IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 2, 2020, accepted March 19, 2020, date of publication March 24, 2020, date of current version April 9, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2983080

Boosted Metaheuristic Algorithms for QoE-Aware Server Selection in Multiplayer Cloud Gaming

HOSSEIN EBRAHIMI DINAKI[1], (Member, IEEE), SHERVIN SHIRMOHAMMADI[1], (Fellow, IEEE), AND MAHMOUD REZA HASHEMI[2], (Senior Member, IEEE)

[1]School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON K1N 6N5, Canada
[2]School of Electrical and Computer Engineering, University of Tehran, Tehran 1439957131, Iran

Corresponding author: Hossein Ebrahimi Dinaki (hebra056@uottawa.ca)

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) Strategic Project under Grant STPGP 506890.

ABSTRACT Cloud Gaming (CG) provides a high performance and cost-effective solution where players with low-end devices can play high-end games without the need for advanced hardware. A cloud-based video game system offloads all the computational tasks to the cloud. Considering the dynamic nature of game workloads and resource capacity, resource management is still a significant challenge. Since CG is a real-time gaming service, graphics processing units (GPUs) are necessary to accelerate game scene rendering. GPUs are one of the most expensive resources in a CG platform. Therefore, service providers have a strong incentive to utilize GPUs efficiently to maximize their economic profit. In addition, players' quality of game experience (QoE) is a crucial parameter that can directly affect a service provider's profit and must be taken into account in any resource scheduling optimization. To satisfy both parties, in this paper we propose two efficient methods for GPU based server selection in CG. The proposed methods are an improved version of two well-known metaheuristic algorithms called Particle Swarm Optimization (PSO) and Genetic Algorithm (GA), which we refer to as Boosted-PSO and Boosted-GA, respectively. The proposed methods consider service providers' profits and players' experience simultaneously. Our objective is to maximize GPU utilization, which will not only lead to the service provider's economic benefit, but also increase the player's QoE. Our simulation results show that compared to the existing methods to solve such an NP-Hard optimization problem, our Boosted-PSO method, followed by Boosted-GA, achieves the highest efficiency in terms of GPU utilization, capacity wastage, and player's QoE.

INDEX TERMS Cloud gaming, server selection, GPU utilization, genetic algorithm, particle swarm optimization, quality of experience, cloud resource allocation.

I. INTRODUCTION

According to Newzoo's report, there were 2.5 billion active gamers worldwide in 2019, with an expected global games market of $152.1 Billions [1]. Remarkable advancements in cloud computing provide inexpensive and flexible opportunities for game service providers to deploy their games in the cloud, known as Cloud Gaming (CG). In the CG model, a.k.a Game as a Service (GaaS), or gaming on demand computationally intensive tasks such as the game engine, graphics rendering, encoding the game scenes is performed on remote servers in the cloud, and the game video is streamed to the player's end device [2]. The cloud servers have higher processing power and memory compared to the players' device, and the only requirement on the client-side is a broadband internet connection and ability to play video, not the need for high-end hardware. These features are attractive for game providers and have encouraged even big stakeholders such as Google, Amazon, Verizon, Apple, and Electronic Arts to develop their own video game streaming services [3]. CG's global market in 2017 was $802 million and is estimated to reach $6.944 billion by 2026 [4]. Onlive, G-cluster, and Gaikai were few of the pioneer companies of GaaS. Sony acquired Gaikai in 2012 and an essential part of Onlive in 2015 when the later discontinued its services. Currently, Sony's game video streaming service, PlayStation Now, is powered by technology from Gaikai. Another company, Broadmedia GC Corp., has been operating its own version of CG using its G-cluster technology since July 2016 [5]–[8]. Most recently, Google announced its own CG service, Google Stadia [9], with much fanfare. To summarize, several CG service providers are available in the market e.g. PlayStation

The associate editor coordinating the review of this manuscript and approving it for publication was Xujie Li.

Now [10], Parsec [11], VORTEX [12], GeForce Now [13], Playkey.net [14] Google Stadia [9].

In CG, players operating on heterogeneous devices; e.g., PC, laptop, tablet, game consoles, desktops, set-top boxes, and smartphones, send their commands to the cloud game server which runs the game engine and makes the appropriate decision to produce the corresponding video frames rendered using a GPU and encoded by a video codec. Afterwards, the compressed frames are streamed to players through the network, and finally decoded and played on user's devices [15]. The tasks that need to be performed on the server-side require a well-organized resource management system to satisfy both the service provider's and end-user's requirements [16]. This management system would be more challenging for Multiplayer CG (MCG) cases.

The real-time and interactive nature of MCG makes response delay a very sensitive factor in the end-user's quality of experience, and thereby many works have been done to address this challenge [17]–[19]. Response delay consists of network delay, processing delay, and playout delay. Usually, the playout delay is assumed to be negligible and does not have a significant impact on player's game experience [20]. To reduce the network delay, normally a large number of games are supplied by service providers and replicated in multiple instances at different datacenters in the cloud, which are geographically distributed. Processing delay depends on the available processing power in the cloud server. Processing power, in turn, is determined by server resources such as CPU, GPU, memory, and storage, and the workload of the game sessions that run on the cloud server.

Furthermore, the server selection strategies would be more complicated, considering that there is heterogeneous computation power of the resources in cloud servers. Furthermore, CG has higher delay sensitivity compared to the other platforms, such as online games [16]. Since each game, depending on its genre and pace, has different resource requirements, assigning the cloud servers to game players in this distributed network is one of the remaining challenges of CG platforms [21].

Since in CG systems, the game's video updates must be streamed to the player's devices, the spatial and temporal quality of the video is a very important parameter to satisfy player's perceived video quality [2]. The better the spatial quality, the higher the resource requirement, e.g., bandwidth and processing power. Many works have been done to deliver high-quality video to players while minimizing resource usage. For example, authors in [22] proposed a method considering the visual attention and object priority, and achieved a noticeable bitrate reduction while keeping good enough player's game experience. To improve the accuracy of the predicted attention maps, the effect of game state was considered in [23]. Authors in [24] conducted a test procedure to evaluate game performance for Samsung mobile phones. Regarding their evaluations, video smoothness (i.e., Frame rate and its stability) and graphic quality are essential factors for performance benchmarking apart from phones' battery, temperature, and swiftness. Also, frame rate contribution on the QoE and performance depends on the user type (e.g., passive or active), technology type (e.g., TV, VR ...), task type, etc. [25]. Although spatial quality is an important parameter for player's QoE in CG, in comparison with the other media, the frame rate has a crucial impact on game player's performance [26]. Therefore, game players are always looking for higher frame rates, and rates of up to 120 fps are now being demanded, depending on the game type. Game developers are trying to address this demand; for example, Battlefield 4 and Counter-Strike run at 120 fps with supporting hardware [27]. This demands more resources such as GPU in CG systems and makes the resource management problem more critical than before.

In terms of hardware resources, GPU's necessity is a fundamental difference between CG and other applications operating on cloud datacenters. GPUs play the leading role for fast video rendering in CG as a real-time service, therefore using GPU-equipped cloud servers in this platform is an essential requirement. Also, GPU is the most expensive infrastructure that CG providers should pay for [21]. Therefore, its maximum utilization is vitally important from their profitability perspective. However, utilization should be efficient to satisfy more players as well, as they are the primary source of revenue for the CG provider [21]. Consequently, GPU is a critical parameter for server selection in the CG platform.

There are two main strategies for server selection, depending on the type of datacenter: public or private. In public datacenters, such as Amazon EC2, Microsoft, and IBM, tenants such as CG providers pay for the reserved resources, and there is no interest in false requests there. Therefore, the resource management methods emphasize the utilization property rather than fairness consideration. For example, if 100 servers can handle 400 requests, the case of 200 requests using 50 servers is rational in this type of cloud. However, for a big enterprise that owns a private datacenter where the users share the resources and will not pay for them, focus on fairness is much more important than utilization, and the trade-off between these two can be taken into account [28]. In our work, our emphasis is on utilization rather than fairness.

In our previous work, we proposed a method that optimally assigns cloud servers to the game sessions requested by game players [29]. The method assesses the requested game and its required video quality in terms of frame rate and the load of the frames while considering the capacity of the *eligible datacenters* to allocate an appropriate cloud server to the player for the requested game session. We modeled the problem of server selection as an optimization problem focusing on maximizing both the GPU utilization and players' QoE. Our objective was the maximization of GPU utilization, which leads to the service provider's economic benefit, while increasing the player's QoE. To determine the priority of the two objectives, an end-to-end lag model weights them adaptively [29]. Then we proposed two efficient methods based on two metaheuristic algorithms, namely: Particle Swarm Optimization (PSO) and Genetic Algorithm (GA). They worked

well in terms of maximizing GPU utilization or minimizing capacity wastage and also improving the QoE.

However, the utilization percentage for small number of players in [29] is not sufficient enough and needs to be improved. For example, the results show almost 20 and 40 percent capacity wastage, for 20 players' case in PSO and GA-based methods respectively, however for 100 players' case, these wastages are half of these percentages. This problem shows that in some cases, a premature convergence takes place, and the algorithm is stuck in a local minimum. To generalize the algorithm and have almost a consistent efficiency for the different numbers of players, we should address this issue adequately. Towards this goal, in this paper, we propose a new sub-algorithm which replaces the random regeneration part of the previous algorithms. The idea behind this sub-algorithm is that when the algorithm understands that it is stuck in a local minimum, it should boost itself to come out from that local point. This sub-algorithm plays a reinforcement role when the main algorithm approaches towards premature convergence. Besides, since it is not always in the main algorithm's chain, its impact on the total speed of the main algorithms would be negligible. We refer to these two new algorithms as Boosted-GA and Boosted-PSO, hereafter.

There are two placement approaches in cloud literature: static and dynamic [30], [31]. The static approaches place the users based on their service demands and the constraints of the problem statement. Dynamic or online approaches, in addition to placement, monitor or predict the demand and update the decision, accordingly. What we present in this study is a static approach, which is a placement at the initial step that works very well in terms of utilization with respect to the constraints.

To measure the improvements of the proposed algorithms, we have compared them with the conventional GA and PSO algorithms presented in [29], as well as four well-known and widely-used bin-packing algorithms called Best Fit Algorithm (BFA), First Fit Algorithm (FFA), Next Fit Algorithm (NFA) and Worst Fit Algorithm (WFA). The results indicate that our boosted algorithms achieve a remarkable improvement in terms of utilization and capacity wastage, and that the Boosted-PSO is the best one among them.

The rest of the paper is organized as follows: Section II describes the relevant background, and related works. Section III represents our Utilization, Delay and User Experience models. Section IV describes the problem statement. The Metaheuristic algorithms are described in section V, while section VI presents the performance evaluation results. Finally, in Section VII, we present the conclusions and future works.

II. BACKGROUND AND RELATED WORKS

A. BACKGROUND

As mentioned above, CG is a flourishing gaming paradigm that brings noticeable benefits for both game players and providers. Three main architectures can be considered for this cloud-based gaming system [32]: Local Rendering GaaS, Remote Rendering GaaS, and Cognitive Resource Allocation.

