Journal of High Speed Networks 27 (2021) 121–138
DOI 10.3233/JHS-210655
IOS Press

QoS and QoE aware multi objective resource allocation algorithm for cloud gaming

Koné Kigninman Désiré [a,c,*], Eya Dhib [b], Nabil Tabbane [b] and Olivier Asseu [a,c]
[a] *Ecole Supérieure Africaine des TIC (ESATIC), LASTIC, Abidjan, Côte d'Ivoire*
[b] *University of Carthage, SUP'COM, MEDIATRON Laboratory, Tunis, Tunisia*
[c] *Institut Polytechnique Felix Houphouet-Boigny (INPHB), Yamoussoukro, Côte d'Ivoire*

Abstract. Cloud gaming is an innovative model that congregates video games. The user may have different Quality-of-Experience (QoE), which is a term used to measure a user's level of satisfaction and enjoyment for a particular service. To guarantee general satisfaction for all users with limited cloud resources, it becomes a major issue in the cloud. This paper leverages a game theory in the cloud gaming model with resource optimization to discover optimal solutions to resolve resource allocation. The Rider-based harmony search algorithm (Rider-based HSA), which is the combination of Rider optimization algorithm (ROA) and Harmony search algorithm (HSA), is proposed for resource allocation to improve the cloud computing system's efficiency. The fitness function is newly devised considering certain QoE parameters, which involves fairness index, Quantified experience of players (QE), and Mean Opinion Score (MOS). The proposed Rider-based HSA showed better performance compared to Potential game-based optimization algorithm, Proactive resource allocation algorithm, QoE-aware resource allocation algorithm, Distributed algorithm, and Yusen Li *et al.*, with maximal fairness of 0.999, maximal MOS of 0.873, and maximal QE of 1.

Keywords: Cloud gaming, resource allocation, QoE, cloud computing, fairness

1. Introduction

Cloud computing has appeared as a crucial tool for processing huge-scale applications, as it can share the available resources amongst different users. The usage of optimization methods to allocate resources in the cloud model showed much interest amongst the researchers. Nowadays, a novel cloud-related service termed Cloud Gaming gained lots of attention in the gaming industries. Cloud gaming reached such popularity due to its rapid expansion in academic and engineering applications [11].

Cloud computing services adapt resource allocation techniques that had enhanced the efficiency of resource allocation strategy to use accessible resources. The resource allocator should make a tradeoff between the quality and profit of the cloud platform to become a commercial one. The overall profits generated by global video games with stream sales and online download captures and such market for gaming can be anticipated to grow multiple times in the future, reaching more than $10 billion in sales [7]. There exist several cloud gaming platforms, such as Gaikai, OnLive, StreamMyGame, and G-Cluster. Cloud gaming is drastically a different form of online gaming service wherein the process of storing, synchronizing, rendering is done in the remote cloud server. It is delivered to players by utilizing streaming technologies. Cloud gaming can also be defined as gaming on demand. Cloud gaming is popular due to 3D video games that make gaming simple at a reasonable price [2]. There is no need to install or download original gaming software for players, and even there are no requirements for players to update their hardware continuously. Cloud gaming is responsible for performing computation-intensive tasks and video streaming for users to play games. The players can react to the game by controlling the signals with mouse clicks or keystrokes to gaming servers [31].

*Corresponding author. E-mail: konekigninmand@gmail.com.

The allocation of resources is an on-demand service and may result in service loss. It can be prevented by assigning the resources to split the modules rather than splitting the complete cloud model. Thus, resource allocation comes in the class of resources management as it permits the resources to be assigned effectively. Network stream produced by cloud gaming needs a substantial network constrained bandwidth and must have latency needs such as Nvidia's GeForce NOW. There can be changes in the availability of network bandwidth over time. Therefore, some cloud gaming service utilizes adaptation algorithm that alters the parameters of video codec accordingly. The Adaptation methods utilized by the GeForce NOW service are discussed in [30]. The adaption algorithm is modeled from the per-game-flow standpoint. The scenario of the manifold game flows shared a regular network concerning bandwidth. The optimized edition amongst different flows led to enhanced utilization of resources and the allocation of accessible bandwidth reasonably, thereby increasing the QoE of concerned gamers [13]. There are various challenges for accomplishing QoE-driven cloud gaming, retrieval of information, and its optimization [26,28], and these adaptations are being discussed in many recent studies. In [31], a constrained stochastic optimization challenge is stated to minimize the total cost for cloud gaming providers by altering the selection of data center, allocation of virtual machine (VM), and video configuration for the individual user. Accomplishing inexpensive VM placement in cloud gaming servers sustain adequate QoE as devised in [12]. In [2], a resource allocation model is devised for cloud centers to focus on precise modeling of delay as QoE [29].

The design of an inexpensive platform based on gaming can offer users having elevated QoE is a big challenge. There are many problems encountered in the whole process. Firstly, in contrast to conventional service, gaming in the cloud is more cooperative and sensitive to delays. In [5], online game players are not patient and are very sensitive, creating interaction delays while playing the game. So cloud gaming service providers need to alter the provisioning of resources to solve issues related to delay. Secondly, there are various genres in-game that pose different necessities on interaction delay. Such necessities on interaction delay vary considerably amongst different game genres [27]. Thus, the choice for provisioning the resources must adapt to different game genres. Thirdly, the different types of user devices like i-pads, smartphones, tablets, and TV insist on adaptive streaming in video games. There are many varieties amongst user devices in terms of resolution of the screen, capacity computation, and network bandwidth. Therefore, it becomes necessary for the cloud gaming platform to adjust the bit-rates of video vigorously for each user and accelerate the fluctuations of network conditions and user demands. Fourth, the cost of delivering cloud gaming services plays a vital role in the cost of rent for allocating VMs, the cost of bandwidth for streaming video games, and much more. For surviving in such market conditions, the cloud gaming service provider should reduce the cost of service and offer good user QoE [31]. Several types of research concerning cloud gaming are briefly illustrated in [6,16–22,24,25]. The video gaming in education motivates the students by improving the skills [14,15].

The research aims to devise a novel cloud-based gaming model by allocating optimal resources in the cloud platforms. Here, a game theory is employed to cloud gaming to meet players' needs with less cost. Thus, the cloud gaming model with resource optimization is adapted to determine optimal solutions to address resource allocation. Here, the resource allocation is performed with the optimization method, as the optimal choice of the task provides enhanced efficiency in cloud computing. The resource is allocated using the proposed Rider-based HSA, which is designed by integrating ROA in HSA. Moreover, the fitness function is newly devised considering QE, MOS, and fairness parameters. The optimization with game theory assists the data center in easing the burden of accumulating network information and handling massive datasets.

The key contribution of the paper:

- ***Proposed Rider-based HSA for optimal resource allocation***: The proposed Rider-based HSA is employed to allocate resources optimally. The proposed Rider-based HSA is the combination of ROA in HSA such that the devised strategy aims to model optimal resources to play games on the internet.

Other sections are organized as Section 2 elaborate illustration of conventional resource allocation strategies utilized in literature and challenges faced. The system model of cloud-based gaming is depicted in Section 3. The proposed method for cloud-based resource allocation is illustrated in Section 4. The outcomes of the proposed strategy with other methods are portrayed in Section 5, and Section 6 presents the conclusion.

2. Motivation

Cloud gaming provides the potential to deliver sophisticated games to end-user devices but with elevated cost and high bandwidth. Here, the cloud is needed to devour more resources to stream videos, particularly while the count of concurrent players attains a definite level. However, the satisfaction of the QoE needs with less resource availability is the major challenge faced by cloud-based applications. Thus, the limitations of conventional resource allocation in cloud-based gaming are motivated to devise a novel cloud-based resource allocation model.

