the proposed algorithm focuses on only CPU performance but not other host resources. In [24], Rybina et al. proposed a model which models which can estimate the energy consumption of the PMs during the VM migration based on the resource utilization parameters of these PMs. By reaching more accurate prediction model, better migration decisions are made in data center power management by applying multiple linear regression techniques. Therefore, VM migration with less time will help in minimizing power cost during the migration process. The quantitative energy cost of migration can help us select the best PM candidates for VM migration. Yang et al. proposed the MASVR model to predict workload for ensuring Web application QoS and remaining low cost for service providers [34]. The LOF-MASVR workload prediction model is developed by combining Moving Average (MA) model and Support Vector Regression (SVR) model. A dynamic provisioning approach was developed using LOF-MASVR model by incorporating a self-adaptive coefficient that may reflect resource usages. Experiments indicate that, as compared to traditional provisioning rules, the suggested method can be effectively adapted to the changing workloads with more evident benefits over ensuring QoS. In [23], Rolik et al. analyzed the possibility of cloud-based applications to 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. The changes in the resource utilization is evaluated to make under-loaded PMs to the sleep mode in order to reduce the power consumption. The Q-learning technique enables for the determination of the best policy for managing PMs without the need for an environment model or preliminary workload information. In [16], Hsieh et al. proposed the virtual

Table 1 Classification of the related work

	Field	Aims	Approach
[20]	Resource management	Multi-objective	Threshold
[11]	VM placement	Multi-objective	Ant colony system
[35]	Resource allocation	Load balancing	Non-cooperative games
[27]	Distributed computing	Load balancing	Evolutionary algorithms
[1]	PM consolidation	Energy-aware	Queuing model
[5]	VM provision	Load balancing	Ant Colony Optimization
[8]	Resource allocation	Elastic resources	Coral-reefs optimization
[3]	Fault tolerant	Multi-objective	Inverse Ant System
[7]	VM consolidation	Energy-efficient	Reinforcement learning
[24]	Live migration	Energy consumption	Linear regression
[34]	Resources allocation	SLA	Adaptive coefficient
[23]	Resource management	SLA	Reinforcement learning
[16]	VM consolidation	Energy-efficient	Gray-Markov-based model
[4]	VM migration	Load balancing	Q-learning and Fuzzy set

machine consolidation approach using the utilization prediction for energy-efficient cloud data centers to reduce the power costs as well as preserve the QoS guarantee. A Gray-Markov-based model is used to properly estimate future resource utilization by UP-POD (the host overload detection) and UP-PUD (the host under-load detection). As a result, the virtual machine consolidation approach effectively decreases energy consumption of the PMs, the number of migrations, and SLA violations. In [4], the VM migration approach for multi-tier applications has been proposed to ensure load balancing. The problem is formulated by Markov decision process and fuzzy set. The VM migration solutions are found by Q-Learning algorithm.

The conflicting stakeholder objectives between service providers and their customers have been widely modeled by game theory [3,8,27]. Besides, machine learning models are applied to support the decision making of VM migration process [4,16,23,34]. However, the game models of VM migration are rarely solved by the machine learning models. Therefore, this paper proposes a reinforcement learning for the game of VM migration problem to determine the safe PMs for the VMs on the detected deteriorating PM with no prior knowledge about characteristics of the cloud system. The classification of related work is described in Table 1 to easily compare with ours.

3 Virtual machine migration game

In this section, the VM migration problem is modeled by a non-cooperative game model with PMs as players. Each player tries to maximize own utility which trades off the load balance and the resource utilization. We proved that there exist Nash

Table 2 Notations

Symbol	Meaning
n	A given number of VMs
m	A given number of PMs
k	Number of PM resource types demanded by a VM
$\mathbf{X}^{n\times m}$	A VM migration matrix of migrating n VMs to m PMs
$\mathcal{M}_i^{n\times k}$	A resource allocation matrix of the PM i
$\mathcal{M}$	A resource allocation vector of m PMs, i.e, $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_m\}$
σ_j	A load imbalance of the resource type j of m PMs
$\mathcal{L}$	A load balancing level of resource types of m PMs
$u_j^{(i)}$	An utilization of the resource type j of the PM i
$\mathcal{H}^{(i)}$	An utilization unevenness of all resource types of the PM i
G	A VM migration game is described as a three-tuple vector $G = (\mathcal{P}, \mathcal{M}_i, f_i)$
$\mathcal{P}$	A set of players in the game, i.e, $\mathcal{P} = \{1, 2, \dots, m\}$
$f^{(i)}$	An utility function of the player i
$\mathcal{S}$	A set of system states
$\mathcal{A}$	A set of VM migration actions
$\mathcal{P}$	A transition probabilities
$\mathcal{R}$	A reward structure
μ	A membership functions of the load balance and the resource utilization
s_t	A system state $s \in \mathcal{S}$ at time t
a	An action $a \in \mathcal{A}$ of migrating a VM to a PM
$Q(s_t, a)$	A Q function of the system state s_t with the action a
$\Pi(s, a)$	A probability of selecting the action a from the state s
η	A given learning rate
γ	A given discount factor determines the importance of future rewards

equilibrium for the VM migration approach. As shown in Table 2, the notations of the VM migration game are presented in details.