In Local Rendering GaaS, the rendering operation is performed at the client-side by using the display instructions set, which represent the gaming graphic and is streamed from the cloud game server. Therefore, client's devices require powerful enough hardware to perform rendering. Since this scenario avoids video streaming over the network, it reduces the burden of the network significantly. However, computational capability and battery-dependency of the client's portable devices are the considerable challenges in this scenario.

The second architecture is called Remote Rendering GaaS (RR-GaaS), in which all of the computational tasks such as gaming logic, video rendering and capturing, video encoding and streaming are performed at the remote servers. In other words, only decoding and displaying the compressed video bitstream are performed at the client-side. Clearly, offloading all computational complexities to the cloud server-side enables players to play high-end games with low-end devices. However, the selected datacenter and server for remote rendering must satisfy the player's tolerable delay and its required processing capacity for the requested game type and quality.

The third architecture of cloud gaming is Cognitive Resource Allocation [33]. In this architecture, depending on the currently available resources in the client and the network, the game may be operated at the client-side or in the cloud server. In other words, this system manages the game's computational operation based on its cognitive capabilities. As gaming is a real-time service and the load of the requested resources is completely stochastic, providing an adaptive optimization system for this cognitive system is a considerable challenge in this scenario.

Among the above architectures, as mentioned earlier, the RR-GaaS is currently available in the market and was implemented by several CG service providers including OnLive, StreamMyGame, Gaikai, and G-Cluster and is used by the new owners of these companies. That is why we consider RR-GaaS in our model in this paper.

Since in this architecture, all complex computations are transferred to the cloud server, and only the video is streamed to the client-side, the quality of the transmitted video and the end-to-end delay are of vital importance for player's experience. To attain a tolerable response delay, both network delay and processing delay should be taken into account. The distance of the assigned datacenter and the network's traffic conditions are the two important parameters to provide an appropriate network delay (e.g., 80 msec) [34]. Selecting a desired rendering server (RS) in cloud datacenters, based on the game's load and computational capacity of the server, has a direct effect on processing delay. Also, there is a close connection among game genre, its required quality, and the processing complexity, and also the hardware rented by the service provider in the cloud server [26], [35].

Consequently, this architecture involves both player's QoE and provider's benefits and profit and in any resource management system such as server selection, considering both parties' concerns is imperative. Furthermore, all these challenges will be intensified for the MCG case.

Fig.1 shows how, in a CG system, players can be assigned to the servers. As shown in the depicted framework, the game server (GS) is the point that manages the connections between players and the cloud gaming system. Processing the state of the game, evaluating the resource requirement of the game, and allocating the resources to players are the main functions of the GS to arrange a game session between a group of players. For each received command, the GS evaluates the state of the game and the required resources of the game scenes and then assigns the game player to a server in an *eligible datacenter*. It is assumed that the *eligible datacenter* has the minimum required delay, bandwidth, and processing power for the assigned game sessions. Therefore, the player can play the game in the datacenter where the game is rendered in a GPU equipped RS, and then the game's high-quality video for the new scenes are streamed to the player.

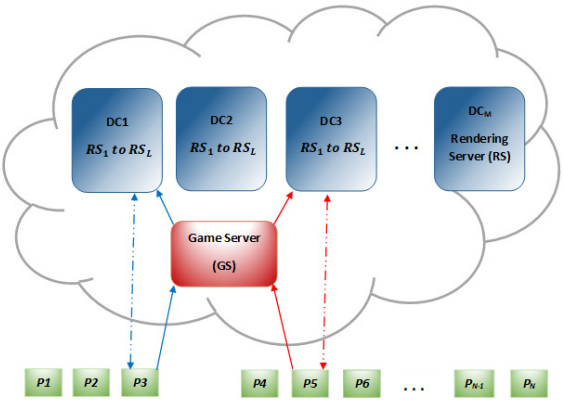

FIGURE 1. Server selection in MCG.

B. RELATED WORKS

There are different approaches to the problem of datacenter or server assignment to players in the literature. Some try to reduce the expense of the service provider while others prefer to increase the player's quality of experience. In this way, they employ infrastructure, network, and video parameters in their optimization problem.

The heuristic game hosting server selection method, presented in [34], tries to place the game according to the end-user's voting-based game-placement strategy. In this method, players vote for a game to be hosted on a server in order to serve more players for on-demand gaming. This gives a significant improvement in the number of served end-users compared to a random game placement. The server selection method introduced in [36] is looking for the optimal servers to serve a group of players' requests where the players are located in the same geographical region. The objective is to assign the servers in such a manner as to minimize latency in data transfer. The authors in [37] formulate a multi-objective problem for virtual machines' provisioning in cloud gaming. The objective of the problem is to minimize inter-player delay among interactive gamers and electricity expenses while providing a good-enough response delay to the gamers. They present a gray wolf-based algorithm that efficiently solves the problem. The network parameters such as response delay and inter-player delay are considered as QoE. In all of the above papers, the servers have been selected only based on Round-trip delay time (RTT) or inter-player delay, and the video quality and computational resources' capacity have not been taken into account.

The work in [21] focuses on server acquisition between available datacenters based on the minimum price to support MCG. They consider four methods, three of them based on price: 1) Lowest Server Price Assignment, 2) Lowest Bandwidth Price Assignment, 3) Lowest Combined Price Assignment, and one based on wastage: Lowest Capacity Wastage Assignment. The results show that compared to Random Assignment and Nearest Assignment, considering wastage is helpful to achieve cost-effectiveness. Authors of [38] formulate the MCG Problem by considering the latency between the client, game server, rendering server, and the datacenter and then present several heuristics to address the server allocation problem for MCG. To solve these NP-Hard problems, they introduce a combination of price and wastage-based assignment algorithms with a greedy hill-climbing approach, called Lowest-Amortized-Cost, to take into account three parameters: server cost, bandwidth cost, and capacity wastage. Their experimental results show that for the same session size (number of players playing the same game), by increasing the latency threshold, the normalized cost of their proposed algorithm remains stable. Another work tries to optimize the inter-players delay between players who are interacting with each other by using an optimization algorithm that places all players' Virtual Machines (VM) and then adjusting the placement [18]. It presents an optimization algorithm to reduce inter-player delay while keeping normal absolute response delay perceived by players, leading to a reduction of up to 30% in inter-player delay. Our approach differs from the above works since their focus is only on delay and bandwidth as quality of service (QoS) metrics, but not QoE in their models and evaluations.

A QoE-aware resource allocation framework is presented in [39] for a mobile cloud gaming system. The proposed system, referred to as EdgeGame, offloads computationally intensive tasks into the edge instead of the remote rendering server. The EdgeGame optimizes users' QoE in the edge using a reinforcement learning method. This system reduces delay and bandwidth costs. However, as they mentioned, it forces a noticeable investment to the service providers at the Edges.

An optimization problem for VM placement in Massively Multi-player Online Gaming was presented in [30]. The goal is to minimize the cost of the VM placement under delay constraints. Toward this goal, they map the problem to a

multi-knapsack problem and propose a heuristic algorithm called Cost-aware VMs placement algorithm as a solution. The results show that their proposed method has higher effectiveness in terms of cost-saving, compared to random, Greedy Allocation Cost and Minimum Allocation Cost placement algorithms. In the cloud gaming system, the game's software should be installed on the distributed servers to avoid slowing down the loading and disrupting players' interaction. Authors in [40] formulate an optimization problem for server provisioning to minimize the software storage cost and the server running cost. Among the multiple heuristic algorithms which are employed to solve the problem, their ordered method and Genetic Algorithm have the best performance in terms of the minimum service cost and robustness for dynamic changes. However, both papers did not consider the impact of their placement on the client's quality of experience.

Authors in [41] evaluate QoS parameters and compare them in two different CG systems, namely OnLive and StreamMyGame (SMG). Based on their measurements, OnLive's downlink bandwidth is between 3 and 5 Mbps while SMG's is between 9 and 18 Mbps, showing that the server traffic is system dependent. Based on their results, game delays are also game-dependent. Then, they show that Processing delays (PD) of the two systems is related to the coding technique and hardware acceleration used in the video encoder. Their evaluation for the effect of computational power on PD shows that from a lowest-end to a highest-end CPU, there is merely a 20% improvement, and they concluded that SMG's longer PD must be due to the nature of its architecture/implementation.

The work in [42] proposes two VM placement algorithms to maximize the provider's total profit while providing just good enough QoE to players. They propose their algorithms for public and closed CGs (e.g., Hotels, Internet Café). The public algorithm sorts the servers according to the network latency for that player and then iterates among the sorted servers to support players based on their required resources. For closed CG, since maximizing the QoE is more important, their second algorithm sorts the servers based on quality degradation levels and then iterates through servers and creates a VM on the first server that can support the player. The results show that their heuristic algorithm attains very good results in terms of profits; however, as specified in their VM evaluation, considering VMs as utilization units have a noticeable overhead to the physical machine. Finally, [43] introduces a cost-effective way of dynamic resource provisioning for CG in Geo-distributed datacenters from the service provider's perspective. It reduces the cost of a CG service by using adaptive ways to select datacenters. It categorizes the users based on the game genres to meet the QoE requirements of different game types and cut down at least 25% of service costs. Their model is cost-effective from the service provider's perspective, and they assign a separate and dedicated server for each player. However, in our model, depending on the load of the games, each server can be used for multiple players simultaneously, which is a more realistic condition.

Furthermore, to satisfy both parties, we align the service provider's benefits and the player's experiences and try to optimize these objectives. By efficiently utilizing the GPUs as the most expensive hardware in CG, we are trying to maximize the service provider's economic benefit, while the end user's experience is taken into account in our quality model.

III. UTILIZATION, DELAY, AND USER'S EXPERIENCE MODELS

A. GPU UTILIZATION MODEL

GPUs play the leading role in speeding up the rendering of the game's video and reducing the processing delay of the CG system and consequently have a considerable effect on end user's experience. Besides, they are the most expensive hardware rented by providers [21]. Furthermore, not all of the servers in datacenters are equipped with a GPU. For these reasons, management of this resource is highly significant. Therefore, we consider its utilization as one dimension of the proposed server selection method. To this end, we formulate the GPU utilization model, and the formulation for the resource which is utilized by one player is as follows:

$$U_{ijl} = \frac{T_{ik} F_{ik}}{f_{jl}} \quad \forall i \in P,\ \forall j \in D,\ \forall l \in G \tag{1}$$

$$T_{ik} \leq \frac{1}{F_{ik}} \tag{2}$$

where player i is connected to GPU l in datacenter j. The proposed GPU utilization model is extracted from [44]. f_{jl} is the GPU frequency located in the server l in datacenter j. T_{ik} is the processing time of one frame in genre k and must be equal or less than $\frac{1}{F_{ik}}$ where F_{ik} is the frame rate of the game in genre k selected by player i. And, $T_{ik}F_{ik}$ is the processing time for doing the task of the requested game by player i in one second. In other words, this is the workload of each player corresponding to the selected game and frame rate. In our problem statement in section IV, we assume that each datacenter has L RSs, each of which can serve multiple players. All notations are summarized in Table 2.