2.1. Literature review

The eight classical cloud gaming strategies based on resource allocation are devised in this section. Guo et al. [8] devised a model based on game theory in the cloud platform for meeting the complete needs of players at less cost. The game was confirmed as a potential game with an effective potential function. The model helped the players attain jointly agreeable steady states, and the model reduced the overhead and attained improved performance than other methods. The method scaled well with the count of users and can be an impending technique to optimize resources. Aslanpour et al. [1] devised a learning-based resource provisioning strategy for MMOG services devised based on autonomic computing models and learning automata (LA). The method's efficiency is based on response time allocated to virtual machines (VMs), and the cost is accessed by simulation. The method reduced the overall cost of resources with enhanced response time. However, the method failed to analyze the resource provisioning method with optimization strategies. Zhu et al. [32] addressed revenue maximization as Stackelberg's game. They evaluated the subsistence and individuality of the game equilibrium. Also, the dynamic pricing method was devised for maximizing both IaaS and SaaS. The outcomes revealed that the devised method was effective in utilizing resources and revenue maximization. However, the method failed to consider cooperation and competition amongst different IaaS providers to select the most apposite providers. Haouari et al. [10] devised a prediction-driven resource allocation model for maximizing the QoE of users and reduce the distribution of resource cost. By utilizing the viewers' location, a machine learning model was devised for predicting the viewer's number with a geo-distributed cloud site. Secondly, with the predicted outcomes, an optimization issue was formed for allocating the resources. However, the method failed to include distributed proactive resources. Slivar et al. [29] devised a method, namely QoE-aware resource allocation, for solving the optimization issue deduced by numerous cloud gaming users. The optimization issue was addressed using techniques that adapt the QoE estimation method obtained from the prejudiced studies for different kinds of games. The results confirmed that both resource providers and cloud gaming service providers adapted video coding attributes for resource allocation. However, the method failed to consider optimized resource allocation in different network links. Lee et al. [20] devised a speculative execution system, namely Outatime for mobile cloud gaming, capable of facade network latency. The Outatime offered tentative rendered frames for resource allocation. The method helped predict the future and facilitated cloud gaming providers to reach bigger communities by maintaining a high user experience level. Depasquale et al. [6] devised a cloud computing model that executed graphics-intensive programs like video games. It uses the GPU to render the graphics and stream the resulting video to the mobile device bandwidth inhibited channels. The mobile controls the user to interact with the game from remote areas. Liu et al. [25] devised two methods for improving the quality of apparent video and minimize computation complexity. Initially, the rendering-based prioritized encoding method was devised for enhancing the alleged game video quality based on the constraints of the network. Then, a macro-block level saliency map was devised to render the information. The prioritized allocation of rates was devised by dynamically adjusting the quantization parameter value for each macroblock based on the saliency mapping. Yiwen et al. [9] developed cloud gaming through the QoE-Oriented resource competition method. The distributed algorithm based on a potential gaming strategy is used to optimize the virtual machine placement in the cloud environment. This method obtained better performance efficiency. The drawback of this paper is that it is not suitable for multidimensional parameters. Yusen et al. [23] developed the resource usage reduction model using QoS in cloud gaming. In this, the brute-force algorithm is used for resource allocation in the cloud. They achieved high precision accuracy and increased resource utilization. The major disadvantage of the method is that it is not applicable to predict the interaction delay.

3. System model

Cloud gaming facilitates users with a short processing ability to play qualitative games using a high-quality internet link connection. The games can be played without installing or downloading any software of games. Moreover, there is no requirement for the user to upgrade the hardware continuously. Hence, they can play many games with less software and hardware costs. The service providers of games use disseminated data centers for presenting their services to the users. Once the cloud gaming infrastructure receives the requests from the user, then the requests of users are transmitted to a particular repository based on a specific method and assign virtual machine (VM) to execute requests of each user. Hence, the VM utilizes streaming encoded games to the user. The cloud model assigned the resources to the user tasks for a specific time to finish before the deadline. The allocator of resources offers synchronization amongst the user and cloud service provider. The resources of VM utilize various configurations with different storage, power, and memory. As resource allocation poses complete control of cloud functions, small performance degradation makes the cloud infrastructure ineffective. Thus, devising a resource allocation method is essential. Whenever the load status of the VM is in the normal state, the tasks are processed normally. As soon as the system senses an overloaded state, a resource allocation algorithm is adapted for assigning the tasks from overloaded VMs to underloaded VMs.

The model devised for allocating resources in the cloud environment is displayed in Fig. 1. The purpose of the devised model is to determine optimal resources that are effective in allocating the resources to each game demanded by the user.

Assume a cloud environment that comprises h PMs, and their representation is $M = \{M_1, M_2, \ldots, M_g, \ldots, M_h\}$; $1 \leqslant g \leqslant h$, and each PM contains multiple VMs. Consider the VMs contained in the gth PM expressed as $N = \{N_1, N_2, \ldots, N_k, \ldots, N_l\}$; $1 \leqslant k \leqslant l$, where l is total VMs in gth PM.

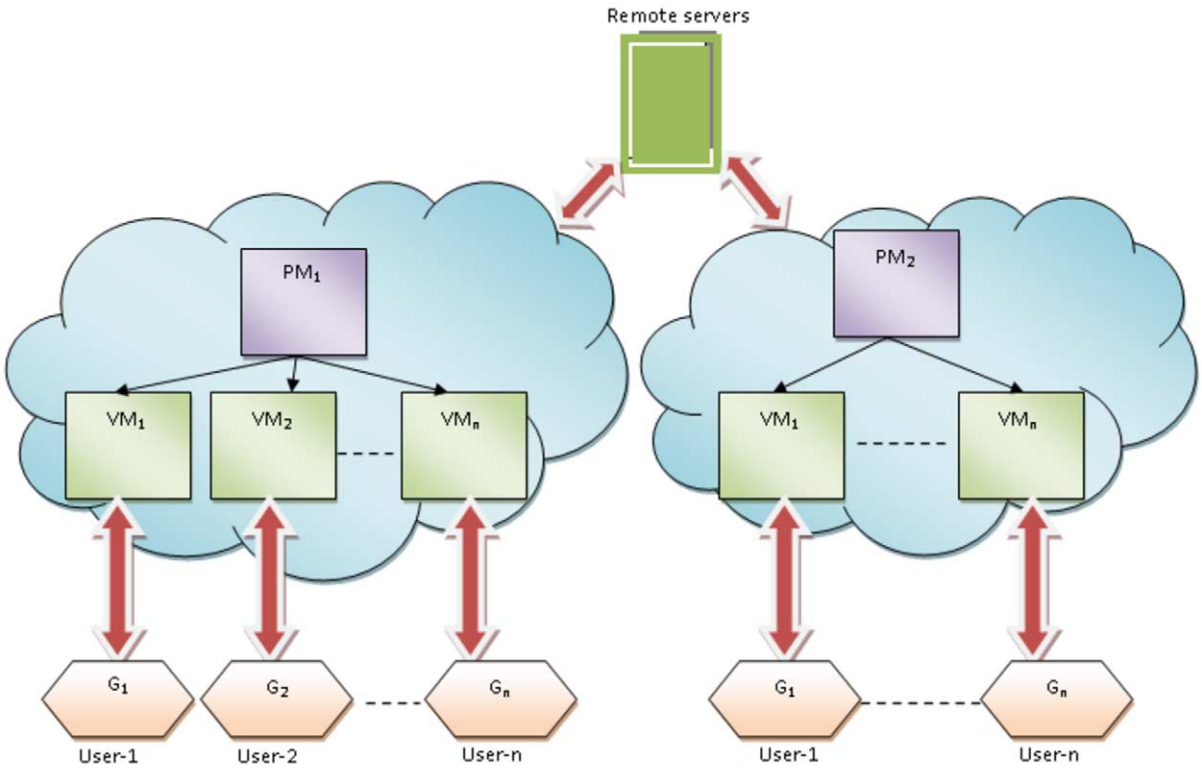

Fig. 1. The system model of cloud infrastructure.

Here, the games requested by each user are allocated to VM in a round-robin manner and are represented as $G = \{G_1, G_2, \ldots, G_p, \ldots, G_q\}$, where q are total games allocated to VM. The user willing to play the games is represented as $U = \{U_1, U_2, \ldots, U_p, \ldots, U_q\}$. In cloud gaming, the 3D video game is rendered remotely distantly from far data centers, and the screen of the game is revealed to users in real-time through internet connectivity. The video resolution is an imperative measure in judging the quality of cloud gaming. The bit rates of each frame are expressed as $B = \{B_1, B_2, \ldots, B_p, \ldots, B_q\}$. For each game, the rates of the video frame are represented as $V = \{V_1, V_2, \ldots, V_p, \ldots, V_q\}$.

The QoE of game played by history users are expressed as $Q = \{Q_1, Q_2, \ldots, Q_s, \ldots, Q_t\}$.

Each VM selected for resource allocation is configured based on some parameters, like bandwidth, processors, memory, Million Instructions Per Second (MIPS), and are expressed as,

$$N_{g,k} = \{R_{g,k}, Y_{g,k}, C_{g,k}, P_{g,k}\} \tag{1}$$

where $R_{g,k}$ represents the number of processors of kth VM in gth PM, $Y_{g,k}$ denotes memory of kth VM in gth PM, $P_{g,k}$ specifies the number of MIPS in kth VM in gth PM, $C_{g,k}$ is the bandwidth in kth VM in gth PM. The parameters $R_{g,k}$, $Y_{g,k}$, $C_{g,k}$, $P_{g,k}$ gain a value that ranges from 1 to a constant f.

The resource units of each game are expressed as,

$$K = \{K_1, K_2, \ldots, K_p, \ldots, K_q\} \tag{2}$$

The user preference level is indicated as [0, 1] wherein 1 represents more preference and 0 indicated not preferred.

Let T signifies the VM initialization time.