3.1 VM migration modeling

Suppose that a cloud infrastructure has large scaled heterogeneous PMs and provides computational resources as on demand model. There are a lot of PMs deploying VMs based on virtualization technology. The cloud provider offers a group of VM types to remove complex selections for customers. Each VM type is specifically determined by the number of CPUs, memory size and storage size. Resource allocations of the cloud infrastructure have been adjusted to meet customers' needs. A VM migration

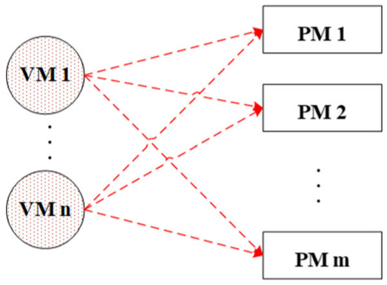


Fig. 1 The directed acyclic graph (DAG) of VM migration problem

process is triggered when the deteriorating PMs are detected. At the moment, the cloud infrastructure has n VMs needing migration to m safe PMs. As shown in the Fig. 1, it is possible to model the VM migration problem in cloud environment with the directed acyclic graph $\mathbf{G}(V, E)$ [5,26]. V is a set of vertices representing the work while E is a set of edges showing a dependency relationship among vertices.

Definition 1 (*Migration decision*) A possible resource allocation of migrating n VMs to m PMs is described as a binary matrix $\mathbf{X}$:

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

where $x_{pi} = \{0, 1\}$, migrating VM $p = \{1, 2, \ldots, n\}$ to PM $i = \{1, 2, \ldots, m\}$ is described $x_{pi} = 1$, otherwise $x_{pi} = 0$.

Definition 2 (*Allocation decision*) Allocating k resource types of the PM i is described as a allocation matrix $\mathcal{M}_i$:

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

where $v_{pj}^{(i)} \in \mathbb{Z}_+$ shows the amount of resource types $j = \{1, 2, \ldots, k\}$ of PM i providing to VM p.

A vector $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_m\}$ is considered as a possible resource allocation strategy for all of PMs. An optimal VM migration problem showing the trade-off between the load balance and the resource utilization can be described by on the non-cooperate game theory.

3.1.1 Load balance

A multi-dimensional vector is adopted to point out a VM or a PM loads. The load balancing of PMs in data-center is calculated through each dimensional load. Like [12], an amount of load imbalance of the resource type j of all PMs is calculated by using the coefficient of variance (CV) theory. Let σ_j denotes that the amount of load imbalance is calculated as following.

$$\sigma_j = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\frac{e_j^{(i)}}{\bar{e}_j} - 1\right)^2} \tag{3}$$

where $e_j^{(i)} = \sum_{p=1}^{n} v_{pj}^{(i)} x_{pi}$ describes the resource type j of PM i allocating all of VMs, $\bar{e}_j = \frac{1}{m}\sum_{i=1}^{m}\sum_{p=1}^{n} v_{pj}^{(i)} x_{pi}$ shows the average resource type j all of PMs. Consequently, the load balancing level of all resource types in data-center is calculated as following.

$$\mathcal{L} = \frac{1}{k}\sum_{j=1}^{k}\sigma_j \tag{4}$$

3.1.2 Resource utilization

To service providers, in order to achieve high profits, the utilization resources should avoid wastage resources of PMs. The concept of skewness in [32] is applied to quantify the unevenness of the different resource utilization of the PM i, which is calculated as following:

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

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

3.2 VM migration game approach

In this section, VM migration problem is formalized by the non-cooperate game theory. The safe PMs are considered as players which trade-off between the load balancing in Eq.(4) and the resource utilization in Eq.(5). The utility function of the player i is designed as following.

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

The utility function has important influence on players' decisions and game outcomes. 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 that resources of the PM i do not exceed its capacity and the constraint (9) ensures that a VM must be migrated to one and only one PM.

Definition 3 (*VM migration game*) A VM migration game is described as a three-tuple vector $G = (\mathcal{P}, \mathcal{M}_i, f_i)$.

(i) $\mathcal{P}$ is the finite set of players in the game, i.e, $\mathcal{P} = \{1, 2, \ldots, m\}$.
(ii) $\mathcal{M}_i$ is the set of available strategies for the player i.
(iii) $f_i = \mathcal{M} \to \mathbb{R}$ is the utility function for the player i.

The Nash equilibrium of the game is a state in which no player can increase his utility by changing his strategy while 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. To any players i, every element $\beta^{(i)} \in \mathcal{M}^{(i)}$ is a strategy for the player i, the $\boldsymbol{\beta}^{(-i)} = [\beta^{(j)}]_{j\in\mathcal{P}, j\neq i}$ describes strategies of all players, excepting the player i, and the $\beta = (\beta^{(i)}, \boldsymbol{\beta}^{(-i)})$ is referred 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 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 all players' strategies except i, the $br^{(i)}$ is the best response of player i, and the $br^{(i)}(\boldsymbol{\beta}^{(-i)*}) = \beta^{(i)*} \in \boldsymbol{\beta} | f^{(i)}(\beta^{(i)*}, \boldsymbol{\beta}^{(-i)}) \ge f^{(i)}(\beta^{(i)}, \boldsymbol{\beta}^{(-i)})$.