B. RESPONSE DELAY MODEL

The response delay is the time between a player command issued on the players' device (e.g., move, run, jump, fire, press a button, etc.) and watching the corresponding result on the display. High delay sensitivity of the RR-GaaS is a serious challenge in this architecture [16], [17]. Intuitively, in server selection, the server with the shortest distance to the game player should be selected to do the rendering operation; however, many other challenges may be involved. For instance, the server may be already assigned to other players and not have enough resources for a new game, or the player is playing with another player, so the location of the other player should be taken into account to minimize

the inter-player delay, and hence the closest server selection is not the only criterion anymore.

For the MCG described in Fig.1, we assume that all players are in the same cloud region; however, they can be assigned to different datacenters. Due to the reliability of the game servers, we assume that the system can assign gamers to the datacenters that pass the minimum required network delay; e.g., 80 msec. In this case, authors in [45] investigate the impact of the frame rate produced at the server on the response delay in the system, and show the synergy between the server-side tick rate and the client-side frame rate. The tick rate is the frequency of the game state update in the server, and the frame rate is the number of frames per second displayed to the user. To model the delay, we utilize their end-to-end lag data from the output of the video game simulation toolkit in [46], which is written in R. We utilized this data calculated for six different frame rates. Then we formulate the response delay based on the frame rate. To this end, we evaluate the regression model for several functions, such as $\sqrt{x}$, $ax^2 + bx + c$, log(x) where the logistic sigmoidal function, $Y(F) = 1/\left(1 - e^{(-\alpha-\beta.F)}\right)$ is the best fit function, by its R-square value of 0.97, and α and β are the model's parameters.

$$RD(F) = \mathrm{Y}(F) = 1/\left(1 - e^{(-\alpha-\beta.F)}\right) \tag{3}$$

The R-square values indicate how close the sigmoidal function follows the input data. The response delay data and the corresponding fit curve of the function have been depicted in Fig. 2. The curve has a sharper slope for lower frame rates however, if we compare the response delays for 30fps and 120fps, it is around 70 msec. Moreover,if we compare this value with the 80 msec allowable network delay [36], we realize that selection between the frame rate is still valuable and has a noticeable effect on the response delay.

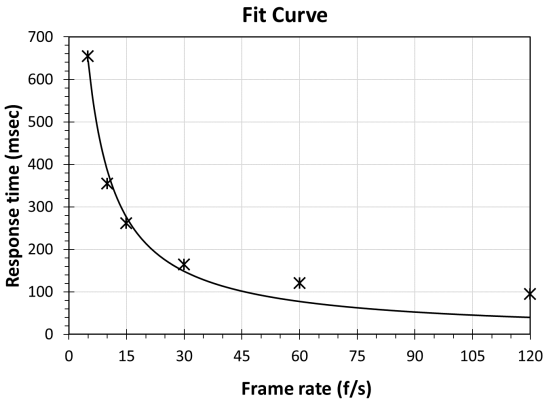

FIGURE 2. Regression model curve fitting for logistic sigmoid.

In equation (4), we denote the λ coefficient, acquired from the derivative of the above sigmoidal function $Y(F)$:

$$\lambda_i = \left[\frac{dY(F)}{dF}\right]_{F=F_{ik}} = |e^{(-\alpha-\beta.F)}/\left(1 - e^{(-\alpha-\beta.F)}\right)^2|_{F=F_{ik}} \tag{4}$$

As we described above, the frame rate has a noticeable effect on the response delay [45]. Hence, we use frame rate as the main variable in our problem statement (described in section IV), and we use λ as a priority factor, representing how quickly response delay changes with respect to the increase or decrease of frame rate. More about this will be shown in section IV.

C. USER'S EXPERIENCE MODEL

As the user's satisfaction has a direct connection with the provider's profit, in any resource management system, end-users' QoE should be taken into account. In RR-GaaS, the game's video streams to the player's device. Compared with regular video streaming, video games require a higher frame rate. Due to higher interactivity and reactivity in video games, this feature would be an essential factor that has a noticeable impact on the performance of players, especially for fast-paced games. The study of Claypool *et al.* in [26] for first-person-shooter games shows that frame rate and resolution both have an impact on the perceived quality, but frame rate has a more significant influence on a player's performance than resolution. The tests conducted by [47] also shows the effects of frame rate on quality of experience. To consider the effect of frame rate, we modeled the QoE as a function of frame rate variation. Towards this end, inspired by the measurement study in [47] that represents some objective QoE metrics as a good approximation for MOS value, we deploy Structural SIMilarity (SSIM) as an objective quality of experience metric and formulate our QoE model. In this way, we chose a dataset of nine games from [35]. They are in three genres: Real-time strategy (RTS), First Person Shooter (FPS), and Role-Playing Game (RPG). The videos were prepared in 6 different frame rates: 15, 30, 45, 60, 90, and 120 frames per second using the ffmpeg open-source software suite [48]. Then we employed a regression model for several functions to find the curve fit for the calculated quality.

The "four parameters logistic" function in (5) is the best fit function for the above measurements.

$$Q(F_{ik}) = d + \frac{a-d}{1 + (\frac{F_{ik}}{c})^b} \tag{5}$$

where a, b, c, and d, are model parameters. A sample of the measurement's results for three video games, Battlefield V (BFV), Age of Mythology (AoM), and War, which are representatives of three genres, are plotted in Fig. 3. The quality models obtained from this study apply only to the measured games. The SSIM values for different genres are different, but the trend of the curve by altering the frame rates is the same, and the same model-driven method can be applied to other genres. We report the R-squared values between the fitted curve and the collected data for the three genres in Table 1.

IV. PROBLEM STATEMENT

An MCG system assigns multiple servers, between the eligible datacenters, to multiple players. The eligible datacenters

TABLE 1. R-squared values of different genres.

GENRE	RTS	FPS	RPG
R^2	0.9372	0.8998	0.8778

TABLE 2. Notation table.

Notations	Definitions	Notations	Definitions
N	Number of Players	T_{ik}	Processing time of one frame of the game in genre k, requested by player i
M	Number of Datacenters	$T_{ik}\ F_{ik}$	The time for processing the task of the player i in genre k in one second
L	Number of GPU equipped servers in each Datacenter	$RD(F_{ik})$	Response delay as a function of frame rate allocated for player i
K	Game genre's number	α, β	Delay Model parameters
F_{ik}	fps allocated to game or player i with genre k	U_{ijl}	Resources utilized by player i in server l in datacenter j
x_{ijl}	Whether Player i is assigned to GPU_l (RS_l) in DC_j or not	$Q(F_{ik})$	QoE for player i, (P_i), playing game with fps F_{ik}
F_{mink}	Minimum required fps for genre k	a, b, c, d	QoE Model's parameters
F_{maxk}	Maximum fps for genre k	f_{jl}	frequency of GPU l located in datacenter j
f_{max}	Maximum GPU frequency	λ_i	Weighting coefficient
$v_1(x)$, $v_1(x)$	Violation functions	$f(x)$	Fitness function
X_i^t	Particle i's position at iteration t	V_i^t	Particle i's velocity at iteration t
$lbest$	Local best particle	$gbest$	Global best particle
ω	Inertial weight	c_1, c_2	PSO velocity model parameters
φ_1, φ_2	Random values between 0 and 1		

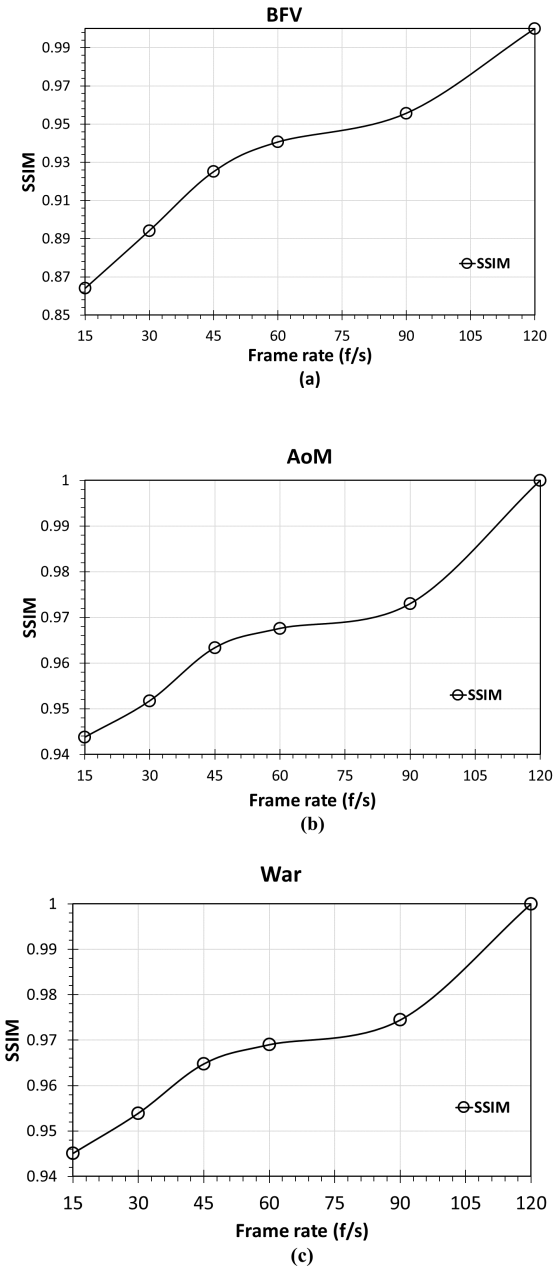

FIGURE 3. SSIM values versus Frame rate (f/s) for three games (a) BFV(FPS), (b) AoM (RTS), (c) War (RPG).

are the datacenters that pass the minimum RTT requirement concerning the game type. As an eligible datacenter is one that has enough capacity for processing a new game and due to the necessity of needing enough bandwidth for an appropriate QoE [49], we assumed that each server in the datacenter has not only the rendering capacity but also sufficient bandwidth. Each server can serve multiple players, including sharing a single physical GPU among multiple users [50], depending on their game's load. The cloud server that has the maximum GPU utilization and still has enough capacity, would be assigned to the new players. The server selection problem is a problem from the bin-packing (BP) problem family, which is NP-hard [51] with no known solution to be run in polynomial time. Most of the BP based algorithms are heuristics, and the four popular ones are First Fit Algorithm (FFA), Best Fit Algorithm (BFA), Next Fit Algorithm (NFA), and Worst Fit Algorithm [52]. The goal in all of these BP algorithms is how to efficiently utilize the resources such that to maximize the utilization and, consequently, the service provider's profits. For a server selection problem, physical machines are bins, and the requests for resources, e.g., CPU, GPU, or storage, should be managed to attain a relevant packing result.