4. Optimization driven resource allocation using proposed Rider-based HSA algorithm

A game theory is adapted to solve allocating resources in cloud gaming and assisting service providers in minimizing maintenance costs. The inclusion of game theory is essential due to the decentralized feature of the system. The users are stressed due to fewer resources in the cloud, and hence they are automated in equally agreeable circumstances. Besides, an optimization on game theory assists the data centers to ease the burden of complicated centralized management. Moreover, the players desire to play various games and acquire an improved experience based on QoE. The game theory can be used for analyzing the competition of resources amongst diverse players with different games. Thus, the cloud gaming model with resource optimization is devised for discovering optimum solutions to resolve the resource allocation issue. Here, the allocation of resources is performed using an optimization method, as the optimal choice of task offers enhanced efficiency in cloud computing. The proposed Rider-based HSA is proposed for resource allocation to improve the efficiency of the cloud computing system. The proposed Rider-based HSA is designed by integrating ROA [3] and HSA [4] for optimal resource allocation. The goal of resource allocation with optimization is to choose optimal tasks apposite to the particular resource. The proposed Rider-based HSA is adapted for reallocating tasks by balancing loads whenever the VM is overloaded considering certain factors, including bandwidth, execution time, priority, and communication cost. Once the tasks are removed, it is added to other VMs for task execution. Besides, the convergence of the proposed algorithm is faster, and it avoids the local minima. Also, it is easy to implement with less adjustable parameters. Hence, the delay is reduced, and the efficiency of the system is improved. The fitness function is newly devised considering certain QoE parameters, which involves fairness index, Quantified experience of players (QE), Mean Opinion Score (MOS). Figure 2 portrays a block diagram of resource allocation using the proposed Rider-based HSA algorithm.

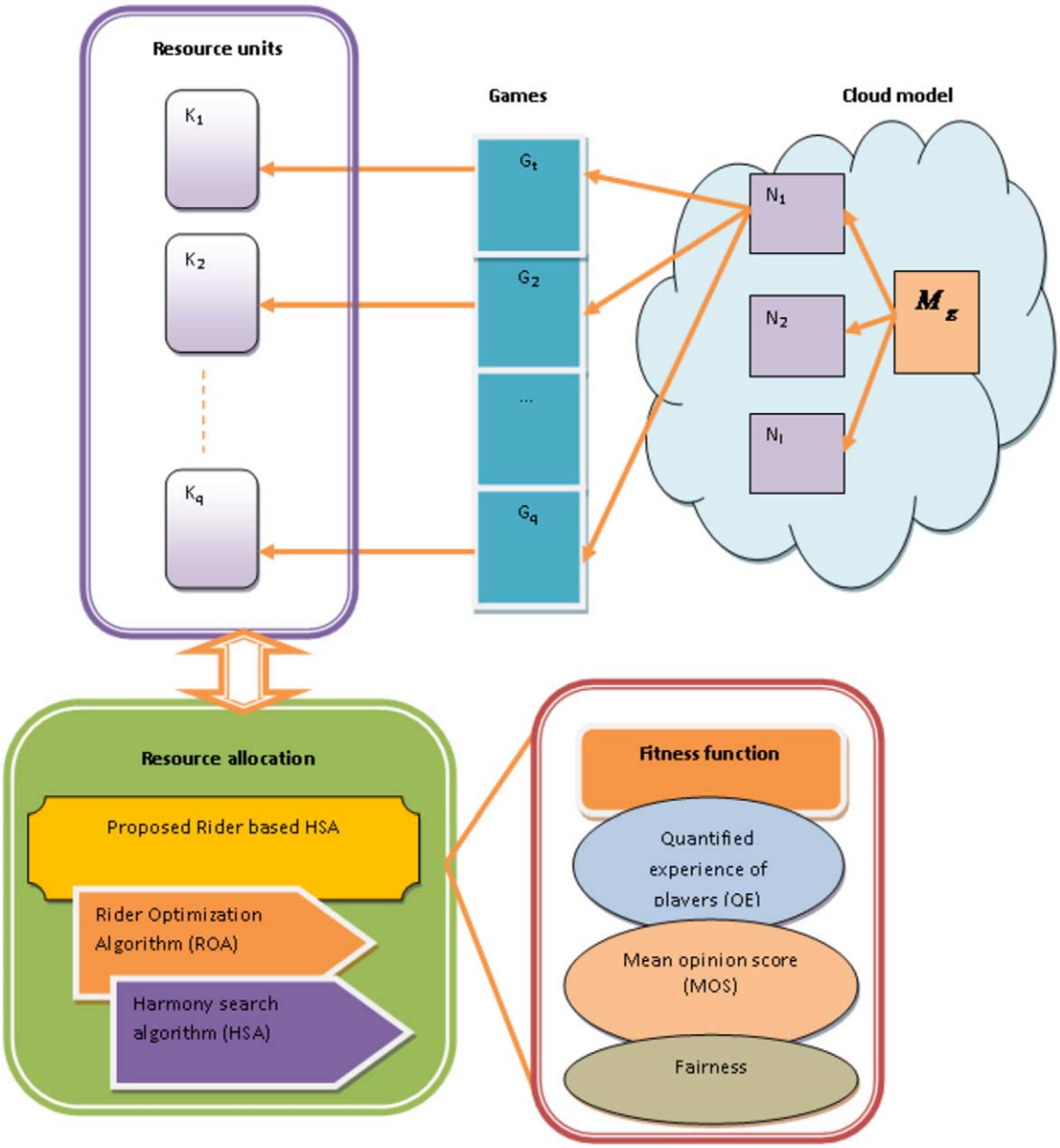

Fig. 2. Block diagram of resource allocation model using proposed Rider-based HSA.

4.1. Solution encoding

The solution representation is essential to determine the best solution for solving optimization issues. Here, the proposed Rider-based HSA algorithm is employed for selecting the appropriate solution, in which optimal games and VMs are selected from available solutions. The optimization discovers the optimum value amongst the solutions present in the solution set, which is first given as a random value. For allocating resources, the solution set contains an optimal set of games. Assume an application with seven games, as displayed in Fig. 3. Thus, the

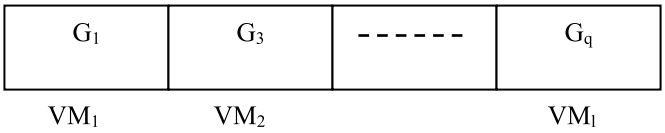

Fig. 3. Solution representation for resource allocation.

solution vector HSA games allocated at random at the initial time. Then based on fitness, the solution set acquires optimal games. These games are assigned to resources based on the newly devised fitness function.

4.2. Fitness function

The fitness is evaluated to determine the best solution using the solution set. The fitness function formulated for the proposed Rider-based HSA is based on three parameters, which involve quantified experience of players, mean opinion score (MOS), and fairness. The quantified experience of players based on the gaming experience loss is the target optimization function. The solution's fitness is considered as a maximization function so that optimal resource allocation is performed for initiating cloud gaming. The fitness function of the proposed Rider-based HSA is represented as,

$$F = QE + MOS + J^T \tag{3}$$

where, QE is the quantified experience of the players, MOS signifies mean opinion score, and J^T represents fairness. The fitness is the maximization problem for which the QE should be maximal, MOS should be maximal, and fairness should be maximal.

The mean score opinion [29] is formulated as,

$$MOS = \frac{1}{l}\sum_{k=1}^{l}(B_k * L_k + V_k * L_k) \tag{4}$$

where, B_k signifies bit rate of the game that runs in kth VM, V_k symbolizes video frame rate of the game running in kth VM, and L_k indicates the resource parameters.

The resource parameters for improving Quality of Service (QoS) are formulated as,

$$L_k = \frac{1}{4}\left(\frac{X^R + X^Y + X^C + X^P}{\max(X^R, X^Y, X^C, X^P)}\right) \tag{5}$$

where, X^R is the number of processors, X^Y is the amount of memory usage, X^C is the amount of bandwidth, and X^P is the number of MIPS utilized for resource allocation.

By integrating delay, Frame per Second, and resolution, the gaming experience loss GL [8] of the player is devised, which is the target of each player and is formulated as,

$$GL = \mu_1 U - \mu_2 W^l - \mu_2 N \tag{6}$$

where, μ_1, μ_2 and μ_3 indicate constant parameters, U signifies delay, W^l indicates experienced frames per second (FPS), and N is quality of gaming video.

The QE of the players should be maximal for the players, while the gaming experience loss should be minimal.

As the VM with games are formed and ruined dynamically, and there may subsist a clone delay amongst user p, which indicates delay in initializing service. By storing the games in the repository, the speed of writing on a hard

disk is expressed as O for ease. If a player selects a game using file size x_k, total delay [8] is expressed concerning the VM initialization time T and is given as,

$$U = \sum_{k=1}^{l} \frac{x_k}{O} + T_k \tag{7}$$

where, x_k indicate file size of the game in kth VM, O refers to writing speed of the hard disk, and T_k is the VM initialization time.

The Frame per Second (FPS) **[8]** practiced by gaming users is a key experience measure. The users practice gaming with a key experience metric, namely Frame per Second (FPS), for dealing with the cloud scenario. Generally, the FPS is discovered with GPU, RAM, and CPU considering the physical server. In cloud gaming, the FPS is formulated as,

$$W^l = \sum_{k=1}^{l} \frac{\omega_1}{1 + e^{\omega_2}[\frac{1}{5}\sum_{k=1}^{l} \vartheta_k + L_k] + \omega_3} L_k \tag{8}$$

Where, ω_1, ω_2 and ω_3 indicate parameters for approximation.