By defining a set of the $br : \boldsymbol{\beta} \to \boldsymbol{\beta}$ by $br(\beta^{(i)}) = \times_{i\in\mathcal{P}} br(\boldsymbol{\beta}^{(-i)})$, the Eq. (10) can be rewritten in a vector form as $\beta^* \in br(\beta^*)$. The existence of β^* for which the $\beta^* \in br(\beta^*)$ 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. in the definition of $br^{(i)}(\boldsymbol{\beta}^{(-i)}) = \underset{\beta^{(i)}}{\text{argmax}} f^{(i)}(\beta^{(i)}, \boldsymbol{\beta}^{(-i)})$ 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:
  $f^{(i)}(\beta^{(i)\prime}, \boldsymbol{\beta}^{(-i)}) \geq f^{(i)}(a_i, \boldsymbol{\beta}^{(-i)})$ for all $a_i \in \beta^{(i)}$,
  $f^{(i)}(\beta^{(i)\prime\prime}, \boldsymbol{\beta}^{(-i)}) \geq f^{(i)}(a_i, \boldsymbol{\beta}^{(-i)})$ for all $a_i \in \beta^{(i)}$.
- The preceding relations imply that for all $\lambda \in [0, 1]$, we have:
  $\lambda f^{(i)}(\beta^{(i)\prime}, \boldsymbol{\beta}^{(-i)}) + (1-\lambda) f^{(i)}(\beta^{(i)\prime\prime}, \boldsymbol{\beta}^{(-i)}) \geq f^{(i)}(a_i, \boldsymbol{\beta}^{(-i)})$ for all $a_i \in \beta^{(i)}$.
- By the linearity of f_i, then we have:
  $f^{(i)}(\lambda\beta^{(i)\prime} + (1-\lambda)\beta^{(i)\prime\prime}, \boldsymbol{\beta}^{(-i)}) \geq f^{(i)}(a_i, \boldsymbol{\beta}^{(-i)})$ 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

The resource management in cloud environment has been considered as an automatic controlling problem by using the reinforcement learning approach. Reinforcement learning is one of the machine learning methods in which the agent takes actions into impacting the environment to minimize the total amount of penalties from action results. The Markov Decision Process (MDP) model includes a set of states $\mathcal{S}$ and a set of actions $\mathcal{A}$, transition probabilities $\mathcal{P}$, and rewarding structures $\mathcal{R}$ which are completely specified. However, transition probabilities are often unknown for real-work setting. Furthermore, the state and action spaces are often too large for algorithms to handle [6]. To solve this problem, the V2PQL algorithm is proposed to find a Nash equilibrium of VM migration strategies by influencing the observed system states and rewards. The V2PQL algorithm does not require the prior knowledge of model parameters.

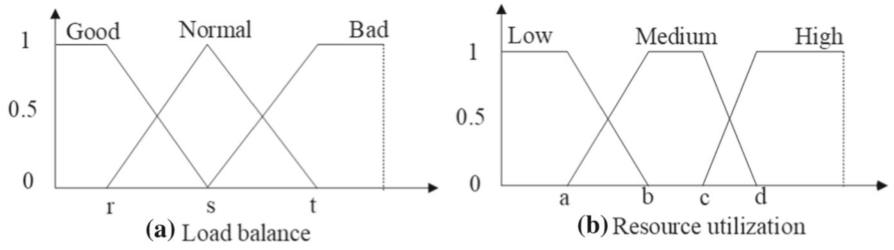


Fig. 2 The membership function charts of load balance, and resource utilization

4.1 MDP framework for VM migration

The V2PQL algorithm is developed by a discrete-time MDP model. 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 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 is applied to the load balance in Eq. (4) and the utilization resource in Eq. (5). The Fig. 2a depicts the membership function of load balance including three states, i.e., $Good$, $Normal$, and Bad, which is calculated as following:

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

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

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

The Fig. 2b shows the membership function of resource utilization including three states, i.e., Low, $Medium$, and $High$, which is calculated as following:

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

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

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

Definition 5 (*Transition probabilities*) A state $s_t \in \mathcal{S}$ at time t is defined as a three-tuple including the load balance state, the resource utilization state, and the type of migrated VM, i.e., $s_t = (\mathcal{L}[t], \mathcal{H}[t], \vartheta[t])$. An action $a \in \mathcal{A}$ of migrating the VM p to the safe PM i corresponds to changing the x_{pi} in Eq. (1) from 0 to 1 and adding the type of migrated VM to vector $\vartheta[t]$. The transition probability matrix $\mathcal{P}(s'|s, a)$ can be analytically derived for a stochastic model.

Definition 6 (*Reward structure*) The optimization problem (7)–(9) describes the benefit of a current VM migration solution showing the system state snapshot. A reward $\mathcal{R}(s, a)$ of the VM migration solution is defined by using the utility function (7):

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

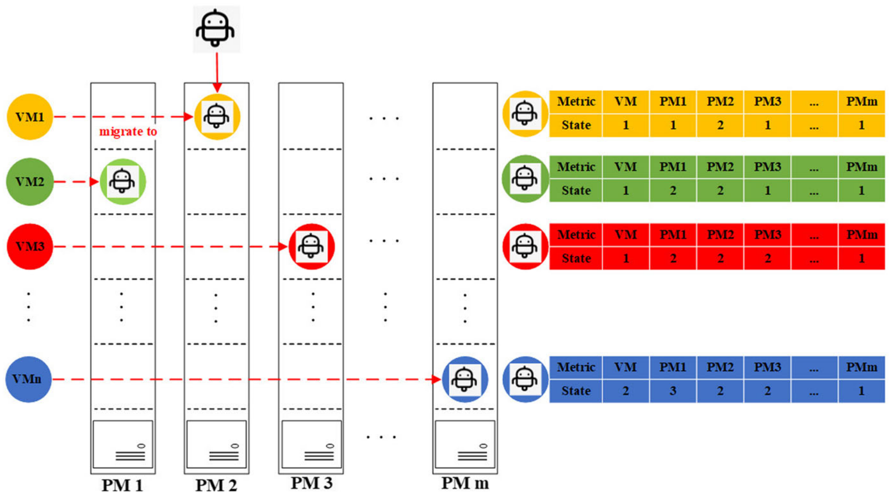
An optimal MDP policy is a mapping from a set of states $\mathcal{S}$ to a set of actions $\mathcal{A}$ based on maximizing the average reward of discounted cumulative reward over time. The reward function can serve as a basic element to change the policy. By using the modified reward function architecture,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

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

where $\alpha < 1$ represents a discounted value and n is number of iterations. The optimal policy is achieved by $\underset{a}{\operatorname{argmax}} Q^*(s)$ where $Q^*(s)$ are convergent values from equation (18).