To simultaneously optimize both the player's QoE and the game provider's profits, we propose an optimization problem to maximize both utilization and the player's experience. Fig. 1 illustrates the system architecture of MCG. In our proposed model, which is a group assignment, the players request the game video frame rates as a range instead of just one value; i.e., depending on the genre, the client's machine will ask for a minimum and a maximum (desired) frame rate. The proposed algorithms, which are run in the GS, assign the players to an eligible cloud server by considering the requested loads and processing power of the servers. Since in our formulation the profit of the service providers, in terms of the maximum utilization of the RS and the users' QoE, are in the same line, our proposed algorithms try to maximize GPU utilization and allocate maximum frame rate to the players subject to satisfying the constraints of the servers. In the following section, we explain our problem formulation and our objective function in detail.

A. PROBLEM FORMULATION

We assume that there are N players $= \{1, 2, \ldots, N\}$, M eligible datacenters $D = \{1, 2, \ldots, M\}$, L GPU included RSs in each datacenter $G = \{1, 2, \ldots, L\}$, and $F = \{F_{mink}, \ldots F_{ik}, \ldots F_{maxk}\}$ is a group of frame rates offered for each genre k. The processing time of each frame is T_{ik}, and $f_{jl} = f_{max}$ is the maximum processing capacity of the GPU. The frame rate set contains minimum frame rate F_{mink} required for the player's minimum QoE, the maximum frame rate F_{maxk} beyond which the player can no longer perceive the higher quality, and other frame rates between these two bounds. The objective function is then formulated based on the minimum requirements of the game, in terms of the frame rate, its corresponding processing time, T_{ik}, the maximum processing capacity of the GPU, f_{max}, and the maximum quality that can be perceived by the player.

Normally, in resource allocation systems, the decision-maker prefers to utilize smaller loads, in our case frame rate, to achieve maximum utilization. Nevertheless, as we discussed in sections III-B and III-C, the smaller frame rate leads to a perceived higher response delay and also lower user experience. On the other hand, assigning the highest frame rates would lead to lower servers' utilization and more capacity wastage. To avoid these, we employ a weighted sum approach, where these weights would adaptively control both objectives of the problem with respect to random variations of input variables. In this way, since our quality objective and the response delay model are aligned, we used λ from eq. (4) in eq. (6) as a priority factor which reflects the rate of changes of the response delay curve, regarding the frame rate. This ratio shows how quickly response delay changes concerning the increase or decrease of frame rate. By using this variable weight, the decision-maker can dynamically determine the preference of the corresponding objective. Regarding the above definitions, our objective function is as follows:

$$
\begin{aligned}
&O(x_{ijl}):\\
&Max\Big(\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{l=1}^{L}\big((1-\lambda_i).\frac{(U_{ijl})^r}{\hat{m}.\hat{l}}\\
&\quad+\lambda_i.Q(F_{ik})\big)\Big).x_{ijl}\\
&\text{Subject to}: \ \text{a)}\ F_{mink}\le F_{ik}\le F_{maxk}\\
&\qquad \forall i\in\{1,2,\ldots,N\},\ \forall k\in\{1,\ldots,K\}\\
&\qquad \text{b)}\ \sum_{i=1}^{N}\sum_{j=1}^{M}T_{ik}F_{ik}.x_{ijl}\le f_{jl}\\
&\qquad \forall l\in\{1,2,\ldots,L\}\\
&\qquad \text{c)}\ T_{ik}\le\frac{1}{F_{ik}}\quad \forall i\in\{1,2,\ldots,N\},\\
&\qquad \forall k\in\{1,\ldots,K\}\\
&\qquad \text{d)}\ \sum_{j=1}^{M}\sum_{l=1}^{L}x_{ijl}=1,\quad \forall i\in\{1,2,\ldots,N\}\\
&\qquad \text{e)}\ x_{ijl}\in\{0,1\}\quad \forall i\in\{1,2,\ldots,N\},\\
&\qquad \forall j\in\{1,2,\ldots,M\},\ \forall l\in\{1,2,\ldots,L\}\\
&\qquad \text{f)}\ \hat{m}.\hat{l}=\sum_{l=1}^{L}\min\Big(\sum_{i=1}^{N}\sum_{j=1}^{M}x_{ijl},1\Big)\quad (6)
\end{aligned}
$$

The above objective function has two parts. The first part is utilization, where U_{ijl} is derived from section III-A and represents the utilization of the GPU l in the datacenter j when player i, playing in genre k, is assigned to server l. Also, $\hat{m}.\hat{l}$ is the number of the eligible used servers, and $r > 1$ is a constant value [53]. The second part is related to the player's QoE concerning the assigned frame rate. The function of the QoE model is derived from section III-C. The $0 < \lambda_i < 1$ is a weighted average coefficient which increases and decreases with the decrease and increase of the frame rate, respectively. Constraint (a) bounds the acceptable range of the frame rate for the genre k, which is played by player i. Constraint (b) determines the maximum allowable workload for each GPU. The workload of the RS_l must be equal or less than the maximum processing capacity f_{max}. Here we assume that all servers are working at their maximum frequency. The processing time of a single frame in genre k must be equal or less than $\frac{1}{F_{ik}}$ which is mentioned in constraint (c). Moreover, constraint (d) recognizes that only one server l in only one datacenter j can be assigned to each player i. The x_{ijl}, in constraint (e), is a binary value where one means that the server l in the datacenter j is assigned to the player i and zero represents no allocation. Finally, (f) calculates the number of the used GPUs.

V. PROPOSED METAHEURISTIC ALGORITHMS

In this section, we present the two proposed metaheuristic algorithms based on Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) and also our proposed sub-algorithm for boosting of these two methods.

A. PROPOSED PSO MODEL

PSO, a well-known member of evolutionary algorithms' family, is first presented by Kennedy, Eberhart in 1995 [54]. Social behaviour of a bird in the flock, a fish in a school,

or an insect in a swarm, constructs the foundation of the PSO algorithm. In this algorithm, each single candidate solution in the search space is called a particle. They provide individuals of the population in the feasible search-space. The individuals communicate with each other to move toward the same regions of problem space. To find the best direction, they adjust their velocity and position iteratively by using their own knowledge, and the information comes from the neighbour swarm members. Fig. 4 shows how an individual updates its position and velocity in a PSO algorithm. Each particle tries to update its current position and velocity regarding the distance between its current position and its best-found position in the feasible search-space. The best-found is updated as the better position found by the particles, and the distance between its current position and global-best achieved among all particles in the swarm. PSO has been successfully deployed in many areas, such as fuzzy system control, function optimization, artificial neural network training, etc. [55], [56]. Fig. 4 represents the process in PSO to update the location of the particles.

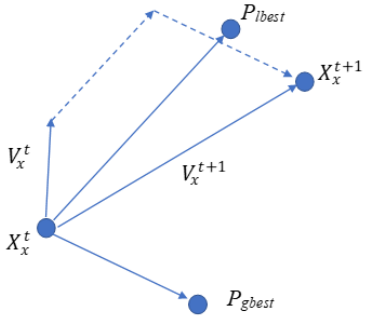

FIGURE 4. How a particle finds a new direction and position by comparing the current position (X_x^t) with current local best position (lbest), global best position (gbest), and velocity (V_x^t).

Now, we propose the steps of our PSO based model for the problem presented at section IV-A:

Step 1: Initial Population: Initialize population of the individual solutions, contains their position, fitness value, velocity. In our proposed model, each particle represents a cloud server allocation outline, $X = \{x_{111}, x_{112}, \ldots, x_{1ML}, x_{211}, x_{212}, \ldots, x_{ijl}, \ldots, x_{NML}\}$ where N is the number of players, M is the number of eligible datacenters, and L is the number of GPU-equipped servers in each datacenter. In this particle, $M \times L$ binary components assign to each player an identifier code, and each binary number in this code represents the cloud server that could be assigned to this player. Since normally we know the number of servers in each datacenter, this particle codes the datacenters as well. We deploy the decimal version of the above binary codes inside the computation of the algorithm, and then we convert the decimal number to the binary to determine the server selection results in the last iteration. So, the definition of an individual particle is: particle $= \{P_1, P_2 \ldots P_i \ldots P_N\}$. We set the number of particles for our experiments to 25.

Step 2: Evaluate particles' fitness: Evaluate the fitness of individual particles with the fitness function.

Fitness Function: In this paper, our fitness value comes from the objective function (6) and its constraints. This fitness value demonstrates the utilization of the GPUs, the players' QoE, and some penalty values to handle the constraints.

To reflect all of the goals of the problem in the fitness function, we define some violations' functions from the above constraints and add them to the objective function as a penalty function. The defined violation function $v_1(x)$ and $v_2(x)$ come from constraints 6(b) and 6(d) respectively:

$$\mathrm{v_1(x)} = \max\left(\sum_{i=1}^{N}\sum_{\mathrm{j}=1}^{\mathrm{M}}\frac{\mathrm{T_{ik}F_{ik}.x_{ijl}}}{\mathrm{f}} - 1, 0\right) \quad (7)$$

$$\mathrm{v_2(x)} = \max\left(\sum_{\mathrm{j}=1}^{\mathrm{M}}\sum_{l=1}^{\mathrm{L}}\mathrm{x_{ijl}} - 1, 0\right) \quad (8)$$

Therefore, we define the following fitness function with respect to the objective function (6) and the punishment functions (7) and (8):

$$\mathrm{f(x)} = \mathrm{O(x)} - \mathrm{C_1 v_1(x)} - \mathrm{C_2 v_2(x)} \quad (9)$$

where C_1 and C_2 are penalty coefficients and $\mathrm{x} = \mathrm{x_{ijl}}$.

Step 3: Update Best particles, *lbest*, Global Particle, *gbest*:

$$lbest_i^t = \begin{cases} X_i^t & if\ f\ \left(X_i^t\right) > f(lbest_i^t) \\ lbest_i^t & else \end{cases} \quad (10)$$

The *gbest* is replaced by the best *lbest* among the particles.

Step 4: Update *velocity* and move each particle to the new position:

$$V_i^{t+1} = \omega V_i^t + \mathrm{c_1}^*\varphi_1{}^*(lbest_i^t - X_i^t) + \mathrm{c_2}^*\varphi_2{}^*(gbest_i^t - X_i^t) \quad (11)$$

$$X_i^{t+1}. = X_i^t + V_i^{t+1} \quad (12)$$

where ω is an inertial weight that creates a balance between global exploration and local exploitation for each particle, and it will be damped in each iteration. $\mathrm{c_1}$ and $\mathrm{c_2}$ are the weights, that manage the particle's tendency towards its local best location, *lbest*, or swarm's best location, *gbest*. To determine the weights during the iterations, we employed the Time-Varying Acceleration Coefficients (TVAC) presented in [57], which is an efficient technique to speed up the algorithm. V_i^t is the velocity of the particles, and X_i^t is the particles' current position. Also, φ_1 and φ_2 are random numbers between 0 and 1.