The quality of game video **[8]** is represented as,

$$N = \sum_{k=1}^{l} \left(\frac{\omega}{\upsilon} \log_2 \left(1 + \frac{\ell N_k}{\omega_o + \sum_{k=1}^{\ell} \ell N_k} \right) \right) \tag{9}$$

where, ω, ℓ and ω_o signifies constant, and N_k represent video resolution of game in jth VM.

Fairness is represented as,

$$J^T = \frac{1}{l \times t} \sum_{k=1}^{l} \left(\sum_{s=1}^{t} Q \right) * UPL \tag{10}$$

where, UPL signifies user preference level and Q be the QoE scores of games played by the history users.

4.3. Algorithmic description of the proposed Rider-based HSA algorithm

The allocation of resources is done using a novel optimization algorithm, namely the Rider-based HSA algorithm. The proposed Rider-based HSA algorithm integrates the ROA algorithm with the HSA. ROA [3] is motivated by riders' actions that pass through reach a proverbial target place to turn into a winner. Here, the riders are chosen from the total riders of each group. Each group undergoes several strategies for reaching the target. Thus, it is noted that this method performs fault diagnosis with enhanced classification accuracy. Also, the ROA is highly productive and follows steps of fictional computing to address the problems of optimization but poses less convergence and is sensitive to hyperparameters. The ROA provides high performance in unknown search spaces and can solve constrained and unconstrained problems. On the other hand, HSA [4] is a recently devised mechanism that improves music. Here, musicians improve the pitches of instruments to obtain a perfect state of harmony. The HSA algorithm provides improved convergence speed and high accuracy. Besides, the HSA provides better tradeoffs between exploitative and explorative tendencies. It is applied to solve engineering optimization issues. Thus, the hybridization of HSA and ROA is done to obtain the optimal global solution. The steps considered in the proposed Rider-based HSA algorithm is described below:

Step 1*) *Initialization of Rider and other algorithmic parameters

The algorithm's initialization is performed with four riders, namely attacker, follower, overtaker, and bypass rider, and initializations of its positions are performed arbitrarily. The initialization of group is expressed as,

$$D_n = \{D_n(u, v)\}; \quad 1 \leqslant u \leqslant H, 1 \leqslant v \leqslant I \tag{11}$$

where, H represents the number of riders, and $D_n(u, v)$ represents the position of the uth rider in vth dimension at nth time instant, and I signifies total dimension.

Step 2*) *Determination of the fitness function:

The solution's fitness is computed with equation (3) as described in Section 4.2.

Step 3*) *Discovery of leading rider:

The fitness function is considered a crucial part of discovering a leading rider. The rider residing near the target location is assumed to pose the highest fitness, and that rider is termed as the leading rider.

Step 4*) *Update position of the riders:

Each rider's position in a set is updated to discover the leading rider and thus the winner. Thus, the rider update position considering the features of each rider described in the definition. The position update of each rider as per ROA [3] is illustrated below:

The follower HSA a tendency to update position using leading rider position to reach the target more quickly and is expressed as,

$$D^f_{n+1}(u, r) = D^S(S, r) + \left[\mathrm{Cos}\left(Z^n_{u,r} * D^S(S, r) * y^r_u\right)\right] \tag{12}$$

where, r is coordinate selector, D^S indicate leading rider position, S represent leading rider index, $Z^n_{u,r}$ represent steering angle of the uth rider in rth coordinate, and y^n_u is the distance.

The over taker's update position is utilized to maximize fitness by detecting the overtaker position and is formulated as,

$$D^o_{n+1}(u, r) = D_n(u, r) + \left[A_n(u) * D^S(S, r)\right] \tag{13}$$

where, $A_n(u)$ represent the direction indicator of the uth rider at a time instant n.

The attacker poses a tendency to seize the leader's position by following the leader's update process and is given as,

$$D^a_{n+1}(u, v) = D^S(S, v) + \left[\mathrm{Cos}\, Z^n_{u,v} * D^S(S, v)\right] + y^n_u \tag{14}$$

The bypass riders pursue a familiar path without following the leading rider. In this context, the update rule of the bypass riders is exhibited in which the standard bypass rider is given as,

$$D_{n+1}(u, v) = \eta\left[D_n(\alpha, v) * \rho(v) + D_n(\gamma, v) * \left[1 - \rho(v)\right]\right] \tag{15}$$

where, η represent ransom number between 1 to H, α specifies a random number ranging between 1 to H and ρ indicate a random number between 0 and 1.

Assume $\alpha = u$, then the equation can be rewritten as,

$$D_{n+1}(u, v) = \eta\left[D_n(u, v) * \rho(v) + D_n(\gamma, v) * \left[1 - \rho(v)\right]\right] \tag{16}$$

Every unit generated considering memory is further evaluated to discover if it should be pitch-adjusted. The update of HSA [4] is given as,

$$D_{n+1}(u, v) = D_n(u, v) \pm \text{rand}(0, 1) \cdot E \tag{17}$$

Where, rand(0, 1) represent the random number in [0, 1], E signifies arbitrary distance bandwidth.

$$D_n(u, v) = D_{n+1}(u, v) \pm \text{rand}(0, 1) \cdot E \tag{18}$$

The bypass riders undergo a familiar path without following the leading rider. HSA enhances the update position of the bypass rider. Thus, the substitution of equation (18) is done in equation (16), and the resultant equation is expressed as,

$$D_{n+1}(u, v) = \eta\Big[\big(D_{n+1}(u, v) \pm \text{rand}(0, 1) \cdot E\big) * \rho(v) + D_n(\gamma, v) * \big[1 - \rho(v)\big]\Big], \tag{19}$$

$$D_{n+1}(u, v) = \eta D_{n+1}(u, v) * \rho(v) + \eta\Big[\big(D_n(\gamma, v) * \big[1 - \rho(v)\big] \pm \text{rand}(0, 1) \cdot E\big) * \rho(v)\Big], \tag{20}$$

$$D_{n+1}(u, v) - \eta D_{n+1}(u, v) * \rho(v) = \eta\Big[\big(D_n(\gamma, v) * \big[1 - \rho(v)\big] \pm \text{rand}(0, 1) \cdot E\big) * \rho(v)\Big], \tag{21}$$

$$D_{n+1}(u, v)\big[1 - \eta\rho(v)\big] = \eta\Big[\big(D_n(\gamma, v) * \big[1 - \rho(v)\big] \pm \text{rand}(0, 1) \cdot E\big) * \rho(v)\Big], \tag{22}$$

$$D_{n+1}(u, v) = \frac{\eta[(D_n(\gamma, v) * [1 - \rho(v)] \pm \text{rand}(0, 1) \cdot E) * \rho(v)]}{1 - \eta\rho(v)} \tag{23}$$

Step 5) Determination of the success rate:

After completing the update, the fitness of the individual rider is evaluated. Thus, the rider's position that is leading state is substituted with the new position rider obtained so far in such a way that the fitness of the new rider is higher.

Step 6) Off time:

The steps are iterated continuously till time attained off time N_{OFF}, wherein the leading rider is discovered. After completion of the race, the leading rider is termed as the winner.

The pseudo-code of the proposed Rider-based HSA algorithm is illustrated in Table 1.

5. Results and discussions

This section describes the effectiveness of the proposed Rider-based HSA using Fairness, MOS, and QE. The analysis is done considering different games size like 100, 200, and 300. The convergence analysis is also done for the proposed methodology.

5.1. Experimental set-up

The proposed strategy is implemented in PYTHON with a PC with Windows 10 OS, 4GB RAM, and Intel i3 core processor. The number of PMs used is 6, the number of VM is equal to the task size, the population size is 5 and, the iteration used is 50.

5.2. Performance measures

The analysis of the methods is performed using Fairness, MOS, and QE parameters and is elaborated briefly in Section 4.2.

Table 1
Pseudo-code of proposed Rider-based HSA algorithm

Input: D_n: Random position of rider, n: iteration, $n_{\max}$: maximum iteration
Output: Leading rider D^S
Begin
  Initialize the set of solutions.
  Initialize other parameter of rider like gear, brake, accelerator and steering angle
  Determine fitness function using equation (3)
While $n < N_{\text{OFF}}$
For $u = 1$ *to* H
  Update bypass rider's position with equation (23)
  Update follower position with equation (12)
  Update overtaker position with equation (13)
  Update attacker position with equation (14)
  Rank riders using fitness function with equation (3)
  Choose the rider with high fitness function
Update gear, brake, accelerator and steering angle
Return D^S
$n = n + 1$
End for
End while
End

5.3. Comparative methods

The techniques adopted for the evaluation involve the Potential game-based optimization algorithm [8], Proactive resource allocation algorithm [10], QoE-aware resource allocation algorithm [29], Distributed algorithm [9], Yusen Li et al. [23], and Proposed Rider-based HSA.