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

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



Metric	VM	PM1	PM2	PM3	...	PMm
State	1	1	2	1	...	1

Metric	VM	PM1	PM2	PM3	...	PMm
State	1	2	2	1	...	1

Metric	VM	PM1	PM2	PM3	...	PMm
State	1	2	2	2	...	1

Metric	VM	PM1	PM2	PM3	...	PMm
State	2	3	2	2	...	1

Fig. 3 Steps to migrate VMs

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

4.2 VM migration algorithm

To find out VM migration strategies, a model-free version of learning agent is developed by applying Q-Learning algorithm. An optimal VM migration solution can be achieved by interacting between a learning agent and its environment without any prior knowledge. The Q-learning model is presented by a finite set of system states S that learning agent can meet perceptual learning, a set of actions A that the agent can execute on cloud resources, a reward given to the agent, and a system state can be changed by an action [9]. The agent's cumulative reward is maximized by interacting its action responding to its observations. An 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 the Fig. 3, finding VM migration solutions is modeled as traveling the graph of the agent. At the first time, a starting state of the agent corresponds without migration events. After the agent performs the action choosing PM2, the agent state changes to state 1 corresponding migration of VM1 to PM2. In each step of the agent, he will select the PM i for hosting the VM p. In the model of stochastic state, the probability transition matrix is described by $P(s'|s, a)$. In a final state, all VMs are migrated to PMs. In every step, the agent receives a reward denoting $Q(s, a)$, which is defined as following.

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

where:

- $Q(s_t, a)$ is expected long-term rewards of executing the action a at the current state s_t.
- $\eta \in [0, 1]$ is a learning rate that indicates how fast the data of new states will be taken into account in the next steps.
- $\gamma \in [0, 1]$ is a discount factor which determines the importance of future rewards.
- $\mathcal{P}(s_{t+1}, a)$ is randomly obtained according to the probabilities defined by $\mathcal{P}$.
- $\max_a Q(\mathcal{P}(s_{t+1}, a))$ is an estimation of 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 agent by environment. It can be obtained through a real-world setting or a simulation engine without requiring the knowledge of either $\mathcal{P}$ or $\mathcal{R}$.

After a sufficiently large number of time steps, an approximate optimization policy mapping from the given state s to the action a^* is taken from the Q-table as following.

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

The agent's objective is to find the best mapping policy $S \rightarrow A$ that maximize expected long-term rewards of executing actions. The agent can choose a control action randomly or greedy for building VM migration solutions. The V2PQL algorithm starts without prior knowledge at innit time. After running enough explorations, the migration policy can be determined by choosing actions that correspond to the highest Q-value. The V2PQL VM migration algorithm is presented as follows.