Step 5: Stop, if convergence or *Termination condition* is satisfied. Otherwise, go to step 6

Termination condition: The algorithm should stop when it reaches a plateau. The terminating condition of our algorithm is: after 100 iterations, there is no progress for the best fitness value in the last 50 iterations, or the algorithm stops at the 400^{th} iteration.

Step 6: If the sub-algorithm's condition is satisfied, run the sub-algorithm (described in the next section), otherwise go to step 2.

B. PROPOSED SUB-ALGORITHM

In our previous work [29], the proposed algorithms were successful in terms of the utilization percentage for a higher number of players; however, they suffered for a small number of players. The results show almost 20 percent capacity wastage, for example, for 20 players' case. To address this issue and have more consistent results, we proposed a heuristic algorithm that can boost the main metaheuristic algorithm and achieve a steady result for various conditions. Since the main reason for this behaviour is a premature convergence of the proposed algorithms, we developed the regeneration part of the algorithms to avoid trapping in local minimum or maximum. The use of this re-initialization section can ensure us that no premature convergence takes place; however, to avoid increasing the complexity of the total algorithm we consider some conditions to bring this sub-algorithm in the main process loop only for some exceptional cases.

In two cases, this sub-algorithm may come to the calculation chain. Firstly, for the case that the algorithm is repeating a single positive fitness value in a large number of iterations. In this case, the algorithm may trap in a local minimum, and there is no progress in the results. Therefore, to avoid premature convergence, the sub-algorithm randomly transfers some loads from the server with high capacity wastage to the servers with low capacity wastage. It would be similar to mutation, but the action is more wisely. Secondly, the main algorithm is repeating a negative fitness value for a large number of initial iterations. This is the case that one or some servers are overloaded, and in this case, this sub-algorithm transfers part of the load of the overloaded server to the next servers that have enough capacity to handle this game's load. By employing this sub-algorithm, most likely, the main algorithm comes out from the premature convergence conditions. It is worth noting that this sub-algorithm is just boosting the main algorithm and is not looking for an optimum solution. The complexity of the sub-algorithm explained in the section VI-D shows that it does not increase the total complexity of the main algorithm. Fig. 5 describes our Boosted-PSO server selection method. The pseudo-code of the proposed Algorithm is as follows:

Fig. 6 compares the convergence of the PSO based algorithms with and without sub-algorithm for the same dataset. For the sake of page limitation, we are not showing the iterations of the algorithm. In this figure, for 20 players, the PSO starts with a good initial fitness value, and the Boosted-PSO converges to the higher value at the end. To attain this result, the sub-algorithm was called nine times into the main chain of the algorithm in iterations, where five of them created progress: iterations 30, 50, 60, 80, and 90. However, the previous PSO version does not experience any changes in fitness value after iteration 105, and it converges to the lower value. This comparison shows the behavior of the sub-algorithm for boosting the PSO and avoiding getting stuck in a local minimum.

Algorithm 1 Pseudo Code for Sub-Algorithm in Boosted-PSO

```
N = Number of Individuals (or Particles)
Iter = Current iteration number
Thrsh = Threshold for entering the sub-algorithm
if Iter > Thrsh && fitness.value (Iter ) == fitness.value
(Iter - Thrsh ) OR Iter > Thrsh /2 && fitness.value < 0
&& fitness.value (Iter ) == fitness.value (Iter – Thrsh /2 )
for i = 1 to N
  ➢ if fitness.value (individual i) <= 0
      ▪ for j = 1 to the number of players in individual i
          • if assigned server to the player j is overloaded
              ♦ Transfer the player j to the next server with
                enough capacity to process the game session
                of this player.
              ♦ Update capacity of the source and destination
                servers after moving.
          • end if
      ▪ end for
  ➢ else if fitness.value (individual i) > 0
      ▪ for j = 1 to the number of players in individual i
          • if the player j assigned to the server with maxi-
            mum capacity wastage.
              ♦ Transfer the player j to the next servers with
                lower capacity wastage and enough capacity
                to process the game sessions of this player.
              ♦ Update capacity of the source and destination
                servers after moving.
          ▪ end if
      • end for
  ➢ end if
Update position of individual i
Update fitness of individual i
Update velocity of individual i
end for
end if
```

C. PROPOSED GA MODEL

Genetic Algorithm is an evolutionary algorithm that becomes popular from the 1990s [58]. In this algorithm, the first individual chromosomes are created randomly as the initial population or generation. A fitness function that reflects the objectives and constraints of the problem will be used to score the chromosomes. The values of the fitness function of the current chromosomes determine which chromosomes are more powerful to be used for producing the next generation. The weak chromosomes will be eliminated, and the most powerful ones have a higher chance of staying alive. The flow chart depicted in Fig. 7 describes our proposed Boosted-GA server selection process, which has the following steps:

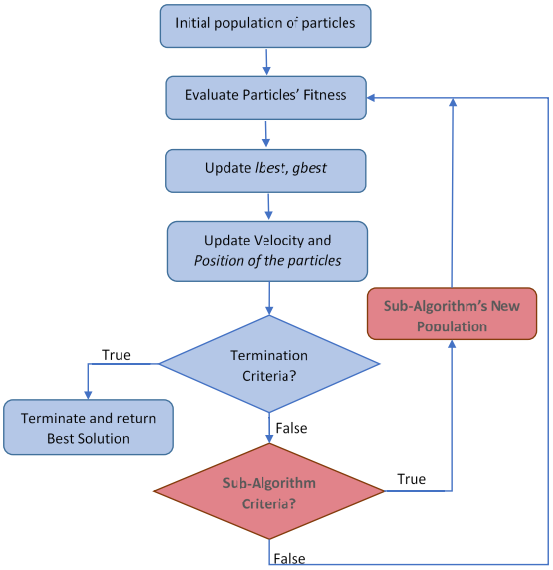

FIGURE 5. Boosted-PSO for server selection.

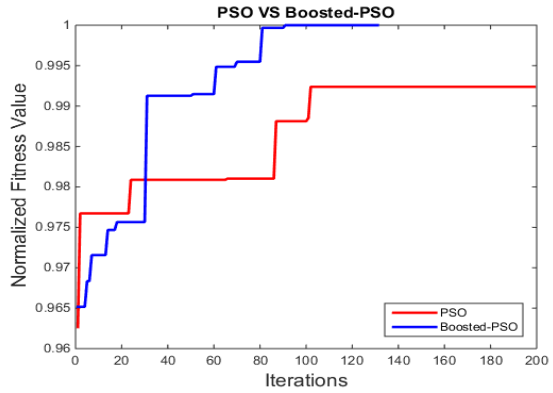

FIGURE 6. Convergence comparison between PSO and Boosted-PSO.

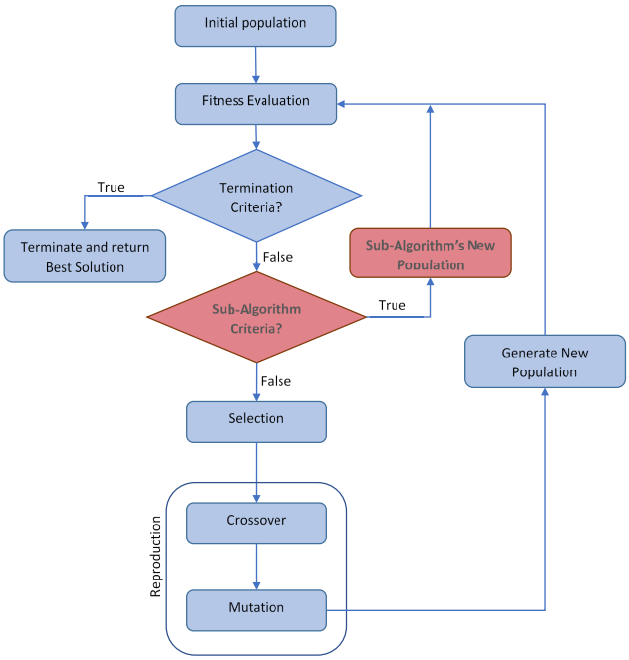

FIGURE 7. Overall Boosted-GA method for server selection.

Step 1. Initial Population: As with all GA models, we set the algorithm's parameters, such as choosing population size, mutation rate, and stopping condition. To produce an initial population for the GA model, we followed the same procedure as PSO, in section V-A, for generating the initial particles. The definition of a chromosome is Chromosome $= \{P_1, P_2 \ldots P_i \ldots P_N\}$. Each P_i symbolizes a gene, which is the basic unit of a genetic operator. We set the initial population to 50 and generate the first generation of chromosomes randomly.

Step 2. Fitness Evaluation: The fitness value should be allocated to each chromosome and considered in each decision. For the GA algorithm, we use the same fitness function presented in equation (9).

Step 3. Selection: After the generation of the initial population, the stronger chromosomes should be selected. These strong chromosomes will produce the next generation. We choose these individuals using a roulette selection operator, developed by Holland [59]. On this selection wheel, the size of the segment for each parent is proportional to its fitness. Obviously, the larger the fitness, i.e., the larger the segment size, the higher the selection's chance. During each successive generation, the roulette selection operator keeps both the utilization and the player's QoE for the next generation by preferentially selecting the best solutions with randomization. The probability P, for each individual chromosome i, is defined by:

$$P\,(Individual\ i) = \frac{f_i}{\sum_{j=1}^{PopSize} f_i} \tag{13}$$

where f_i equals the fitness of individual i.

Step 4. Reproduction: This step is to produce a second-generation population of solutions from those selected through genetic operators: crossover and mutation.

a) Crossover: We adopt the one-point crossover method where a gene is a basic unit to form the next generation, and two parents produce two offsprings (i.e., new solutions) that inherit information from both. Our algorithm randomly picks the crossover point in the parents' chromosomes. Then, it swaps the ends of the parents, from the crossover point to the end of the chromosome, to generate two new solutions. The percentage of the crossover in our algorithm is set to 50%.

b) Mutation: This operator is used to alter the value of a gene in the selected chromosomes randomly. For our case, changing the value of a gene turns the combination of the server assignment in the chromosome and produces a new solution. This new offspring will be added to the new population. The mutation ratio is the parameter that specifies the number of mutation chromosomes. The percentage of mutation in our proposed method is set to 50%.

Step 5. Termination: The algorithm should stop when it reaches a plateau. The terminating condition of our algorithm is: after 100 iterations, there is no progress for the best fitness value in the last 50 iterations, or the algorithm stops at the 400^{th} iteration. If the termination conditions have not been met, go to step 6.

Step 6. Check sub-algorithm criteria: If sub-algorithm's condition is satisfied, run the sub-algorithm, otherwise go to step 2.

Since the chromosome in GA and the particle in PSO have the same concept and structure in our problem, the same sub-algorithm described in section V-B is deployed in our proposed GA method. The performance of both proposed methods will be presented in the next section.