5.4. Comparative analysis

The analysis of methods is done using Fairness, MOS, and QE parameters by varying the number of iterations. The analysis is done considering three games, size 100, 200, and 300, respectively.

a) **Analysis with game size 100**

Figure 4 portrays the analysis of methods with game size 100 using Fairness, MOS, and QE parameters. The analysis of methods considering the parameter Fairness is portrayed in Fig. 4a). When the number of iterations is 10, the fairness values evaluated by Potential game-based optimization algorithm, Proactive resource allocation algorithm, Yusen Li *et al.*, Distributed algorithm, QoE-aware resource allocation algorithm, and Proposed Rider-based HSA are 0.106, 0.138, 0.138, 0.138, 0.138, and 0.148. Similarly, when the number of iterations is 50, the fairness values evaluated by Potential game-based optimization algorithm, Proactive resource allocation algorithm, Yusen Li *et al.*, Distributed algorithm, QoE-aware resource allocation algorithm, and Proposed Rider-based HSA are 0.135, 0.155, 0.156, 0.156, 0.156, and 0.177. The analysis of methods considering the MOS parameter is portrayed in Fig. 4b). When the number of iterations is 10, the MOS values evaluated using the potential game-based optimization algorithm, the Proactive resource allocation algorithm, Yusen Li et al., Distributed algorithm, and a QoE-aware resource allocation algorithm, and Proposed Rider-based HSA are 0.759, 0.781, 0.784, 0.789, 0.792, and 0.793. Similarly, when the number of iterations is 50, the MOS values evaluated by Potential game-based optimization algorithm, Proactive resource allocation algorithm, Yusen Li et al., Distributed algorithm, QoE-aware resource allocation algorithm, and Proposed Rider-based HSA are 0.889, 0.916, 0.917, 0.917, 0.917, and 0.921.

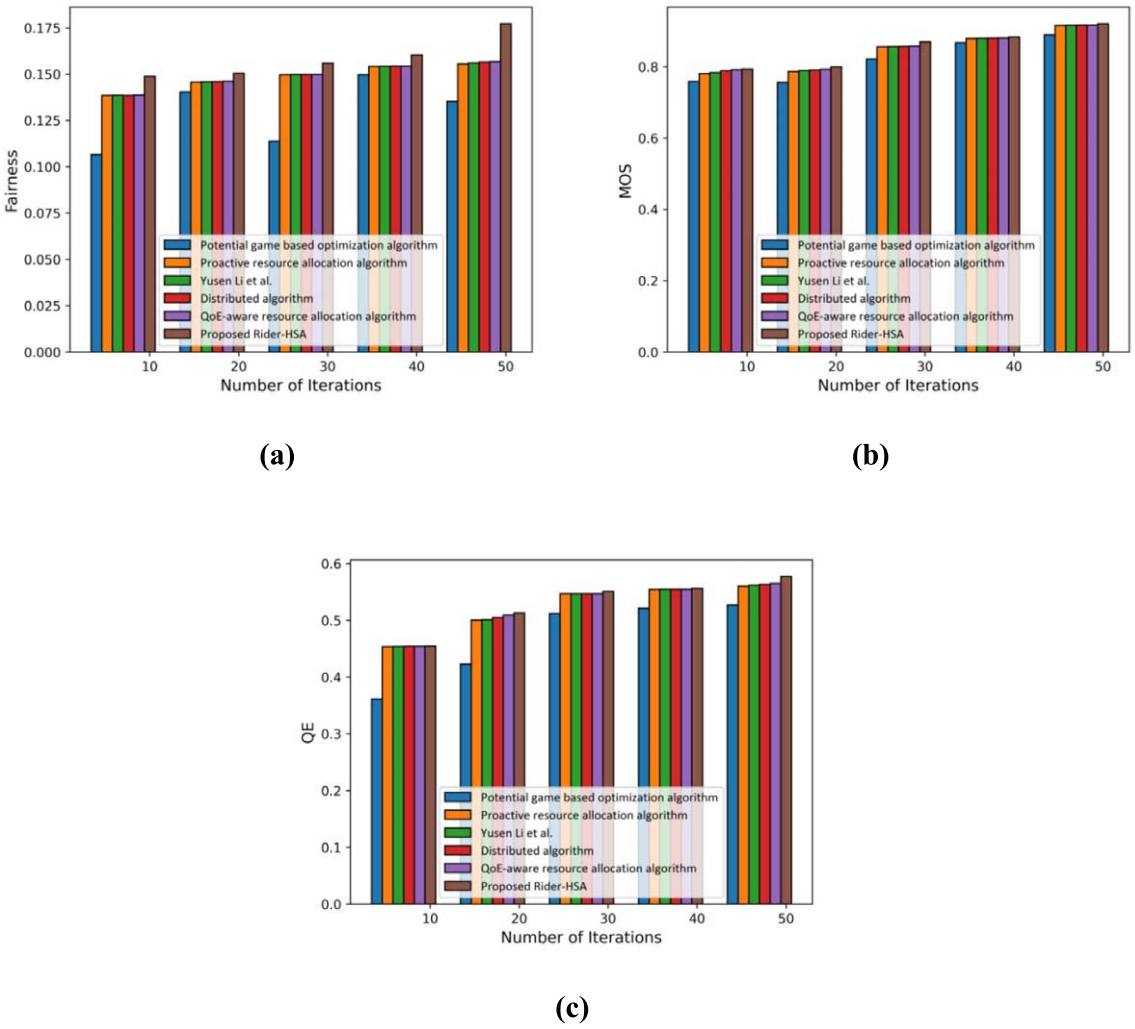

Fig. 4. Analysis of methods with game size 100 using a) Fairness, b) MOS, c) QE.

The analysis of methods considering the parameter QE is portrayed in Fig. 4c). When the number of iterations is 10, the QE values are evaluated by the Potential game-based optimization algorithm, Proactive resource allocation algorithm, Yusen Li *et al.*, Distributed algorithm, QoE-aware resource allocation algorithm, and Proposed Rider-based HSA are 0.361, 0.453, 0.454, 0.454, 0.454, and 0.455. Similarly, when the number of iterations is 50, the QE values evaluated by Potential game-based optimization algorithm, Proactive resource allocation algorithm, Yusen Li et al., Distributed algorithm, QoE-aware resource allocation algorithm, and Proposed Rider-based HSA are 0.526, 0.560, 0.561, 0.563, 0.565, and 0.578.

b) **Analysis with game size 200**

Figure 5 portrays the analysis of methods with game size 200 using Fairness, MOS, and QE parameters. The analysis of methods considering the parameter Fairness is portrayed in Fig. 4a). When the number of iterations is 10, the fairness values evaluated by the Potential game-based optimization algorithm, the Proactive resource allocation algorithm, Yusen Li *et al.*, Distributed algorithm, QoE-aware resource allocation algorithm, and Proposed Rider-based HSA are 0.230, 0.440, 0.442, 0.444, 0.447, and 0.461. Similarly, when the number of iterations is 50,