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

```
Input: $\epsilon, \eta, \gamma$
Output: $Q^*$

1: Initialize $Q^*$
2: $Q[i, j] = 0, i \in S, j \in A$
3: Choose the action $a$ for the current state $i$
4: $a = \underset{j}{\operatorname{argmax}} Q[i, j]$ with probability $1 - \epsilon$ in Eq.(21)
5: $a = random\{j | j \in A\}$ with probability $\epsilon$
6: The action $a$ is taken and let the system go to the next state $s_{t+1}$
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 [9] which is later established convergence w.p.1 by Watkins and Dayan [31].

Table 3 The PM configuration for generating data

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

Table 4 The VM types

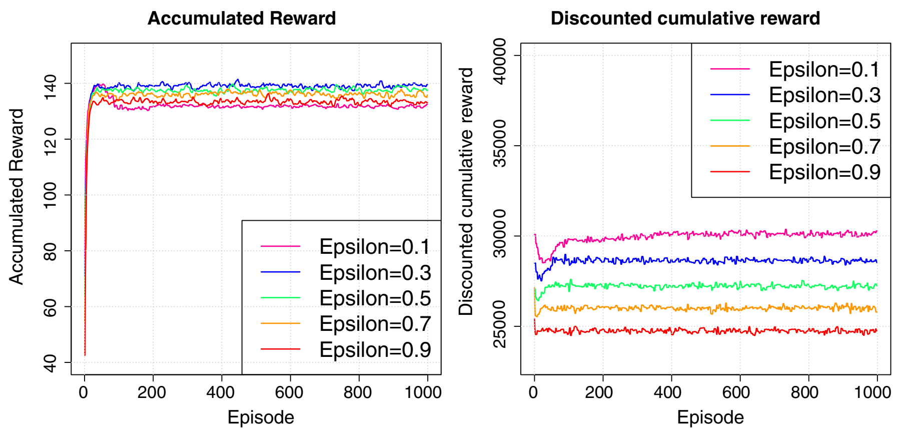
VM type	CPU Core	RAM (GB)	Disk (GB)
Tiny	1	1	5
Small	1	3	15
Medium	2	6	30
Large	4	12	60
X Large	8	24	80

5 Evaluation

In this section, the efficiency and effectiveness of the proposed VM migration approach are demonstrated through a large scale infrastructure cloud computing simulated on CloudSim. The cloud infrastructure with a lot of PMs provides infrastructure services to multi-tier applications. The V2PQL algorithm prototype is deployed to evaluate the VM migration approach. It is divided into training phase and extraction phase. In training phase, VM migration policies are explored by the V2PQL algorithm. The cumulative reward and the temporary evolution of Q-values which show the efficiency of exploration/exploitation strategies are studied by changing the ϵ parameter of V2PQL algorithm. In extraction phase, the VM migration policies obtained in the training phase are continuously applied to the real time VM migration process . During execution, the VM migration polices showing the strength of reinforcement learning are continuously updated. The utility of players, the load balancing, the resource utilization, and the running time which show the efficiency of V2PQL algorithm are benchmarked with Round-Robin, inverse Ant System (iAS), Max–Min Ant System (MMAS), Ant System (AS) algorithms which have been evaluated in [3].

5.1 Environment setup

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. A VM migration process is triggered when the deteriorating PMs are detected. To reduce the complexity of simulations, three kinds of resources are considered in our simulations, i.e., CPU, RAM, Storage of PM, and VM configuration. The heterogeneous data-center which deploys multi-tier applications is simulated by using the parameters of Table 3. Each multi-tier application is deployed in a cluster VMs in which the configuration of VMs are randomly chosen by Table 4. The simulations are deployed on the computer (8GB RAM, Core i5, 256GB SSD) to evaluate the V2PQL performance.



(a) The average reward as function. **(b)** The discounted reward as function.

Fig. 4 The rewards of V2PQL algorithm

5.2 Training phase

VM migration policies depending on the V2PQL parameters are investigated in this phase. An optimal policy can be determined through the cumulative reward over time following actions randomly or greedy. The learning strategies helping to find appropriate parameters are considered in the Eq.(20) including ϵ, η, and γ. The exploration/exploitation strategies (cf. step 2 in V2PQL algorithm) 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 (cf. step 4 in V2PQL algorithm) is set to $\gamma = 0, 8$ like [18]. The efficiency of the V2PQL algorithm is evaluated in terms of the reward and the temporal evolution of Q-value.