D. COMPUTATIONAL COMPLEXITY

For both algorithms, the complexity of computing the initial population and the fitness for one individual is O(QMS) and O(MS), respectively. Where Q is the number of populations, M is the number of players, and S is the total number of eligible servers. The maximum value of S is N× L. The sub-algorithm's complexity is O(QMS). The computation complexity to produce the particles in the Boosted-PSO is O(QMS + QMS); therefore, the total complexity is O(QMS). Since we limit the number of iterations to 'I', the maximum complexity of the Boosted-PSO will be O(IQMS). For GA, the computational complexity of selecting Q pairs of parents is O(QlogQ). The complexity of producing a new generation is O(QMS). Since the complexity of the sub-algorithm is the same, the maximum complexity of Boosted-GA would be O(I(QMS + QlogQ)). For an equal number of DCs, RSs, and players, since our population's size of the Boosted-PSO is half of the Boosted-GA, its maximum complexity would be around half, too. Consequently, our sub-algorithm does not increase the maximum complexity of the GA and PSO. However, as the results in the next section show, it causes a noticeable improvement in the efficiency of the algorithms.

VI. EXPERIMENTS AND RESULTS

In this section, we evaluate the proposed methods at two levels. Firstly, we compare them with each other and with the server selection method introduced in [29], using the same setting and frame rate set as mentioned there, i.e., F1 = {15, 30, 45, 60}. Secondly, we compare them with four greedy algorithms called First Fit Algorithm (FFA), Best Fit Algorithm (BFA), Next-Fit Algorithms (NFA), and Worst-Fit Algorithm (WFA). These are common and efficient solutions to the bin-packing NP-Hard problems used in Cloud Gaming and the other similar contexts. For example, WFA was used by Onlive for game placement [60]. Also, the concept of the BFA, FFA, and NFA employed in other cloud gaming studies [21], [38], [40]. In this part of the experiments, we employ a new frame rate set, i.e., F2 = {30, 45, 60, 90, 120}, which includes some higher and realistic frame rates for today's games. The simulations run on a Quade core Intel Core i7 3.2 GHz workstation with 12 GB of RAM.

A. COMPARISION WITH GA AND PSO

To compare the proposed algorithms with each other and with [29], we created four groups of datasets with a different number of players and processing loads. We assumed that there are 10 eligible datacenters that can satisfy the player's maximum tolerable latency, and only 20 servers in each datacenter are GPU-equipped to be eligible for using as an RS. To allocate a range of quality to the players regarding the problem's constraints presented in IV-A, we consider four popular used frame rates [61] in this experiment, F = {15, 30, 45, 60}. Besides, a uniform distribution of the processing load is randomly assigned to the frame rates to satisfy equation (2) in the utilization model. Also, we assumed that all GPUs are working at their maximum frequency, which is considered 500,000 cycles per second. The simulation runs for different number of players, P = {20, 40, 60, 80, 100}. As our problem formulation presented in section IV has two components, i.e., utilization and end user's experience, we consider both parameters to evaluate the efficiency of the algorithms. We examine the capacity wastage and utilization percentage, as comparison criteria in terms of GPU utilization, and the QoE value as a metric of the end user's experience. The evaluation's results are presented in fig 8 to 10.

Fig 8 (a) and (b) show the progress of both Boosted-GA and Boosted-PSO algorithms respectively, compared to the methods presented in [29], in terms of utilization percentage. From Fig. 8(a), we can see that compared to our

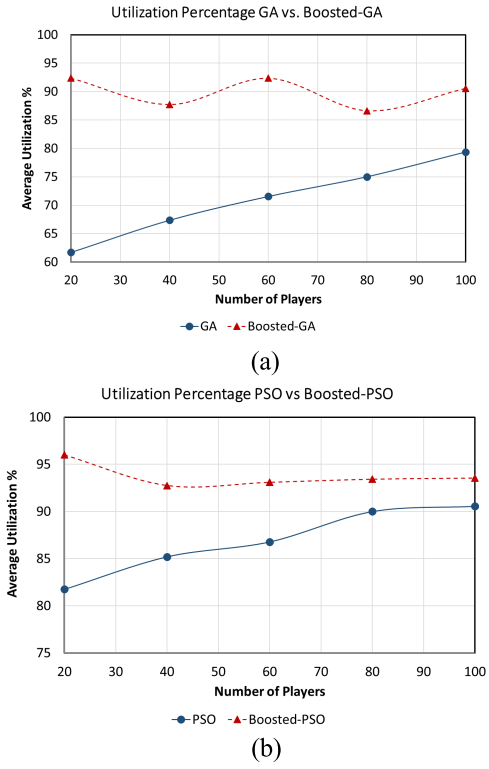

FIGURE 8. Progress in terms of utilization percentage compared with results achieved in [29] for (a) Boosted-GA and (b) Boosted-PSO methods.

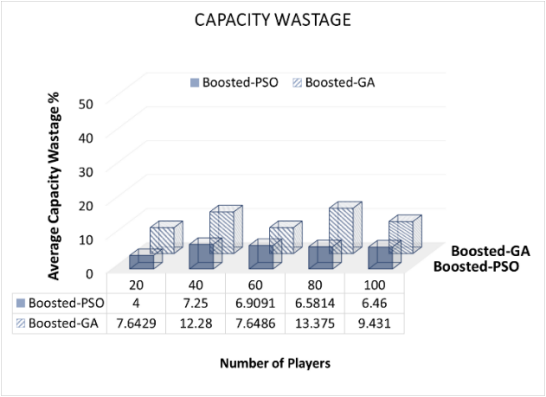

FIGURE 9. Capacity wastage of the Boosted-PSO and Boosted-GA algorithms for different number of players.

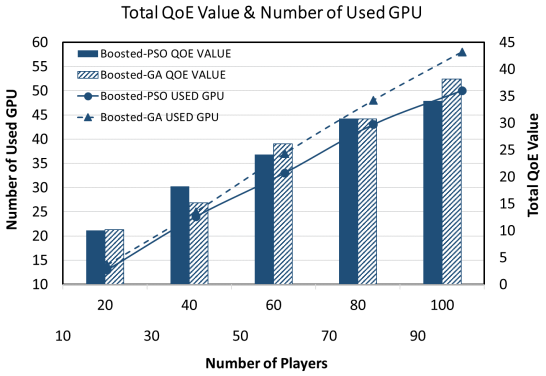

FIGURE 10. The number of utilized GPUs by the Boosted-PSO and Boosted-GA algorithms and corresponding QoE values for different number of players.

previous work, a considerable improvement happened for the GA-based method; e.g., around 30% improvement in utilization for 20 players, and more than 10% for 100 players. Similarly, the progress for the PSO-based method depicted in Fig. 8(b) is noticeable, especially for the small number of players. Another significant improvement is the consistency and steadiness of the utilization curve in the proposed methods versus the number of players, which makes these algorithms more reliable than our previous approach in [28].

To compare the proposed methods with each other, we consider two parameters such as capacity wastage and total QoE in figures 9 and 10 and show how these two parameters vary for the different number of players. In Fig. 9, we see that the Boosted-PSO has lower capacity wastage than the Boosted-GA. This figure shows the higher efficiency of Boosted-PSO with a 6.85% average capacity wastage compared to 11.09% for the Boosted-GA method.

Another evaluation has been done based on QoE and the number of used GPUs that service providers should pay for. Fig. 10 depicts the achieved QoE values and the number of used GPUs after server assignment for the different number of players. The curves in this figure show that the Boosted-PSO always allocates a smaller number of servers. Moreover, the close value of its QoE to the Boosted-GA method is shown in Fig. 10. Also, in some cases, e.g., 60 and 100 players, the QoE of the Boosted-GA is a little bit greater than PSO but at the expense of using several extra GPUs. For example, for 100 players, the Boosted-GA's QoE is higher by 4; however, it utilizes 8 more GPUs. For other cases, i.e., 20, 40, and 80, Boosted-PSO has the same or higher QoE while using a lower number of GPUs.

B. COMPARISION WITH FOUR COMMON BIN-PACKING ALGORITHMS

To evaluate the efficiency of these algorithms in terms of utilization, we compare them with four greedy algorithms that are well-known for solving bin-packing problems, FFA, BFA, NFA, and WFA. We used the server selection's data of our proposed algorithms as inputs of the above greedy processes and compared their results with our proposed metaheuristic algorithms. Through these experiments, we can notice if there is a better assignment which is not considered by our algorithms and perceive how they are weaker or stronger than these four popular methods.

Fig. 11 to Fig. 14 compare the results of the Boosted-GA and Boosted-PSO algorithms with all of these greedy algorithms. We do these evaluations using the frame rates employed in [29]. The frame rate bounds in [29] are 15 to 60 fps. The 15 fps is used because it is a comparison baseline in some other studies, such as [25], [61]. Today, however, 30 fps to 120 fps is more realistic. Therefore, the frame rate set used in our experiments in this section is F2 = {30,45,60,90,120}. To pass the constraint (b) of the problem statement for F2, the maximum frequency of the GPUs is set to 1.2GHz. The setting of this experiment includes the number of eligible datacenters and servers is the same as section VI-A.

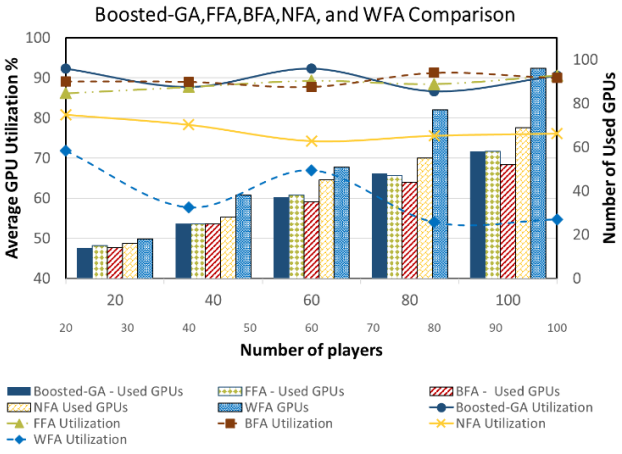

FIGURE 11. Comparison among Boosted-GA, FFA, BFA, NFA and WFA methods in terms of the Average GPU Utilization and the number of utilized GPUs, for different number of players. Using F1 frame set.

Fig. 11, and Fig. 12 are the experimental results for F1, and Fig. 13 and Fig. 14 represent the evaluations for F2. Fig. 11 shows that the Boosted-GA method's efficiency is comparable to BFA and FFA. However, in most of the cases,

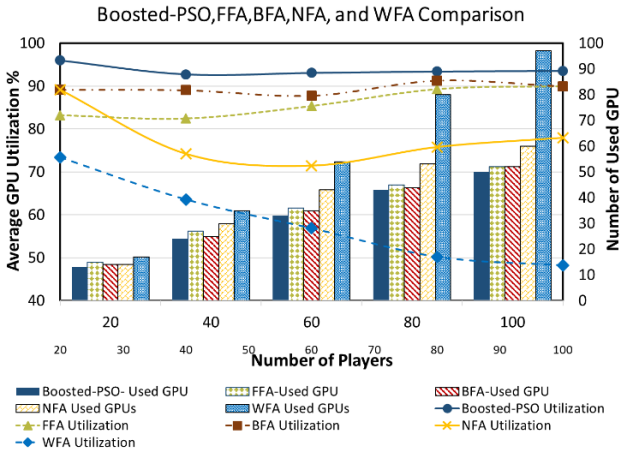

FIGURE 12. Comparison among Boosted-PSO, FFA, BFA, NFA and WFA methods for different number of players in terms of the Average GPU Utilization and the number of utilized GPUs. Using F1 frame set.