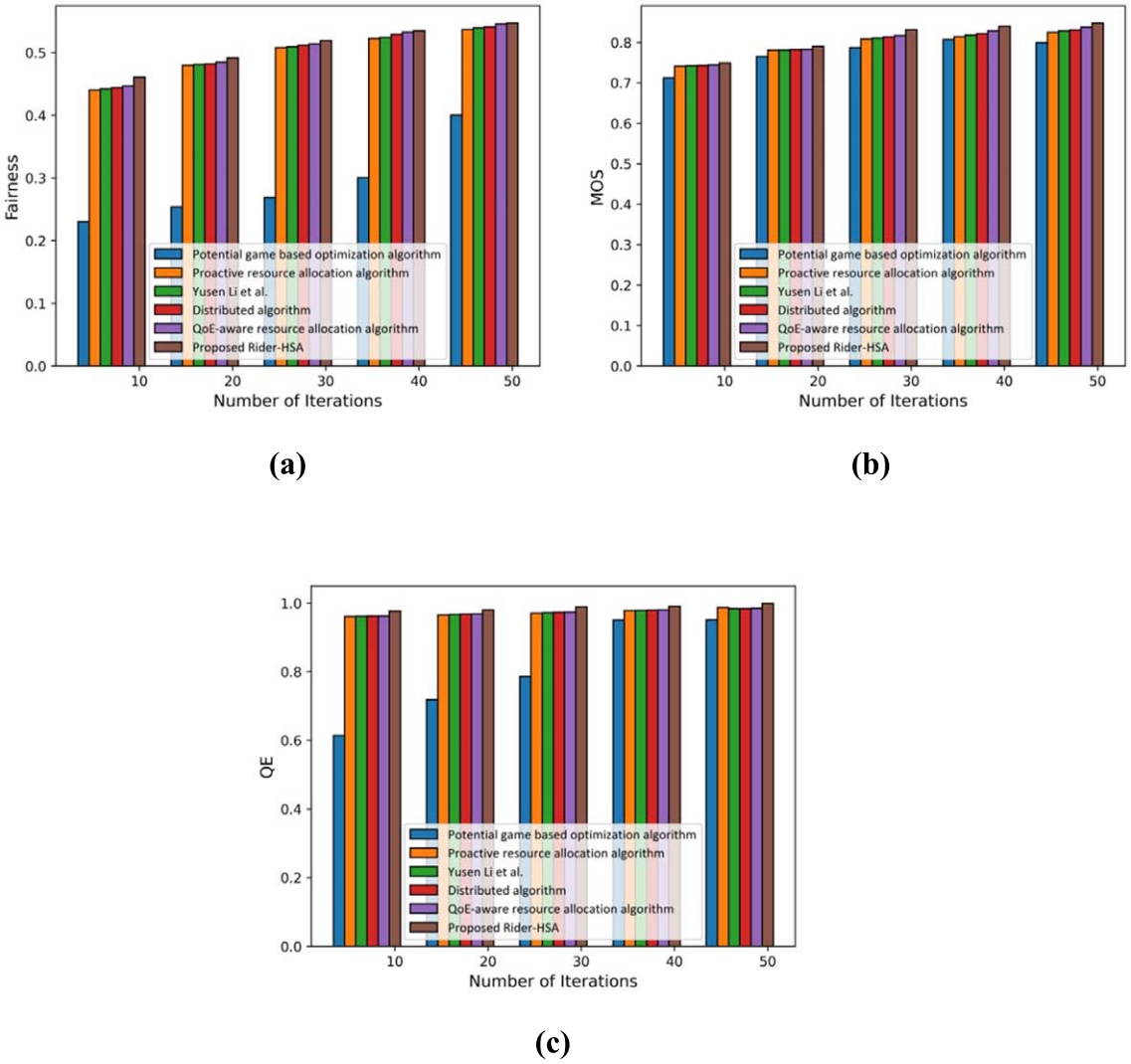

Fig. 5. Analysis of methods with game size 200 using a) Fairness, b) MOS, c) QE.

the fairness values measured by Potential game-based optimization algorithm, Proactive resource allocation algorithm, Yusen Li et al., Distributed algorithm, QoE-aware resource allocation algorithm, and Proposed Rider-based HSA are 0.401, 0.537, 0.539, 0.541, 0.545, and 0.547. The analysis of methods considering MOS is portrayed in Fig. 5b). When the number of iterations is 10, the MOS values estimated by the Potential game-based optimization algorithm, Proactive resource allocation algorithm, Yusen Li et al., Distributed algorithm, QoE-aware resource allocation algorithm, and Proposed Rider-based HSA are 0.712, 0.741, 0.742, 0.743, 0.745, and 0.750. Likewise, when the number of iterations is 50, the MOS values measured by Potential game-based optimization algorithm, Proactive resource allocation algorithm, Yusen Li et al., Distributed algorithm, QoE-aware resource allocation algorithm, and Proposed Rider-based HSA are 0.80, 0.824, 0.829, 0.831, 0.838, and 0.848. The analysis of methods considering the parameter QE is portrayed in Fig. 5c). When the number of iterations is 10, the QE values are measured by the Potential game-based optimization algorithm, Proactive resource allocation algorithm, Yusen Li et al., Distributed algorithm, QoE-aware resource allocation algorithm, and Proposed Rider-based HSA are 0.614, 0.961, 0.962, 0.962, 0.963, and 0.977. Likewise, when the number of iterations is 50, the QE values

measured by Potential game-based optimization algorithm, Proactive resource allocation algorithm, Yusen Li et al., Distributed algorithm, QoE-aware resource allocation algorithm, and Proposed Rider-based HSA are 0.951, 0.988, 0.984, 0.984, 0.985, and 0.999.

c) **Analysis with game size 300**

Figure 6 portrays the analysis of methods with game size 300 using Fairness, MOS, and QE parameters. The analysis of methods considering the parameter Fairness is portrayed in Fig. 6a). When the number of iterations is 10, the fairness values measured by the Potential game-based optimization algorithm, Proactive resource allocation algorithm, Yusen Li et al., Distributed algorithm, QoE-aware resource allocation algorithm, and Proposed Rider-based HSA are 0.416, 0.846, 0.872, 0.894, 0.921, and 0.955. Similarly, when the number of iterations is 50, the fairness values measured by Potential game-based optimization algorithm, Proactive resource allocation algorithm, Yusen Li et al., Distributed algorithm, QoE-aware resource allocation algorithm, and Proposed Rider-based HSA are 0.538, 0.977, 0.980, 0.983, 0.990, and 0.999. The analysis of methods considering the MOS parameter is portrayed in Fig. 6b). When the number of iterations is 10, the MOS values measured by the Potential game-based

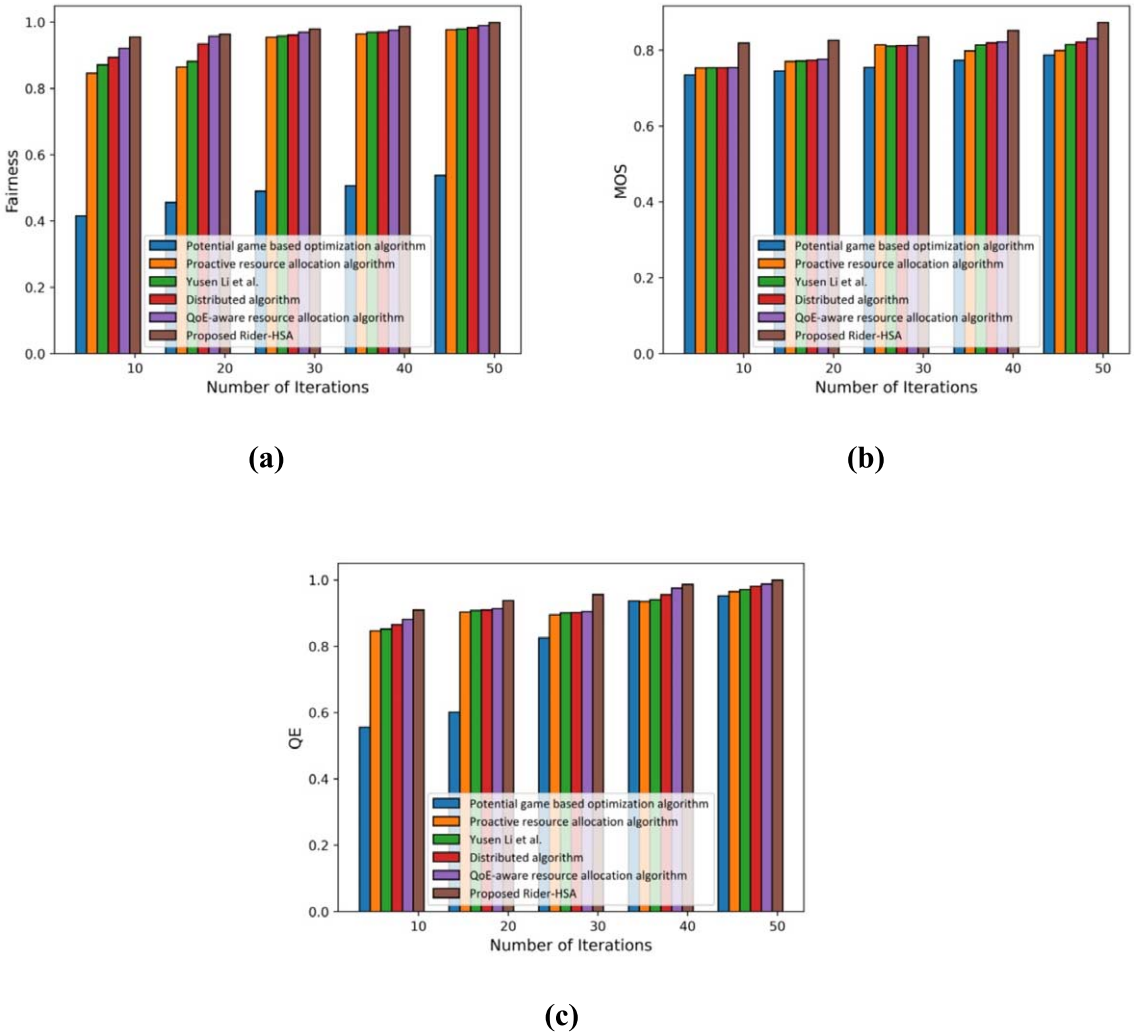

Fig. 6. Analysis of methods with game size 300 using a) Fairness, b) MOS, c) QE.

optimization algorithm, Proactive resource allocation algorithm, Yusen Li et al., Distributed algorithm, QoE-aware resource allocation algorithm, and Proposed Rider-based HSA are 0.735, 0.753, 0.754, 0.754, 0.754, and 0.818. Similarly, when the number of iterations is 50, the MOS values measured by Potential game-based optimization algorithm, Proactive resource allocation algorithm, Yusen Li et al., Distributed algorithm, QoE-aware resource allocation algorithm, and Proposed Rider-based HSA are 0.788, 0.799, 0.815, 0.821, 0.83, and 0.873. The analysis of methods considering the QE parameter is portrayed in Fig. 6c). When the number of iterations is 10, the QE values measured by the Potential game-based optimization algorithm, Proactive resource allocation algorithm, Yusen Li et al., Distributed algorithm, QoE-aware resource allocation algorithm, and Proposed Rider-based HSA are 0.556, 0.847, 0.852, 0.865, 0.881, and 0.91. Similarly, when the number of iterations is 50, the QE values measured by Potential game-based optimization algorithm, Proactive resource allocation algorithm, Yusen Li et al., Distributed algorithm, QoE-aware resource allocation algorithm, and Proposed Rider-based HSA are 0.952, 0.965, 0.971, 0.981, 0.987, and 1.