An episodic task refers to a complete sequence of interaction from start to finish. The agent reaches a terminal state when the list of needed VM migration is completely processed. The V2PQL algorithm can exploit such knowledge by initializing the Q-value (cf. step 1 in the 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 of the $\epsilon = \{0.1, 0.3, 0.7, 0.9\}$ are described in the Fig. 4a. The $\epsilon = 0.1$ shows that the agent can choose among actions taken previously, otherwise, the $\epsilon = 0.9$ shows that the agent focuses on discovering new actions. The discounted cumulative rewards depict in the Fig. 4b.

The temporary evolution of Q-values refers to each state-action pairs in learning strategies. The Q-values will be updated when the system state s_t takes the action a_i. For instance, in the Fig. 5, the q(23,24) describes the temporary evolution of Q-value which the system state $s_t = 23$ and action $a_i = 24$. The almost Q-values are convergence in $episodic = 1000$.

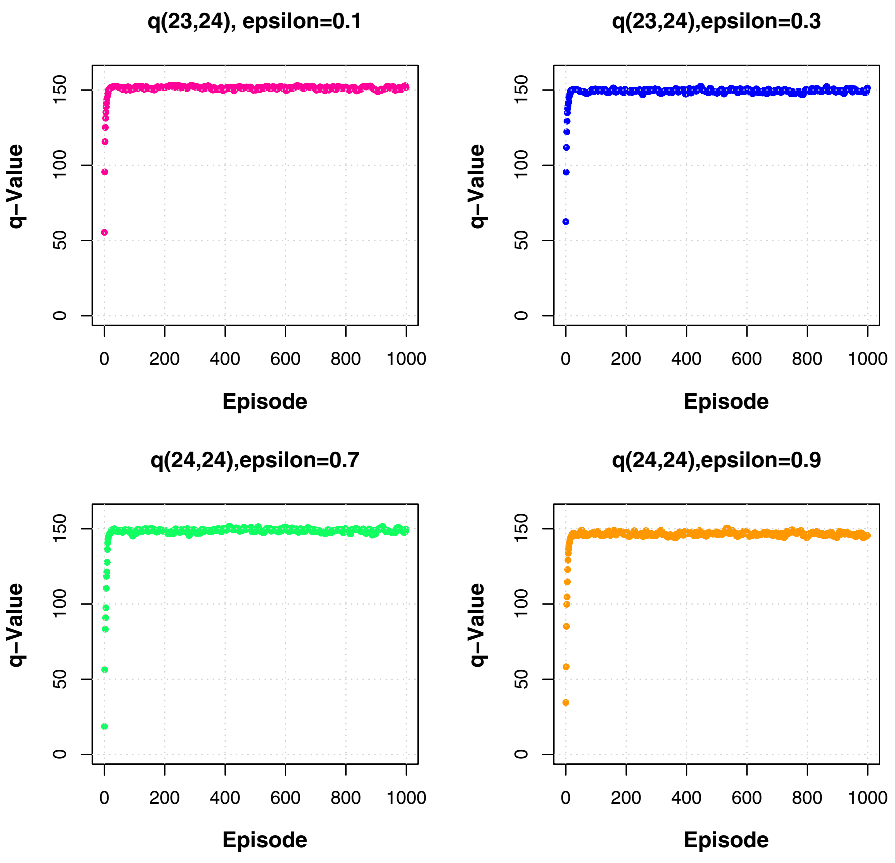


Fig. 5 The temporary evolution of Q-values

5.3 Extraction phase

The effectiveness of the V2PQL algorithm is considered in term of the running time which have been benchmarked with the RoundRobin. It shows the fast response ability of the V2PQL algorithm at run-time. As shown in the Fig. 6, the running time of the V2PQL algorithm with $\epsilon = 0.3$ is better than Round-Robin algorithm in whole VM migration while the running time of V2PQL algorithms with $\epsilon = \{0.7, 0.9\}$ is better than Round-Robin algorithm from 500th to 1543rd VMs. As the result, the VM migration of the V2PQL algorithm has a promising running time.

Furthermore, the effectiveness of the V2PQL algorithm after migrating 1543 VMs is benchmarked with RounRobin (RR), inverse Ant System (iAS), Max–Min Ant System (MMAS), and Ant System (AS) algorithms. The action extraction depends on the current system s_t and the Q-values. Choosing an optimal action from the policies for the current system state s_t is considered in term of the load balancing in the Eq.(4), the resource utilization in Eq.(5), and the utility function in Eq.(6). As shown in the Fig. 7a, the load balancing of the V2PQL algorithm is better than all of the

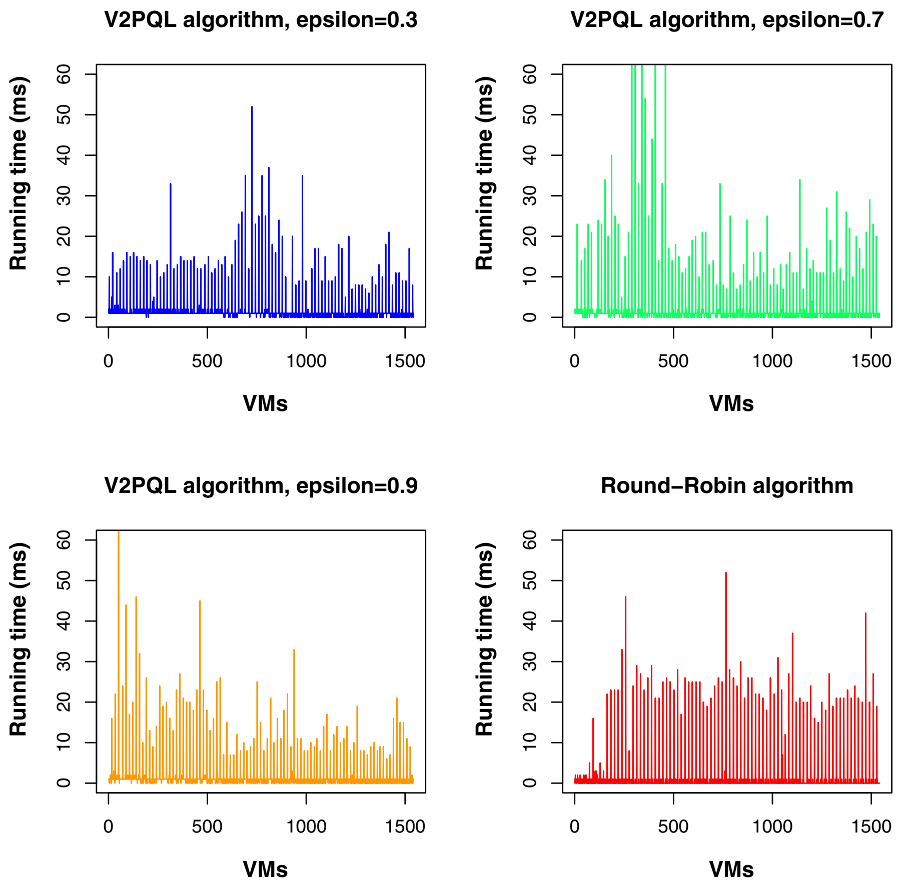


Fig. 6 The running time of V2PQL and Round-Robin algorithms

benchmarked algorithms while the resource utilization of the V2PQL algorithm is higher than all of benchmark algorithms in the Fig. 7b. As shown in the Fig. 8a, the self-learning ability of the V2PQL algorithm is presented by increasing the utility values when the system has more and more VM migrations in run-time. The utility of the V2PQL algorithm with the $\epsilon = 0.3$ is almost higher than all of the benchmarked algorithms shown in the Fig. 8b.

6 Conclusions and future work

In this paper, the V2PQL algorithm is proposed to solve the VM migration for three-tier applications of the infrastructure cloud. The conflict of stakeholders' objectives is addressed through an non-cooperate game approach proved the existence of Nash equilibrium. The V2PQL algorithm is designed through two-step approach including training and extraction phases. The VM migration policies are investigated in train-

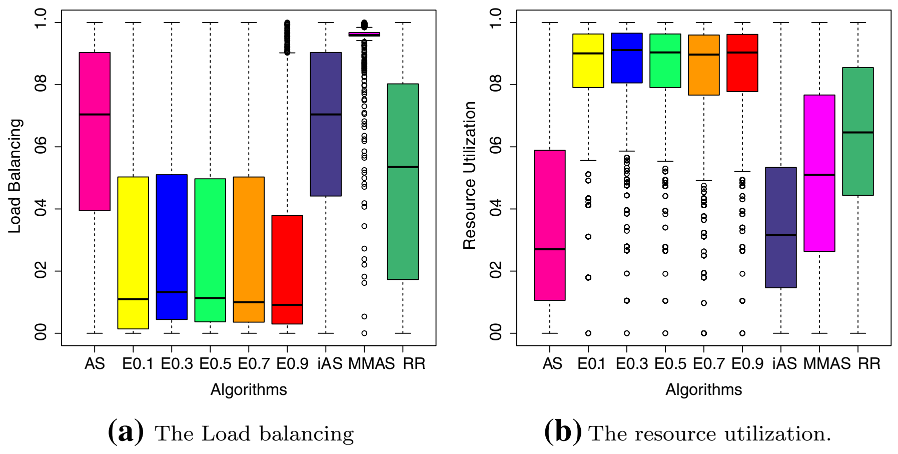


Fig. 7 The load balancing and resource utilization of the V2PQL algorithm

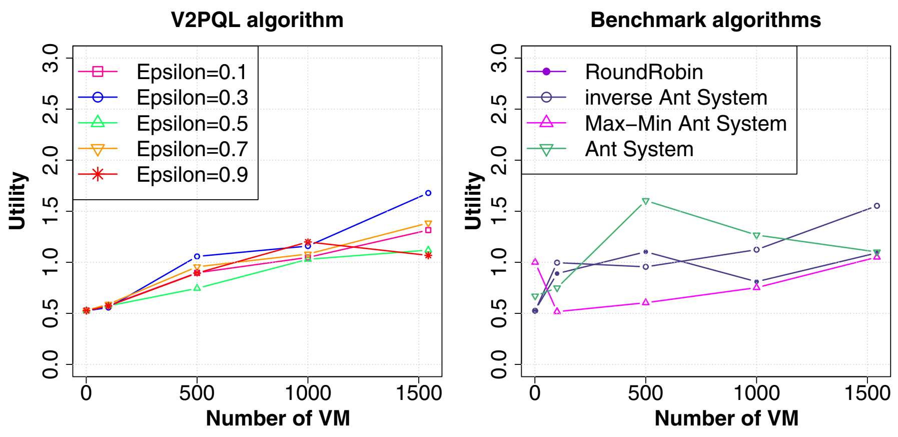


Fig. 8 The utility of V2PQL algorithm is benchmarked with RoundRobin, iAS, MMAS, AS algorithms

ing phase without prior knowledge of the infrastructure cloud. An optimal policy can be achievable with enough many data training presenting various practice situations. Investigating optimal policies depends on the control the parameters of the V2PQL algorithm. The exploration/exploitation strategies have been studied by changing the ϵ. In extraction phase, the optimal VM migration policy is simply applied by choosing the action from Q-Table at the specify system state. The V2PQL efficiency is evaluated in terms of rewards, temporal evolution of Q-values in the training phase. The V2PQL effectiveness is evaluated in term of the running time, the load balancing, the resource utilization, and the utility function which have been benchmarked with the Round-Robin, iAS, MMAS, AS algorithms. The simulation results show that the VM migration of the V2PQL algorithm has a promising running time and higher utility values of the benchmarked algorithms. In the future, many other RL algorithms will be developed to compare the evaluation with the proposed algorithm.