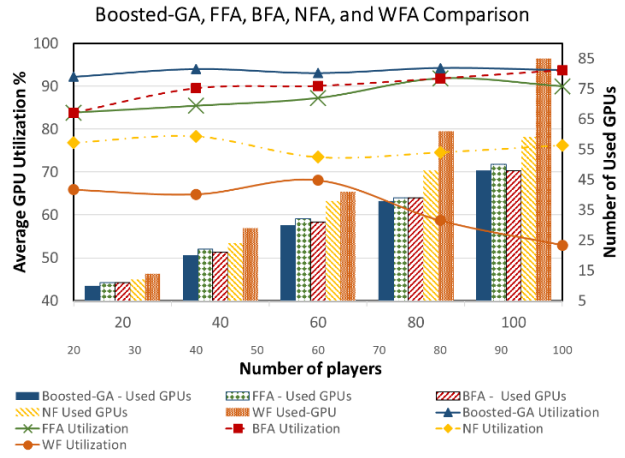

FIGURE 13. Comparison among Boosted-GA, FFA, BFA, NFA and WFA methods in terms of the Average GPU Utilization and the number of utilized GPUs, for different number of players. Using F2 frame set.

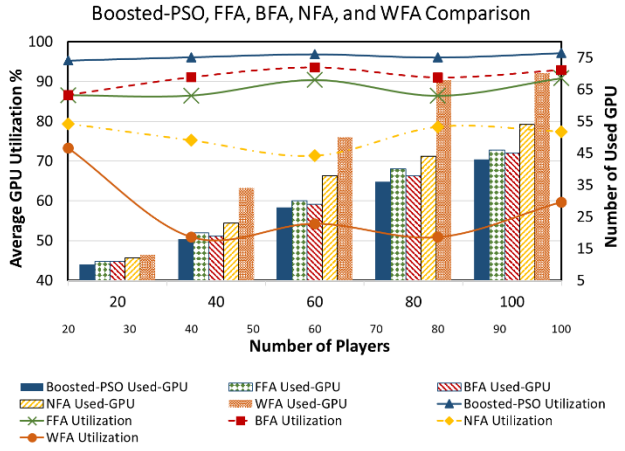

FIGURE 14. Comparison among Boosted-PSO, FFA, BFA, NFA and WFA methods for different number of players in terms of the Average GPU Utilization and the number of utilized GPUs. Using F2 frame set.

the Boosted-GA outperforms the BFA and FFA. Regarding Fig. 13, the Boosted-GA shows better efficiency for the new frame rate set compared to the greedy algorithms.

The results in Fig. 12 show that BFA is more efficient than FFA, NFA, and WFA. However, none of them can meet the Boosted-PSO's results. Similar behavior presented for F2 results in Fig.14. For all different number of players, records of the Boosted-PSO are stronger than the four other algorithms. In summary, the Boosted-PSO method always has a higher utilization percentage and a lower number of utilized GPUs. The NFA and WFA represent the lowest efficiency in these experiments. By increasing the number of players, the WFA faces noticeable efficiency degradation, which is compatible with the experiments' results conducted by Claypool *et al.* in [60]. This algorithm suffers from the over-provisioning of the resources.

The Boosted-PSO algorithm in MATLAB and on the test computer, which is mentioned in section VI, takes around 55 seconds to converge for 20 players with 200 servers. By increasing the number of players, this time will increase linearly. This processing time is based on the said processing power, which is remarkably less than a professional cloud server. Furthermore, the programing language and implementation techniques matter. A study on the speed of multiple programming languages in [62] shows that MATLAB is between 9 to 11 times slower than C++ for computing. Therefore, to have a practical sense of the convergence time, these points should be taken into account.

Overall, with these comparisons, the Boosted-PSO method shows its superiority to our previous methods and all Boosted-GA, BFA, FFA, NFA, and WFA in terms of Utilization and the number of utilized GPUs. Also, it has a minimum fluctuation and shows a higher level of stability to variations of the number of players.

VII. CONCLUSION

In CG, finding an appropriate resource allocation method to satisfy the service provider's profit and the end user's quality of experience simultaneously, is still a challenge. In this work, we address this challenge by presenting a server selection method that considers the provider's and the client's concerns together. The presented objective maximizes GPU utilization while increasing the player's experience. To solve the problem, we proposed two metaheuristic algorithms, Boosted-PSO and Boosted-GA. The proposed methods considerably compensate for the lack of efficiency in our previous work, especially for the low number of players. We evaluated the boosted algorithms in two aspects, utilization and player's experience. Also, we compared the proposed algorithms to the GA and PSO methods presented in [29] and four popular bin-packing algorithms. The simulation results showed that our Boosted-PSO method achieves the highest efficiency among the other methods, including Boosted-GA, FFA, BFA, NFA, and WFA in terms of GPU utilization, capacity wastage, and player's QoE. Also, it has remarkable stability for the different number of players. In our future work, we will consider other quality metrics and also the network parameters to evaluate the problem from the other aspects and to attain a more comprehensive solution.

REFERENCES

[1] *The Global Games Market Will Generate $152.1 Billion in 2019|Newzoo*. Accessed: Mar. 20, 2020. [Online]. Available: https://newzoo.com/insights/articles/the-global-games-market-will-generate-152-1-billion-in-2019-as-the-u-s-overtakes-china-as-the-biggest-market/

[2] M. Semsarzadeh, M. Hemmati, A. Javadtalab, A. Yassine, and S. Shirmohammadi, "A video encoding speed-up architecture for cloud gaming," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops (ICMEW)*, Jul. 2014, pp. 1–6.

[3] *Sony Says Cloud Gaming Could be Future Threat to PlayStation Business—Business Insider*. Accessed: Mar. 20, 2020. [Online]. Available: https://www.businessinsider.com/sony-says-cloud-gaming-future-threat-to-playstation-business-2019-2

[4] *Global Cloud Gaming Market Will Reach USD 6, 944 Million By 2026: Zion Market Research*. Accessed: Mar. 20, 2020. [Online]. Available: https://www.globenewswire.com/news-release/2018/12/31/1679151/0/en/Global-Cloud-Gaming-Market-Will-Reach-USD-6-944-Million-By-2026-Zion-Market-Research.html

[5] Sony Computer Entertainment to Acquire Gaikai Inc. *A Leading Interactive Cloud Gaming Company SCE To Build A Cloud Service Bringing Gaikai's Cloud Based-Streaming Technologies into Its Network Business*. Accessed: Mar. 20, 2020. [Online]. Available: https://www.sie.com/en/corporate/release/2012/120702.html

[6] *OnLive*. Accessed: Mar. 20, 2020. [Online]. Available: http://onlive.com/

[7] *PlayStation Now Game-Streaming Service Coming Summer 2014 (Update)—Polygon*. Accessed: Mar. 20, 2020. [Online]. Available: https://www.polygon.com/2014/1/7/5284504/sony-playstation-now-gaikai-based-streaming-service-ps-ps2-ps3

[8] *Company Profile | Broadmedia GC Corporation*. Accessed: Mar. 20, 2020. [Online]. Available: https://www.broadmediagc.co.jp/en/corporation/

[9] *Stadia Founder's Edition—One Place for All the Ways We Play—Google Store*. Accessed: Mar. 20, 2020. [Online]. Available: https://store.google.com/ca/product/stadia_learn

[10] *PlayStation Now—Online Streaming Services on PS4 or PC—PlayStation*. Accessed: Mar. 20, 2020. [Online]. Available: https://www.playstation.com/en-ca/explore/playstation-now/

[11] *Game Streaming—Play Games With Friends|Parsec*. Accessed: Mar. 20, 2020. [Online]. Available: https://parsecgaming.com/

[12] *Vortex—Cloud Gaming for Android, PC and macOS*. Accessed: Mar. 20, 2020. [Online]. Available: https://vortex.gg/

[13] *Game Anywhere on Your Mac, Windows PC, or SHIELD Device With NVIDIA's Cloud Gaming Service*. Accessed: Mar. 20, 2020. [Online]. Available: https://www.nvidia.com/en-us/geforce/products/geforce-now/

[14] *PLAYKEY|Games Online*. Accessed: Mar. 20, 2020. [Online]. Available: https://welcome.playkey.net/en/lp/eu-quiz-before/

[15] S. Shirmohammadi, M. Abdalla, D. T. Ahmed, K.-T. C. A. K. A. S. Chen, Y. Lu, and A. Snyatkov, "Introduction to the special section on visual computing in the cloud: Cloud gaming and virtualization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 12, pp. 1955–1959, Dec. 2015.

[16] K.-T. Chen, C.-Y. Huang, and C.-H. Hsu, "Cloud gaming onward: Research opportunities and outlook," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops (ICMEW)*, Jul. 2014, pp. 1–4.

[17] M. Amiri, H. A. Osman, S. Shirmohammadi, and M. Abdallah, "Toward delay-efficient game-aware data centers for cloud gaming," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 12, no. 5s, pp. 1–19, Sep. 2016.

[18] Y. Chen, J. Liu, and Y. Cui, "Inter-player delay optimization in multiplayer cloud gaming," in *Proc. IEEE 9th Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2016, pp. 702–709.

[19] M. Amiri, A. Sobhani, H. Al Osman, and S. Shirmohammadi, "SDN-enabled game-aware routing for cloud gaming datacenter network," *IEEE Access*, vol. 5, pp. 18633–18645, Sep. 2017.

[20] K.-T. Chen, Y.-C. Chang, P.-H. Tseng, C.-Y. Huang, and C.-L. Lei, "Measuring the latency of cloud gaming systems," in *Proc. 19th ACM Int. Conf. Multimedia (MM)*, 2011, p. 1269.

[21] Y. Deng, Y. Li, X. Tang, and W. Cai, "Server allocation for multiplayer cloud gaming," in *Proc. 24th ACM Int. Conf. Multimedia ACM*, 2016, pp. 918–927.

[22] H. Ahmadi, S. Khoshnood, M. R. Hashemi, and S. Shirmohammadi, "Efficient bitrate reduction using a game attention model in cloud gaming," in *Proc. IEEE Int. Symp. Haptic Audio Vis. Environ. Games (HAVE)*, Oct. 2013, pp. 103–108.

[23] E. Babaei, M. R. Hashemi, and S. Shirmohammadi, "A state-based game attention model for cloud gaming," in *Proc. 15th Annu. Workshop Netw. Syst. Support Games (NetGames)*, Jun. 2017, pp. 34–36.

[24] H. Dar, J. Kwan, Y. Liu, O. Pantazis, and R. Sharp, "The game performance index for mobile phones," 2019, *arXiv:1910.13872*. [Online]. Available: http://arxiv.org/abs/1910.13872

[25] R. Ellerweg, "Make frame rate studies useful for system designers," in *Proc. Int. Conf. Graph. Interact. (ICGI)*, Nov. 2018, pp. 1–8.