5.5. Analysis based on convergence

Figure 7 portrays the convergence analysis of methods with game sizes 100, 200, and 300. The analysis of convergence by considering game size 100 is portrayed in Fig. 7a). When the number of iterations is 10, the fitness values measured by the Potential game-based optimization algorithm, Proactive resource allocation algorithm, Yusen Li et al., Distributed algorithm, QoE-aware resource allocation algorithm, and Proposed Rider-based HSA are 5326.509, 5340.325, 5342.521, 5289.075, 5342.250, and 5191.308. Similarly, when the number of iterations is 50, the fitness values measured by Potential game-based optimization algorithm, Proactive resource allocation

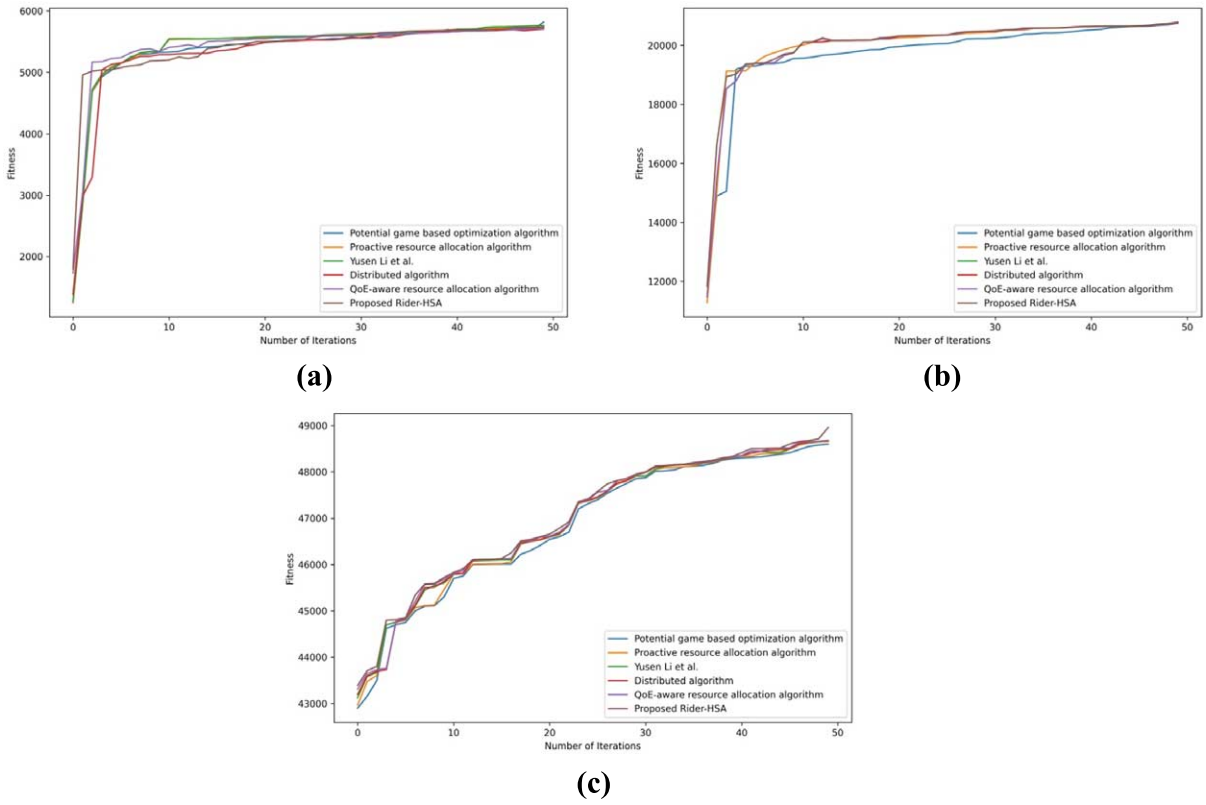

Fig. 7. Convergence graph. a) Game size 100, b) game size 200, and c) game size 300.

algorithm, Yusen Li et al., Distributed algorithm, QoE-aware resource allocation algorithm, and Proposed Rider-based HSA are 5820.160, 5741.236, 5763.874, 5706.295, 5716.335, and 5735.412. The convergence analysis by game size 200 is portrayed in Fig. 7b). When the number of iterations is 10, the fitness values measured by the Potential game-based optimization algorithm, Proactive resource allocation algorithm, Yusen Li *et al.*, Distributed algorithm, QoE-aware resource allocation algorithm, and Proposed Rider-based HSA are 19552.850, 19950.152, 19742.654, 19749.215, 19751.125, and 19754.451. Similarly, when the number of iterations is 50, the fitness values measured by Potential game-based optimization algorithm, Proactive resource allocation algorithm, Yusen Li et al., Distributed algorithm, QoE-aware resource allocation algorithm, and Proposed Rider-based HSA are 20748.991, 20743.512, 20750.654, 20752.615, 20764.618, and 20789.215. The analysis by considering the game size 300 is portrayed in Fig. 7c). When the number of iterations is 10, the fitness values measured by the Potential game-based optimization algorithm, Proactive resource allocation algorithm, Yusen Li et al., Distributed algorithm, QoE-aware resource allocation algorithm, and Proposed Rider-based HSA are 045300.412, 45458.891, 45600.155, 45635.412, 45680.412, and 45720.652. Similarly, when the number of iterations is 50, the fitness values measured by Potential game-based optimization algorithm, Proactive resource allocation algorithm, Yusen Li et al., Distributed algorithm, QoE-aware resource allocation algorithm, and Proposed Rider-based HSA are 48600.412, 48647.132, 48670.512, 48673.536, 48681.158, and 48954.541.

5.6. Comparative discussion

Table 2 illustrates the analysis of methods considering fairness, MOS, and QE using game sizes 100, 200, and 300, respectively. Considering game size of 100, the performance improvement based on fairness using proposed Rider-based HSA with QoE-aware resource allocation algorithm is 11.299%, with the distributed algorithm is 11.29%, with Yusen Li et al., is 11.86%, with Proactive resource allocation algorithm is 11.864% and with Potential game-based optimization algorithm is 23.728%. The MOS computed by the proposed Rider-based HSA is 0.921. In contrast, the MOS of the Potential game-based optimization algorithm, Proactive resource allocation algorithm, QoE-aware resource allocation algorithm, Yusen Li et al., and Distributed algorithm are 0.890, 0.916, 0.918, 0.917, and 0.917. The QE computed by the proposed Rider-based HSA is 0.578. In contrast, the MOS of the Potential game-based optimization algorithm, Proactive resource allocation algorithm, QoE-aware resource allocation algorithm, Yusen Li et al., and Distributed algorithm are 0.527, 0.560, 0.565, 0.562, and 0.564. Considering game size = 200, the proposed Rider-based HSA measured maximal fairness of 0.547, maximal MOS of 0.848, and maximal QE of 0.999. Considering game size = 300, the proposed Rider-based HSA measured maximal fairness of 0.999, maximal MOS of 0.873, and maximal QE of 1. Through analysis, it is noted that the proposed Rider-based HSA offers superior performance in allocating resources with cloud infrastructures with maximal fairness of 0.999, maximal MOS of 0.873, and maximal QE of 1.

Table 2
Comparative analysis

Game size	Metrics	Potential game based optimization algorithm	Proactive resource allocation algorithm	Yusen Li et al.	Distributed algorithm	QoE-aware resource allocation algorithm	Proposed Rider-based HSA
100	Fairness	0.135	0.156	0.156	0.157	0.157	0.177
	MOS	0.890	0.916	0.917	0.917	0.918	0.921
	QE	0.527	0.560	0.562	0.564	0.565	0.578
200	Fairness	0.401	0.537	0.540	0.541	0.546	0.547
	MOS	0.800	0.825	0.829	0.831	0.838	0.848
	QE	0.951	0.988	0.984	0.984	0.985	0.999
300	Fairness	0.538	0.978	0.980	0.984	0.990	0.999
	MOS	0.788	0.799	0.815	0.822	0.831	0.873
	QE	0.952	0.965	0.971	0.981	0.988	1.000

6. Conclusion

This paper devises a novel cloud-based gaming model by allocating optimal resources in the cloud platforms to satisfy players' desires with less cost. Here, the cloud gaming model with resource optimization is devised to determine optimal solutions to address resource allocation. Thus, the Rider-based HSA is proposed for resource allocation to improve the efficiency of the cloud computing system. The proposed Rider-based HSA is designed by integrating ROA and HSA for optimal resource allocation. The fitness function is newly devised considering certain QoE parameters, including fairness index, Experience rating, and MOS. The proposed Rider-based HSA in game theory helps data centers alleviate the burden of accumulating user's network information and managing massive datasets. The experimentation of the proposed Rider-based HSA is performed using fairness, MOS, and QE, considering different game sizes. The analysis discovered that the performance of the proposed Rider-based HSA is superior as compared to other methods with maximal fairness of 0999, maximal MOS of 0.873, and maximal QE of 1, respectively. In the future, the mutual consideration of optimal allocation of resources in network links can be considered.

References

[1] M.S. Aslanpour, M. Ghobaei-Arani, M. Heydari and N. Mahmoudi, LARPA: A learning automata-based resource provisioning approach for massively multiplayer online games in cloud environments, *International Journal of Communication Systems* (2019), e4090.

[2] M. Basiri and A. Rasoolzadegan, Delay-aware resource provisioning for cost-efficient cloud gaming, *IEEE Transactions on Circuits and Systems for Video Technology* **28**(4) (2018), 972–983. doi:10.1109/TCSVT.2016.2632121.

[3] D. Binu and B.S. Kariyappa, RideNN: A new rider optimization algorithm-based neural network for fault diagnosis in analog circuits, *IEEE Transactions on Instrumentation and Measurement* **68**(1) (2018), 2–26. doi:10.1109/TIM.2018.2836058.

[4] P. Chakraborty, G.G. Roy, S. Das, D. Jain and A. Abraham, An improved harmony search algorithm with differential mutation operator, *Fundamenta Informaticae* **95**(4) (2009), 401–426. doi:10.3233/FI-2009-157.

[5] C. Chambers, W.C. Feng, S. Sahu, D. Saha and D. Brandt, Characterizing online games, *IEEE/ACM Transactions on Networking (TON)* **18**(3) (2010), 899–910. doi:10.1109/TNET.2009.2034371.

[6] E. Depasquale, A. Zammit, M. Camilleri, S. Zammit, A. Muscat, P. Mallia and S. Scerri, An analytical method of assessment of RemoteFX as a cloud gaming platform, in: *Proc. of IEEE Conference on Mediterranean Electro Technical Conference (MELECON'14)*, Beirut, Lebanon, 2014, pp. 127–133.

[7] Distribution and monetization strategies to increase revenues from cloud gaming, 2012, [Online]. Available at http://www.cgconfusa.com/report/documents/Content-5minCloudGamingReportHighlights.pdf.

[8] D. Guo, Y. Han, W. Cai, X. Wang and V.C.M. Leung, QoE-oriented resource optimization for mobile cloud gaming: A potential game approach, in: *ICC 2019 – 2019 IEEE International Conference on Communications (ICC)*, 2019.

[9] Y. Han, D. Guo, W. Cai, X. Wang and V.C.M. Leung, Virtual machine placement optimization in mobile cloud gaming through QoE-oriented resource competition, in: *IEEE Transactions*, 2020.

[10] F. Haouari, E. Baccour, A. Erbad, A. Mohamed and M. Guizani, QoE-aware resource allocation for crowdsourced live streaming: A machine learning approach, in: *International Conference on Communications (ICC)*, 2019, pp. 1–6.

[11] M.M. Hassan, M.S. Hossain, A.M.J. Sarkar and E.-N. Huh, Cooperative game-based distributed resource allocation in horizontal dynamic cloud federation platform, *Information Systems Frontiers* **16**(4) (2012), 523–542. doi:10.1007/s10796-012-9357-x.

[12] H. Hong, D. Chen, C. Huang, K. Chen and C. Hsu, Placing virtual machines to optimize cloud gaming experience, *IEEE Transactions on Cloud Computing* **3**(1) (2015), 42–53. doi:10.1109/TCC.2014.2338295.

[13] H.-J. Hong, C.F. Hsu, T.H. Tsai, C.Y. Huang, K.T. Chen and C.H. Hsu, Enabling adaptive cloud gaming in an open-source cloud gaming platform, *IEEE transactions on Circuits and Systems for Video Technology* **25**(12) (2015), 2078–2091.

[14] N.T. Hung, A model of international students' choice: A mixed-methods study, in: *International Virtual Conference on Public Administration, Social Science & Humanities*, 2020.

[15] N.T. Hung, The motivation of Vietnamese students to study in Taiwan under the background of the new southbound policy, *School Administrators* **131** (2021), 157–185.

[16] K. Kim, S. Bae, D. Lee, C. Cho, H. Lee and K. Lee, Cloud-based gaming service platform supporting multiple devices, *ETRI Journal* **35**(6) (2013), 960–968. doi:10.4218/etrij.13.2013.0076.

[17] S. Kim, K. Kim and J. Won, Multi-view rendering approach for cloud-based gaming services, in: *Proc. of International Conference on Advances in Future Internet (AFIN'11)*, French Riviera, France, 2011, pp. 102–107.

[18] U. Lampe, Q. Wu, S. Dargutev, R. Hans, A. Miede and R. Steinmetz, Assessing latency in cloud gaming, in: *Proc. of International Conference on Cloud Computing and Services Science (CLOSER'14)*, Barcelona, Spain, 2014, pp. 52–68.

[19] J. Laulajainen, T. Sutinen and S. Järvinen, Experiments with QoS aware gaming-on-demand service, in: *Proc. of International Conference on Advanced Information Networking and Applications (AINA'06)*, Vienna, Austria, 2006, pp. 805–810.
[20] K. Lee, D. Chu, E. Cuervo, J. Kopf, S. Grizan, A. Wolman and J. Flinn, Out a time: Using speculation to enable low-latency continuous interaction for cloud gaming, in: *Proc. of Annual International Conference on Mobile Systems, Applications, and Services (MobiSys'15)*, Florence, Italy, 2015, pp. 151–165. doi:10.1145/2742647.2742656.
[21] K. Lee, D. Chu, E. Cuervo, A. Wolman and J. Flinn, Demo: Delorean: Using speculation to enable low-latency continuous interaction for mobile cloud gaming, in: *Proc. of Annual International Conference on Mobile Systems, Applications, and Services (MobiSys'15)*, Florence, Italy, 2015, p. 347.
[22] Y. Li, X. Tang and W. Cai, Play request dispatching for efficient virtual machine usage in cloud gaming, *IEEE Transactions on Circuits and Systems for Video Technology* **25**(12) (2015), 2052–2063. doi:10.1109/TCSVT.2015.2450152.
[23] Y. Li, C. Zhao, X. Tang, W. Cai, X. Liu, G. Wang and X. Gong, Towards minimizing resource usage with QoS guarantee in cloud gaming, *IEEE Transactions on Parallel and Distributed Systems* **32**(2) (2021).
[24] L. Lin, X. Liao, G. Tan, H. Jin, X. Yang, W. Zhang and B. Li, Live render: A cloud gaming system based on compressed graphics streaming, in: *Proc. of ACM International Conference on Multimedia (MM'14)*, Orlando, FL, 2014, pp. 347–356. doi:10.1145/2647868.2654943.
[25] Y. Liu, S. Dey and Y. Lu, Enhancing video encoding for cloud gaming using rendering information, *IEEE Transactions on Circuits and Systems for Video Technology* **25**(12) (2015), 1960–1974. doi:10.1109/TCSVT.2015.2450175.
[26] M.A. Mousa, H.B. Li and H.B. Abdalla, Optimization driven Adam–Cuckoo search-based deep belief network classifier for data classification, *IEEE Access* **8**(1) (2020).
[27] P. Quax, A. Beznosyk, W. Vanmontfort, R. Marx and W. Lamotte, An evaluation of the impact of game genre on user experience in cloud gaming, in: *Games Innovation Conference (IGIC)*, 2013, pp. 216–221. doi:10.1109/IGIC.2013.6659141.
[28] M.A.A. Sibahee, H.B. Abdalla and A.M. Ahmed, Optimization driven MapReduce framework for indexing and retrieval of big data, *KSII Transactions on Internet and Information Systems* **14**(5) (2020).
[29] I. Slivar, L. Skorin-Kapov and M. Suznjevic, QoE-aware resource allocation for multiple cloud gaming users sharing a bottleneck link, in: *22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 2019, pp. 118–123. doi:10.1109/ICIN.2019.8685890.
[30] M. Suznjevic, I. Slivar and L. Skorin-Kapov, Analysis and QoE evaluation of cloud gaming service adaptation under different network conditions: The case of NVIDIA GeForce NOW, in: *8th Intl. Conf. on Quality of Multimedia Experience (QoMEX)*, 2016, pp. 1–6.
[31] H. Tian, D. Wu, J. He, Y. Xu and M. Chen, On achieving cost-effective adaptive cloud gaming in geo-distributed data centers, *IEEE Transactions on Circuits and Systems for Video Technology* **25**(12) (2015), 2064–2077. doi:10.1109/TCSVT.2015.2416563.
[32] Z. Zhu, J. Peng, K. Liu and X. Zhang, A game-based resource pricing and allocation mechanism for profit maximization in cloud computing, *Soft Computing* (2019).