Acknowledgements This research is funded by Thu Dau Mot University under Grant Number DT.21.1-080. We would like to thank Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for the support of time and facilities for this study.

References

1. Bai WH, Xi JQ, Zhu JX, Huang SW (2015) Performance analysis of heterogeneous data centers in cloud computing using a complex queuing model. Math Probl Eng. https://doi.org/10.1155/2015/980945
2. Baker T, Mackay M, Randles M, Taleb-Bendiab A (2013) Intention-oriented programming support for runtime adaptive autonomic cloud-based applications. Comput Electr Eng 39(7):2400–2412. https://doi.org/10.1016/j.compeleceng.2013.04.019
3. Bui KT, Ho HD, Pham TV, Tran HC (2020) Virtual machines migration game approach for multi-tier application in infrastructure as a service cloud computing. IET Netw 9(6):326–337. https://doi.org/10.1049/iet-net.2019.0204
4. Bui KT, Nguyen LV, Tran TV, Pham TV, Tran HC (2021) 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. Springer, pp 617–628. https://doi.org/10.1007/978-981-15-7527-3_58
5. Bui KT, Pham TV, Tran HC (2016) A load balancing game approach for vm provision cloud computing based on ant colony optimization. In: International conference on context-aware systems and applications. Springer, pp 52–63. https://doi.org/10.1007/978-3-319-56357-2_6
6. Duong T, Chu YJ, Nguyen T, Chakareski J (2015) Virtual machine placement via q-learning with function approximation. In: 2015 IEEE global communications conference (GLOBECOM), pp 1–6. IEEE. https://doi.org/10.1109/GLOCOM.2015.7417491
7. Farahnakian F, Liljeberg P, Plosila J (2014) 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. IEEE, pp 500–507. https://doi.org/10.1109/PDP.2014.109
8. Ficco M, Esposito C, Palmieri F, Castiglione A (2018) A coral-reefs and game theory-based approach for optimizing elastic cloud resource allocation. Fut Gener Comput Syst 78:343–352. https://doi.org/10.1016/j.future.2016.05.025
9. Fujiwara-Greve T (1989) Learning from delayed rewards, vol 1. King's College, Cambridge
10. Fujiwara-Greve T (2015) Non-cooperative game theory, vol 1. Springer, Berlin
11. Gao Y, Guan H, Qi Z, Hou Y, Liu L (2013) A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. J Comput Syst Sci 79(8):1230–1242. https://doi.org/10.1016/j.jcss.2013.02.004
12. Ghasemi A, Toroghi Haghighat A (2020) A multi-objective load balancing algorithm for virtual machine placement in cloud data centers based on machine learning. Computing 102:2049–2072. https://doi.org/10.1007/s00607-020-00813-w
13. Ghumman NS, Kaur R (2015) 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). IEEE, pp 1–5. https://doi.org/10.1109/ICCCNT.2015.7395172
14. Guo Y, Stolyar A, Walid A (2018) Online vm auto-scaling algorithms for application hosting in a cloud. IEEE Trans Cloud Comput. https://doi.org/10.1109/TCC.2018.2830793
15. Hartmanis J (1982) Computers and intractability: a guide to the theory of np-completeness. SIAM Rev 24(1):90. https://doi.org/10.1137/1024022
16. Hsieh SY, Liu CS, Buyya R, Zomaya AY (2020) Utilization-prediction-aware virtual machine consolidation approach for energy-efficient cloud data centers. J Parallel Distrib Comput 139:99–109. https://doi.org/10.1016/j.jpdc.2019.12.014
17. Huang G, Wang S, Zhang M, Li Y, Qian Z, Chen Y, Zhang S (2016) Auto scaling virtual machines for web applications with queueing theory. In: 2016 3rd International conference on systems and informatics (ICSAI). IEEE, pp 433–438. https://doi.org/10.1109/ICSAI.2016.7810994

18. Jamshidi P, Sharifloo AM, Pahl C, Metzger A, Estrada G (2015) Self-learning cloud controllers: fuzzy q-learning for knowledge evolution. In: 2015 International conference on cloud and autonomic computing. IEEE, pp 208–211. https://doi.org/10.1109/ICCAC.2015.35
19. Levin E, Pieraccini R, Eckert W (1998) 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. IEEE, pp 201–204. https://doi.org/10.1109/ICASSP.1998.674402
20. Minarolli D, Freisleben B, (2011) Utility-based resource allocation for virtual machines in cloud computing. In: 2011 IEEE symposium on computers and communications (ISCC). IEEE, pp 410–417. https://doi.org/10.1109/ISCC.2011.5983872
21. Morton T, Pentico DW (1993) Heuristic scheduling systems: with applications to production systems and project management, vol 3. Wiley
22. Noshy M, Ibrahim A, Ali HA (2018) Optimization of live virtual machine migration in cloud computing: a survey and future directions. J Netw Comput Appl 110:1–10. https://doi.org/10.1016/j.jnca.2018.03.002
23. Rolik O, Zharikov E, Koval A, Telenyk S (2018) Dynamie management of data center resources using reinforcement learning. In: 2018 14th International conference on advanced trends in radioelecrtronics, telecommunications and computer engineering (TCSET). IEEE, pp 237–244. https://doi.org/10.1109/TCSET.2018.8336194
24. Rybina K, Schill A (2016) Estimating energy consumption during live migration of virtual machines. In: 2016 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), pp 1–5. IEEE. https://doi.org/10.1109/BlackSeaCom.2016.7901567
25. Sahni J, Vidyarthi DP (2017) Heterogeneity-aware adaptive auto-scaling heuristic for improved qos and resource usage in cloud environments. Computing 99(4):351–381. https://doi.org/10.1007/s00607-016-0530-9
26. Saovapakhiran B, Michailidis G, Devetsikiotis M, (2011) Aggregated-dag scheduling for job flow maximization in heterogeneous cloud computing. In: 2011 IEEE global telecommunications conference-GLOBECOM 2011. IEEE, pp 1–6. https://doi.org/10.1109/GLOCOM.2011.6133611
27. Siar H, Kiani K, Chronopoulos AT (2015) An effective game theoretic static load balancing applied to distributed computing. Clust Comput 18(4):1609–1623. https://doi.org/10.1007/s10586-015-0486-0
28. Tsai CW, Rodrigues JJ (2013) Metaheuristic scheduling for cloud: a survey. IEEE Syst J 8(1):279–291. https://doi.org/10.1109/JSYST.2013.2256731
29. Van Laarhoven PJ, Aarts EH, Lenstra JK (1992) Job shop scheduling by simulated annealing. Oper Res 40(1):113–125. https://doi.org/10.1287/opre.40.1.113
30. Van Otterlo M, Wiering M (2012) Reinforcement learning and markov decision processes. In: Reinforcement learning. Springer, pp 3–42. https://doi.org/10.1007/978-3-642-27645-3_1
31. Watkins CJ, Dayan P (1992) Q-learning. Mach Learn 8(3–4):279–292
32. Xiao Z, Song W, Chen Q (2012) Dynamic resource allocation using virtual machines for cloud computing environment. IEEE Trans Parallel Distrib Syst 24(6):1107–1117. https://doi.org/10.1109/TPDS.2012.283
33. Xu X, Yu H (2014) A game theory approach to fair and efficient resource allocation in cloud computing. Math Probl Eng. https://doi.org/10.1155/2014/915878
34. Yang L, Feng Y, Li K (2017) 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). IEEE, pp 80–87. https://doi.org/10.1109/ISPA/IUCC.2017.00021
35. Ye D, Chen J (2013) Non-cooperative games on multidimensional resource allocation. Fut Gener Comput Syst 29(6):1345–1352. https://doi.org/10.1016/j.future.2013.02.004
36. Zhang Q, Cheng L, Boutaba R (2010) Cloud computing: state-of-the-art and research challenges. J Internet Serv Appl 1(1):7–18. https://doi.org/10.1007/s13174-010-0007-6