[26] M. Claypool, K. Claypool, and F. Damaa, "The effects of frame rate and resolution on users playing first person shooter games," in *Proc. Multimedia Comput. Netw.*, Jan. 2006, Art. no. 607101.

[27] J. A. Berton and K.-L. Chuang, "Effects of very high frame rate display in narrative CGI animation," in *Proc. 20th Int. Conf. Inf. Vis. (IV)*, Jul. 2016, pp. 395–398.

[28] Y. Gao, Y. Xue, and J. Li, "Utilization-aware allocation for multi-tenant datacenters," in *Proc. IEEE 33rd Int. Conf. Distrib. Comput. Syst. Workshops*, Jul. 2013, pp. 82–87.

[29] H. E. Dinaki and S. Shirmohammadi, "GPU/QoE-aware server selection using Metaheuristic algorithms in multiplayer cloud gaming," in *Proc. 16th Annu. Workshop Netw. Syst. Support Games (NetGames)*, Jun. 2018, pp. 1–6.

[30] E. Dhib, K. Boussetta, N. Zangar, and N. Tabbane, "Cost-aware virtual machines placement problem under constraints over a distributed cloud infrastructure," in *Proc. 6th Int. Conf. Commun. Netw. (ComNet)*, Mar. 2017, pp. 1–5.

[31] A. Laghrissi and T. Taleb, "A survey on the placement of virtual resources and virtual network functions," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1409–1434, 2nd Quart., 2019.

[32] W. Cai, M. Chen, and V. C. M. Leung, "Toward gaming as a service," *IEEE Internet Comput.*, vol. 18, no. 3, pp. 12–18, May 2014.

[33] X. Nan, X. Guo, Y. Lu, Y. He, L. Guan, S. Li, and B. Guo, "Delay–rate–distortion optimization for cloud gaming with hybrid streaming," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 12, pp. 2687–2701, Dec. 2017.

[34] S. Choy, B. Wong, G. Simon, and C. Rosenberg, "A hybrid edge-cloud architecture for reducing on-demand gaming latency," *Multimedia Syst.*, vol. 20, no. 5, pp. 503–519, Oct. 2014.

[35] M. Claypool, "Motion and scene complexity for streaming video games," in *Proc. 4th Int. Conf. Found. Digit. Game (ACM)*, 2009, pp. 34–41.

[36] P. B. Beskow, P. Halvorsen, and C. Griwodz, "Latency reduction in massively multi-player online games by partial migration of game state," in *Proc. 2nd Int. Conf. Internet Technol. Appl.*, Wales, U.K., 2007, pp. 153–163.

[37] Y. Gao, L. Wang, and J. Zhou, "Cost-efficient and quality of experience-aware provisioning of virtual machines for multiplayer cloud gaming in geographically distributed data centers," *IEEE Access*, vol. 7, pp. 142574–142585, Oct. 2019.

[38] Y. Deng, Y. Li, R. Seet, X. Tang, and W. Cai, "The server allocation problem for session-based multiplayer cloud gaming," *IEEE Trans. Multimedia*, vol. 20, no. 5, pp. 1233–1245, May 2018.

[39] X. Zhang, H. Chen, Y. Zhao, Z. Ma, Y. Xu, H. Huang, H. Yin, and D. O. Wu, "Improving cloud gaming experience through mobile edge computing," *IEEE Wireless Commun.*, vol. 26, no. 4, pp. 178–183, Aug. 2019.

[40] Y. Li, Y. Deng, X. Tang, W. Cai, X. Liu, and G. Wang, "Cost-efficient server provisioning for cloud gaming," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 14, no. 3s, pp. 1–22, Jun. 2018.

[41] K.-T. Chen, Y.-C. Chang, H.-J. Hsu, D.-Y. Chen, C.-Y. Huang, and C.-H. Hsu, "On the quality of service of cloud gaming systems," *IEEE Trans. Multimedia*, vol. 16, no. 2, pp. 480–495, Feb. 2014.

[42] H.-J. Hong, D.-Y. Chen, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "Placing virtual machines to optimize cloud gaming experience," *IEEE Trans. Cloud Comput.*, vol. 3, no. 1, pp. 42–53, Jan. 2015.

[43] H. Tian, D. Wu, J. He, Y. Xu, and M. Chen, "On achieving cost-effective adaptive cloud gaming in geo-distributed data centers," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 12, pp. 2064–2077, Dec. 2015.

[44] S. Morishima, M. Okazaki, and H. Matsutani, "A case for remote GPUs over 10 GbE network for VR applications," in *Proc. 8th Int. Symp. Highly Efficient Accel. Reconfigurable Technol.*, 2017, vol. 6.

[45] F. Metzger, A. Rafetseder, and C. Schwartz, "A comprehensive end-to-end lag model for online and cloud video gaming," in *Proc. 5th ISCA/DEGA Workshop Perceptual Qual. Syst. (PQS)*, Aug. 2016, pp. 15–19.

[46] *GitHub—Mas-Ude/Onlinegame-Lag-Sim*. Accessed: Mar. 20, 2020. [Online]. Available: https://github.com/mas-ude/onlinegame-lag-sim

[47] T. Zinner, O. Hohlfeld, O. Abboud, and T. Hossfeld, "Impact of frame rate and resolution on objective QoE metrics," in *Proc. 2nd Int. Workshop Qual. Multimedia Exp. (QoMEX)*, Jun. 2010, pp. 29–34.

[48] *FFmpeg*. Accessed: Mar. 20, 2020. [Online]. Available: http://ffmpeg.org/

[49] I. Slivar, L. Skorin-Kapov, and M. Suznjevic, "Cloud gaming qoe models for deriving video encoding adaptation strategies," in *Proc. 7th Int. Conf. Multimedia Syst. (MMSys)*, 2016, pp. 185–196.
[50] George Millington. AMD Press Release. *New AMD RadeonTM Pro V340 Graphics Card Delivers Accelerated Performance and High User Density to Power Datacenter Visualization Workloads*. Accessed: Aug. 26, 2018. [Online]. Available: https://www.globenewswire.com/news-release/2018/08/26/1556638/0/en/New-AMD-Radeon-Pro-V340-Graphics-Card-Delivers-Accelerated-Performance-and-High-User-Density-to-Power-Datacenter-Visualization-Workloads.html
[51] M. R. Johnson and D. S. Garey, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA, USA: Freeman, 1979.
[52] E. G. Cooman, M. R. Garey, and D. S. Johnson, "Approximation algorithms for bin packing: A survey," in *Approximation Algorithms for NP-hard problems*. 1996, pp. 46–93.
[53] E. Falkenauer and A. Delchambre, "A genetic algorithm for bin packing and line balancing," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 1992, pp. 1186–1192.
[54] R. Kennedy and J. Eberhart, "Particle swarm optimization (PSO)," in *Proc. IEEE Int. Conf. Neural Netw.*, Perth, WA, Australia, Nov. 1995, pp. 1942–1948.
[55] H. Moayedi, M. Mehrabi, M. Mosallanezhad, A. S. A. Rashid, and B. Pradhan, "Modification of landslide susceptibility mapping using optimized PSO-ANN technique," *Eng. Comput.*, vol. 35, no. 3, pp. 967–984, Jul. 2019.
[56] A. Jaafari, E. K. Zenner, M. Panahi, and H. Shahabi, "Hybrid artificial intelligence models based on a neuro-fuzzy system and Metaheuristic optimization algorithms for spatial prediction of wildfire probability," *Agricult. Forest Meteorol.*, vols. 266–267, pp. 198–207, Mar. 2019.
[57] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 240–255, Jun. 2004.
[58] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Mach. Learn.*, vol. 3, nos. 2–3, pp. 95–99, 1988.
[59] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: MI Univ. Michigan Press, 1975.
[60] D. Finkel, M. Claypool, S. Jaffe, T. Nguyen, and B. Stephen, "Assignment of games to servers in the OnLive cloud game system," in *Proc. Annu. Workshop Netw. Syst. Support Games*, 2014.
[61] R. M. Nasiri, J. Wang, A. Rehman, S. Wang, and Z. Wang, "Perceptual quality assessment of high frame rate video," in *Proc. IEEE 17th Int. Workshop Multimedia Signal Process. (MMSP)*, Oct. 2015, pp. 1–6.
[62] S. B. Aruoba and J. Fernández-Villaverde, "A comparison of programming languages in economics," Nat. Bur. Econ. Res., Cambridge, MA, USA, White Paper w20263, 2014.

HOSSEIN EBRAHIMI DINAKI (Member, IEEE) is currently pursuing the Ph.D. degree with the Department of Electrical Engineering and Computer Science, University of Ottawa, ON, Canada. His research interests include optimization, applied AI, QoE-aware resource management, network diagnosis, and cloud gaming. He has expertise in evolutionary algorithms and deep artificial neural networks. He is a member of the ACM. Also, he has professional experience on frequency and network planning of digital video broadcasting.

SHERVIN SHIRMOHAMMADI (Fellow, IEEE) received the Ph.D. degree in electrical engineering from the University of Ottawa, Canada.

He is currently a Professor with the School of Electrical Engineering and Computer Science, University of Ottawa. He is also the Director of the Distributed and Collaborative Virtual Environment Research Laboratory, doing research in applied AI for multimedia systems and networks, specifically video systems, gaming systems, and multimedia-assisted healthcare systems. The results of his research, funded by more than $14 million from public and private sectors, have led to over 350 publications, three best paper awards, over 70 researchers trained at the postdoctoral, Ph.D., and master's levels, holds over 20 patents and technology transfers to the private sector, and a number of awards. He is a Fellow of the IEEE for contributions to multimedia systems and network measurements and a Lifetime Senior Member of the ACM. He was the winner of the 2019 George S. Glinski Award for Excellence in Research and the University of Ottawa Gold Medalist. He is also a licensed Professional Engineer in Ontario. He is the Editor-in-Chief of the IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT, and an Associate Editor of *ACM Transactions on Multimedia Computing, Communications, and Applications*, having been numerously recognized as the Associate Editor of the year by both of these and other journals.

MAHMOUD REZA HASHEMI (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in electrical engineering from the University of Tehran, Tehran, Iran, in 1989 and 1993, respectively, and the Ph.D. degree from the University of Ottawa, ON, Canada, in 2001. He is currently a tenured Associate Professor with the School of Electrical and Computer Engineering, University of Tehran. He is also a Co-Founder and the Director of the Multimedia Processing Laboratory (MPL), University of Tehran. His research interests include multimedia systems and networking, security and trust in cloud computing, and cloud gaming. He has also served as a member of technical committees of the IEEE International Conference on Multimedia and Expo (ICME), the IEEE International Symposium on Multimedia, and ACM Multimedia Systems. He was a recipient of the ICME Quality Reviewer Awards, in 2011 and 2013. He is an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY.