

Dynamic Resource Allocation for Metaverse Applications with Deep Reinforcement Learning

Nam H. Chu¹, Diep N. Nguyen¹, Dinh Thai Hoang¹, Khoa T. Phan²,
Eryk Dutkiewicz¹, Dusit Niyato³, and Tao Shu⁴

¹School of Electrical and Data Engineering, University of Technology Sydney, Australia

²School of Engineering and Mathematical Sciences, La Trobe University, Melbourne, Australia

³School of Computer Science and Engineering, Nanyang Technological University, Singapore

⁴Department of Computer Science and Software Engineering, Auburn University, USA

Abstract—This work proposes a novel framework to dynamically and effectively manage and allocate different types of resources for Metaverse applications, which are forecasted to demand massive resources of various types that have never been seen before. Specifically, by studying functions of Metaverse applications, we first propose an effective solution to divide applications into groups, namely MetaInstances, where common functions can be shared among applications to enhance resource usage efficiency. Then, to capture the real-time, dynamic, and uncertain characteristics of request arrival and application departure processes, we develop a semi-Markov decision process-based framework and propose an intelligent algorithm that can gradually learn the optimal admission policy to maximize the revenue and resource usage efficiency for the Metaverse service provider and at the same time enhance the Quality-of-Service for Metaverse users. Extensive simulation results show that our proposed approach can achieve up to 120% greater revenue for the Metaverse service providers and up to 178.9% higher acceptance probability for Metaverse application requests than those of other baselines.

Index Terms—Metaverse, deep reinforcement learning, semi-Markov decision process, network slicing.

I. INTRODUCTION

Although the Metaverse paradigm was first introduced in 1992, the recent efforts from big companies (e.g., Facebook, NVIDIA, and Microsoft) have made the Metaverse to be one of the most active fields for both academia and industry [1]. The major difference between the Metaverse and existing virtual worlds (e.g., Pokemon Go and Second Life) is that Metaverse can be recognized as the seamless integration of multiple virtual worlds [1]. Specifically, conventional virtual worlds limit the users' experiences in their specific environment, whereas the Metaverse allows users to seamlessly move between virtual worlds by a unified representation (e.g., an avatar). Therefore, similar to our real lives, users in Metaverse can preserve their belongings and appearance regardless of their application that they are experiencing. Moreover, the Metaverse is likely to blend the physical world and the digital world with the aid of innovative technologies, such as Digital Twin and Extended Reality (XR) [1]. Thus, the Metaverse is expected to revolutionize various aspects of our lives, such as healthcare, smart industries, and e-commerce.

However, to fulfil the user experience and Quality-of-Service (QoS) requirements, the Metaverse indeed requires extremely intensive resource demands that have never been seen before. First, the comprehensive integration of XR in the Metaverse demands enormous data collected from perceived networks, e.g., the Internet of Things (IoT), intensive computing for rendering three-dimensional objects, and ultra-high-speed and low-latency connections for guaranteeing the QoS and seamless user experience. Second, millions of users are expected to join the Metaverse concurrently, making the network data usage increase more than 20 times [2]. Third, the Metaverse puts new constraints on networking. In particular, in the current online system (e.g., massive multiplayer online games), the downlink requires a much higher throughput than that of the uplink [3]. In contrast, the Metaverse likely demands intensively high throughput for both down and up connections. It stems from the fact that users can create, share, and trade their assets to other users in any virtual world in the Metaverse. Thus, the Metaverse will demand paramount resources exceeding those of any existing online platform [1]. As a result, resource management in Metaverse is one of the biggest challenges hindering the deployment of Metaverse.

To address the Metaverse resource management, utilizing the multi-tier cloud computing architect can be considered as a promising solution. First, multi-tier computing can relieve the burden of massive resource demands on end-users for running Metaverse applications. Second, the multi-tier architecture can mitigate point-of-congestion problems of the centralized computing resource allocation architecture, where all resources are gathered and allocated from a centralized node. Third, the distribution of resources (e.g. computing, storage, and networking) along the path from end-users to the cloud can reduce the stress due to the enormous amount of data exchanged by the Metaverse operation over the networks. Finally, moving resources nearer to end-users results in decreasing delay, which is one of the most important factors in user experiences [4]. Thus, the multi-tier computing architecture can be considered to be the most suitable solution for the Metaverse.

Since the Metaverse is only at the beginning stage, only a few research works consider resource management [5]–[8]. In [5], the authors address the problem of allocating

resources by considering an edge-computing architecture to allocate computing resource to nearby Metaverse users. The work in [5] is extended in [6] and [7] by considering different types of resources (e.g., storage and radio). Differently, the authors in [8] consider the resource allocation for perception networks (e.g., the IoT) that are employed to get real-world data for the Metaverse applications. Note that none of the above works investigates the multi-tier computing architecture for resource allocation in Metaverse. Instead, they only consider a single-tier computing architecture, which is unable to facilitate the Metaverse’s massive resource demand. In addition, it can be observed that applications of Metaverse may share some same functions. For example, in practice, many applications, e.g., the Walking Dead: Our World and Pokemon Go, are currently using the same functions provided by the Google Maps API [9]. Thus, resource utilization can increase dramatically if a common function can be shared between applications. Nevertheless, all of the above studies are unable to take advantage of this aspect to maximize resource usage efficiency. Moreover, the resource demands of Metaverse are highly dynamic and uncertain since users can come and leave at any time, making resource management based on conventional optimization methods ineffective. Therefore, effective solutions to address these problems are urgently needed.

To address the above challenges, this paper proposes a novel framework that can automatically and intelligently manage various resource types of the underlying multi-tier computing architecture to maximize the performance of the Metaverse system. First, we propose a new application decomposition technique for Metaverse applications, by which functions of a Metaverse application can be executed separately at different tiers of the computing architecture depending on the available resources of each tier and the requirements of these functions. As such, this technique can leverage resources at different tiers simultaneously, thereby providing a flexible and high efficient solution for managing Metaverse applications. Second, we propose a novel paradigm, namely MetaInstance, that can exploit the similarities of Metaverse applications to maximize resource utilization. Third, we develop a highly-effective framework based on the semi-Markov decision process to capture the real-time property of the Metaverse together with a self-learning algorithm based on deep reinforcement learning to automatically learn the optimal policy for the system under the resource demand’s uncertainty and dynamic. Finally, we perform extensive simulation and show that our proposed solution clearly outperforms other baseline approaches.

II. DYNAMIC MULTI-TIERS RESOURCE ALLOCATION ARCHITECTURE FOR METAVERSE

As illustrated in Fig. 1, this work considers a Metaverse system managed by a service provider, where multiple Metaverse applications (e.g., healthcare and education) can operate simultaneously. To address the massive resource demands of Metaverse, we propose a novel multi-tier resource management framework that can dynamically and effectively distribute various resource types for Metaverse’s applications.

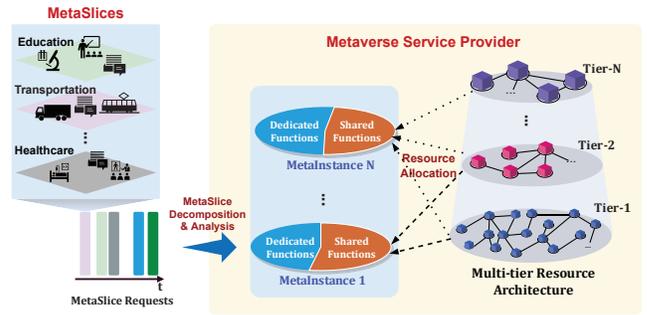


Fig. 1: The system model of the proposed framework, in which various resource types at different tiers can be simultaneously utilized and shared to create MetaSlices.

Specifically, we consider that the system has P types of resources (e.g., networking, storage, and computing), which are distributed at different tiers along the way from end-users to the cloud. By doing so, this framework can offer a more flexible, efficient, and robust solution for deploying Metaverse applications compared to that of the centralized cloud architecture. It can be observed that a Metaverse application, namely MetaSlice, consists of independence functions, and thus we propose an application decomposition technique, in which functions of an application can be separately created at different tiers depending on their requirement and the available resources at each tier. For example, a MetaSlice for navigation may include a driving assistant, real-time traffic, and a digital map. Since the driving assistant and real-time traffic require low delay, they can be created at a low tier (e.g., tier-1). In contrast, the digital map has a low update frequency, and thus it can be created at a higher tier. Thus, the application decomposition can provide an effectively and flexibly way for deploying MetaSlices. The decomposition of applications can be done by existing methods, e.g., [10].

We consider that there are G classes of MetaSlices according to their characteristics, e.g., QoS, user experience, and technical requirements. In addition, the system can host various MetaSlice types simultaneously, e.g., education, navigation, and tourism. As analyzed in the previous section, concurrent MetaSlices in the system may share some common functions. If common functions are shared among MetaSlices, a lot of resources can be saved, thereby increasing system resource usage efficiency and revenue of the provider. To that end, this study proposes to group MetaSlices into clusters, i.e., MetaInstances. As illustrated in Fig. 1, a MetaInstance can be determined by shared functions and dedicated functions of specific MetaSlices. To manage MetaSlices, we consider that each MetaInstance maintains a function description, which is updated whenever a MetaSlice is added or departs. Technically, a MetaSlice can be deployed in a similar way as that of a network slice in 5G network slicing [11]. However, their functions and the ways they are implemented are very different. Specifically, the 5G network slicing aims to provide various types of end-to-end connections from mobile users to service providers, e.g., ultra-reliable low-latency communi-

cations (uRLLC) and enhanced Mobile Broadband (eMBB). In network slicing, a network slice is a logical network built on top of a physical network to support one connection type (e.g., uRLLC and eMBB). Each network slice is created based on multiple predefined functions of 5G network providers. On the other hand, our proposed solution offers an effective and flexible approach to implementing and managing the Metaverse applications. In particular, our proposed framework first decomposes a MetaSlice (i.e., Metaverse application) into independent functions. Then, each function will be allocated different types of resources (e.g., radio, computing, and storage) at different network tiers regarding its requirements. In addition, because functions' serving capabilities are limited, the user experience may be degraded if too many MetaSlices share the same function. Therefore, this work considers that the maximum of N_L MetaSlices can share one function simultaneously. The above analysis demonstrates that the proposed framework can not only benefit the provider by maximizing resource usage efficiency but also offer a better user experience and QoS for end-users. To obtain these results, MetaSlice admission control and resource management play the key roles. On the one hand, accepting/rejecting a MetaSlice request determines the provider's revenue and the user's QoS (e.g., service availability). On the other hand, better resource utilization can help the provider to save more resources required to host future MetaSlices, thereby increasing its long-term revenue. Thus, this paper focuses on the admission control and resource management in the proposed framework.

Upon a MetaSlice request arrives, the Metaverse system makes a decision (i.e., accept or reject) based on the system's available resources and the requested MetaSlice's required resources and class. If the request of MetaSlice m is accepted, the system will examine the accepted MetaSlice to determine its similarities with ongoing MetaInstances as follows. Let the functions in a MetaSlice m be denoted by a function vector $\mathbf{f}_m \triangleq \{F_m^f\}_{f=1}^K$ where K is the total number of functions supported by the system and F_m^f represents an appearance of function f in this MetaSlice, i.e., $F_m^f = 1$ if MetaSlice m uses function f , and $F_m^f = 0$, otherwise. Similarly, the function vector of MetaInstance i is given as $\mathbf{f}_i \triangleq \{F_i^f\}_{f=1}^K$. Then, the similarity index between MetaSlice m and MetaInstance j is calculated by any similarity function such as cosine and Jaccard [12]. Here, we use the Jaccard similarity function, denoted by d_{Jaccard} , which is defined as follows:

$$d_{\text{Jaccard}}(\mathbf{f}_m, \mathbf{f}_j) = \frac{\mathbf{f}_m \cdot \mathbf{f}_j}{\|\mathbf{f}_m\|^2 + \|\mathbf{f}_j\|^2 - \mathbf{f}_m \cdot \mathbf{f}_j}, \quad (1)$$

where $\|\cdot\|$ is the 2-norm of a vector and the nominator is the dot product of two vectors.

After obtaining all the similarity indexes, the system will add MetaSlice m to the MetaInstance that has the highest similarity index. If MetaSlice m has dedicated functions, the system will allocate resources to create these functions. In a case that MetaSlice m does not have any common function with ongoing MetaInstances, the system will create a new MetaInstance for it. Once a MetaSlice completes/departs, its

occupied resources are released. In practice, the request arrival and MetaSlice departure processes are highly dynamic and uncertain. To address these challenges, we develop a semi-Markov decision process-based framework in the next section.

III. METAVERSE ADMISSION CONTROL FORMULATION

This paper develops a semi-Markov Decision Process (sMDP) to enable the system to adaptively decide to accept/reject a request based on (i) its currently available resources, (ii) the requested MetaSlice's required resources, and (iii) its class, under the high dynamic and uncertainty of request arrival and MetaSlice departure processes. In addition, the sMDP makes decisions in a real-time manner, and thus it can capture the real-time admission control. The rest of this section will describe our proposed sMDP-based framework.

A. State Space and Action Space

Since the resources of the provider are limited, to maximize the provider revenue, it is necessary to consider the system's available resources and the required resources of requests. In addition, the class identification (i.e., class ID) of a requested MetaSlice is also important because each class can bring different income for the provider. Generally, the sMDP specifies decision epochs as time points where decisions are taken [13]. As such, in this work, we can define the decision epochs as inter-arrival time between two consecutive MetaSlice requests. Therefore, the system state s at a decision epoch can be defined as $s \triangleq (\mathbf{n}_u, \mathbf{n}_m, g)$, where g is the class ID of the requested MetaSlice, and $\mathbf{n}_u = \{n_u^p\}_{p=1}^P$ and $\mathbf{n}_m = \{n_m^p\}_{p=1}^P$ are two vectors representing the system available resources and the required resources of a requested MetaSlice, respectively. Each coordinate of these vectors, i.e., n_u^p and n_m^p , specifies the number of resources types p . Thus, the system state space is given as follows:

$$\mathcal{S} \triangleq \left\{ (\mathbf{n}_u, \mathbf{n}_m, g) : g \in \{1, \dots, G\}; \right. \\ \left. n_u^d \text{ and } n_m^p \in \{0, \dots, N^p\} \forall p \in \{1, \dots, P\} \right\}, \quad (2)$$

where N^p is the total number of resources type p of the provider.

Note that in our proposed sMDP, a state transition from state s to state s' only happens when an event arises, e.g., the arrival of a MetaSlice request. Let $\mathbf{e} = \{e_g\}_{g=1}^G$ denote the system event, where $e_g \in \{-1, 0, 1\}$ indicates that (i) if $e_g = 1$, a request class- g arrives, (ii) if $e_g = -1$ a MetaSlice class- g departs, and (iii) $e_g = 0$, otherwise. Consequently, we can derive the set of all possible events as follows:

$$\mathcal{E} \triangleq \left\{ \mathbf{e} : e_g \in \{-1, 0, 1\}; \sum_{g=1}^G |e_g| \leq 1 \right\}. \quad (3)$$

For the case when there is no MetaSlice of any class departing or arriving, we can define a trivial event, i.e., $\mathbf{e}^* \triangleq (0, \dots, 0)$.

Suppose that at state s a request arrives, then the system must take an action a_s , which is to accept (i.e., $a_s = 1$) or reject (i.e., $a_s = 0$) this request to maximize the provider's long-term revenue. Therefore, we can define the action space at state s as $\mathcal{A}_s \triangleq \{0, 1\}$.

B. Transition Probability and Immediate Reward Function

To obtain the transition probabilities that the system transits from one state to another, we adopt the uniformization technique [13]. In practice, end-users join and leave the system at any time, which is unknown in advance. Thus, this paper considers that the arrival process of requests class- g and the departure process of MetaSlices class- g follow the Poisson distribution with mean λ_g and the exponential distribution with mean $1/\mu_g$, respectively [13]. Let x_g denote the number of ongoing MetaSlices class- g , we then can represent the number of all running MetaSlices by a vector $\mathbf{x} \triangleq \{x_g\}_{g=1}^G$. Given the above, parameters of uniformization are defined as:

$$z = \max_{\mathbf{x} \in \mathcal{X}} \sum_{g=1}^G (\lambda_g + x_g \mu_g), \quad (4)$$

$$z_{\mathbf{x}} = \sum_{g=1}^G (\lambda_g + x_g \mu_g), \quad (5)$$

where \mathcal{X} denotes the set of all possible values for \mathbf{x} . Now, the occurrence probability of the next event \mathbf{e} is given as follows. The probability of an arrival of request class- g appears in \mathbf{e} is λ_g/z . The probability of a departure of MetaSlice class- g appears in \mathbf{e} is $x_g \mu_g/z$, and the probability that the next event is a trivial event \mathbf{e}^* is $1 - z_{\mathbf{x}}/z$. Then, we can obtain the transition probabilities for the sMDP.

To maximize the long-term revenue for the provider, the immediate reward function needs to consider the revenue from leasing resources. In addition, the proposed application decomposition and MetaInstance techniques can offer a higher resource utilization by sharing resources among the MetaSlices. As such, accepting a MetaSlice with lower occupied resources will benefit the provider in the long run, and thus the resource occupation of a requested MetaSlice is another important factor. Thus, we can define the reward function as:

$$r(\mathbf{s}, a_s) = \begin{cases} r_g - \sum_{p=1}^P w_p n_o^p, & \text{if } e_g=1 \text{ and } a_s=1, \\ 0, & \text{otherwise,} \end{cases} \quad (6)$$

where r_g is an income obtained by releasing resources for a MetaSlice class- g , and n_o^p is an amount of resources type p occupied by this MetaSlice. Here, the weights, i.e., $w_{p=1}^P$, define the tradeoff between these factors, which can be set based on the provider's business strategies. In (6), it can be observed that even MetaSlices have the same income (i.e., r_g), accepting the one with lower occupied resources gets a greater reward, thereby maximizing the provider's long-term revenue.

The objective of this work is to find an optimal admission policy π^* for the system to maximize the long-term average reward function, i.e., $\mathcal{R}(\pi)$, as follows:

$$\max_{\pi} \mathcal{R}(\pi) = \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T \mathbb{E}[r_t(s_t, \pi(s_t))], \quad (7)$$

where $\pi(s_t)$ is the action that is selected at state s_t at time t according to the admission policy π and r_t is an immediate reward derived from (6). In the next section, we discuss our proposed intelligent algorithm that can automatically and effectively find the optimal policy π^* for the system.

IV. INTELLIGENT METASLICE ADMISSION CONTROL

In practice, the request arrival and MetaSlice departure processes are unknown in advance because end-users can come and leave at any time. Therefore, it is ineffective to apply conventional optimization techniques to find an optimal admission policy for the system. In this context, Deep Q-learning techniques can help the system gradually learn an optimal admission policy without requiring complete information about the request arrival and MetaSlice departure processes. However, conventional deep Q-learning techniques face the overestimation problem when estimating the optimal Q-values [14], i.e., $Q^*(s, a)$, thereby possibly leading to an unstable learning process or even resulting in a sub-optimal policy. To address this issue, we develop a highly effective Deep Reinforcement Learning (DRL)-based algorithm for the system, namely iMSAC, that adopts recent advanced techniques, i.e., the buffer replay mechanism, the dueling neural network architecture [15], and the double Q-learning [14]. The iMSAC is described in details in Algorithm 1.

Algorithm 1 The iMSAC

Initialize ϵ , buffer \mathbf{M} , and Q-network \mathcal{Q} with random parameters θ .

Create target Q-network $\bar{\mathcal{Q}}$ by cloning the Q-network.

for $step = 1$ to T **do**

 Get action a_t following the ϵ -greedy policy as follows:

$$a_t = \begin{cases} \underset{a \in \mathcal{A}}{\operatorname{argmax}} Q(s_t, a; \theta_t), & \text{with probability } 1 - \epsilon, \\ \text{random action } a \in \mathcal{A}, & \text{otherwise.} \end{cases} \quad (8)$$

 Execute a_t , then observe reward r_t and next state \mathbf{s}_{t+1} .

 Store experience $(\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1})$ in \mathbf{M} .

 Sample \mathbf{M} randomly to get a mini-batch of experiences.

 Using (9) and (10) to get the Q-value and the target Q-value, respectively.

 Update θ based on SGD algorithm.

 Decrease the value of ϵ .

 Set $\bar{\theta} = \theta$ at every C steps.

end for

In Algorithm 1, at time t , the MetaSlicing system is at state s_t and performs an action a_t derived from the ϵ -policy, as in (8). Then, the system moves to a new state s_{t+1} and receives an immediate reward r_t . Since experiences, i.e., tuples $\langle s_t, a_t, s_{t+1}, r_t \rangle$, obtained in sMDP are highly correlated, using them directly to train the Deep Neural Network (DNN) may lead to a slow convergence speed [17]. To mitigate this problem, we adopt the replay buffer mechanism, in which experiences are stored in a buffer \mathbf{M} . Then, to train the DNN, a mini-batch of experiences is sampled randomly from \mathbf{M} . In iMSAC, the DNN is leveraged for estimating the Q-values so that the input and output layers are set according to the state dimension (i.e., available resources, required resources and class ID of the incoming MetaSlice request) and the action

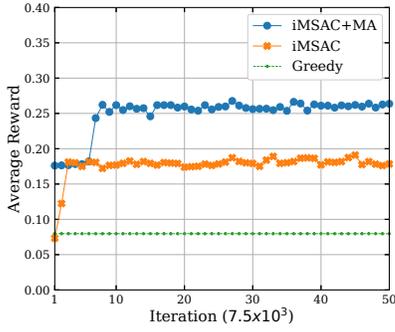


Fig. 2: Convergence rate of proposed algorithm iMSAC.

dimension (i.e., accept and reject), respectively. By inputting a state s in the DNN, estimated Q-values for all actions at state s are obtained, each corresponding to a neuron in the output layer. To stabilize the learning process, we adopt the dueling architecture that divides the iMSAC's DNN into two streams, one to estimate the state-value function $\mathcal{V}(s)$ and another for estimating the advantage function $\mathcal{W}(s, a)$. It is worth noting that while $\mathcal{V}(s)$ determines how good to be at a state s , $\mathcal{W}(s, a)$ specifies the importance of action a in comparison with others at state s . Let β and ζ denote parameters of the $\mathcal{V}(s)$ and $\mathcal{W}(s, a)$ streams, respectively. Thus, the Q-value of performing action a at state s is estimated by the iMSAC's DNN as follows [15]:

$$\mathcal{Q}(s, a; \beta, \zeta) = \mathcal{V}(s; \beta) + \left(\mathcal{W}(s, a; \zeta) - \frac{1}{|\mathcal{A}_s|} \sum_{a' \in \mathcal{A}_s} \mathcal{W}(s, a'; \zeta) \right). \quad (9)$$

To mitigate the overestimation problem, we adopt the double Q-learning method that uses two identical DNNs, which are Q-network \mathcal{Q} for action selections, and target Q-network $\bar{\mathcal{Q}}$ for evaluating action. Then, the target Q-value at time t is given as follows:

$$Z_t = r_t(s_t, a_t) + \alpha \bar{\mathcal{Q}}(s_{t+1}, \underset{a}{\operatorname{argmax}} \mathcal{Q}(s_{t+1}, a; \theta_t); \bar{\theta}_t), \quad (10)$$

where θ and $\bar{\theta}$ denote the \mathcal{Q} 's and $\bar{\mathcal{Q}}$'s parameters, respectively. The importance of future rewards is reflected by the discount factor α . Since the objective of training \mathcal{Q} is minimizing the distance between the estimated Q-value and the target Q-value, we can define a loss function at time t as follows:

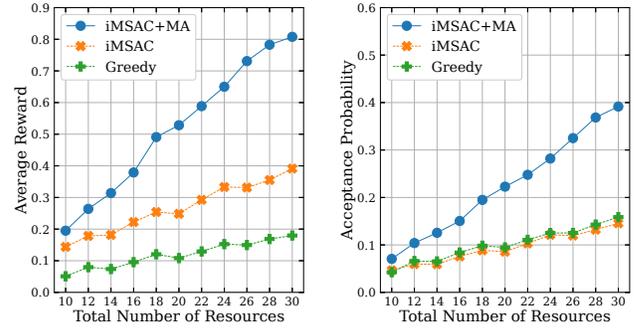
$$\mathcal{L}_t(\theta_t) = \mathbb{E}_{(s, a, r, s')} \left[(H_t - \mathcal{Q}(s, a; \theta_t))^2 \right], \quad (11)$$

where $\mathbb{E}[\cdot]$ is the expectation regard with data points, i.e., (s, a, r, s') , in \mathcal{M} . In this work, we employ Stochastic Gradient Descent (SGD) to minimize the loss function $\mathcal{L}_t(\theta_t)$ because this method has lower computing complexity compared with that of conventional Gradient Descent while the convergence is still guaranteed [16]. Note that to stabilize the learning process, as in [17], the target Q-network $\bar{\mathcal{Q}}$ is not updated at every time step. Instead, at every C steps, $\bar{\theta}$ is cloned from θ .

V. PERFORMANCE EVALUATION

A. Simulation Setting

Unless otherwise stated, the simulation parameters are set as follows. The system supports nine function types. There are three different functions for each MetaSlice. We consider that a



(a) Average rewards

(b) Acceptance probability

Fig. 3: Vary the total number of system resources.

function can be shared among a maximum of five MetaSlices. MetaSlices are classified into three classes, i.e., class-1, class-2, and class-3. The immediate reward r_g is set to 1, 2, and 4, and λ_g is set to 60, 40, and 25 request/hours for class-1, class-2, and class-3, respectively. In the MetaSlice departure processes of all classes, μ_g is set at two MetaSlices/hours. We consider three resource types, i.e., radio, storage, and computing, and each MetaSlice's function requires similar resources as functions in the Network Slice [18], e.g., 40 MHz, 40 GB, and 40 GFLOPS/s. Note that the proposed algorithm iMSAC does not require complete information about surrounding environment (e.g., arrival and departure rates and total system resources) in advance. Instead, it can adapt its policy accordingly to practical demands and requirements.

The parameters of the iMSAC algorithm are set as follows. The value of ϵ is first set to one, and then it slowly decreases to 0.001. We set the discount factor α to 0.9. The hyperparameters of Q-network are set similar to those in [15], e.g., learning rate is 10^{-3} , and $C = 10^4$. Recall that the proposed solution includes the self-learning algorithm iMSAC and the MetaSlice analysis. Without the MetaSlice analysis, MetaInstances cannot be created, and thus no function is shared among ongoing MetaSlices. Hence, to evaluate our proposed solution, namely iMSAC+MA, we use two baseline methods, i.e., iMSAC and Greedy policy (that always accepts requests if the system has sufficient resources).

B. Simulation Results

Fig. 2 shows the convergence rate of iMSAC in two schemes, i.e., with and without the MetaSlice Analyzer. The average reward obtained by the Greedy policy is also shown as a baseline. In this experiment, the radio, storage, and computing resources are set to 480 MHz, 480 GB, and 480 GFLOPS/s, respectively. It can be observed that although the iMSAC+MA converges slower than the iMSAC does, the average reward of the policy obtained by iMSAC+MA (i.e., 0.264) is 48% greater than that of the iMSAC (i.e., 0.17). In addition, the average rewards of iMSAC+MA and iMSAC are 230% and 123% greater than that of the Greedy, respectively.

Next, we investigate the robustness of our proposed solution by varying the radio, storage, and computing resources from 400 MHz, 400 GB, and 400 GFLOPS/s to 1200 MHz, 1200

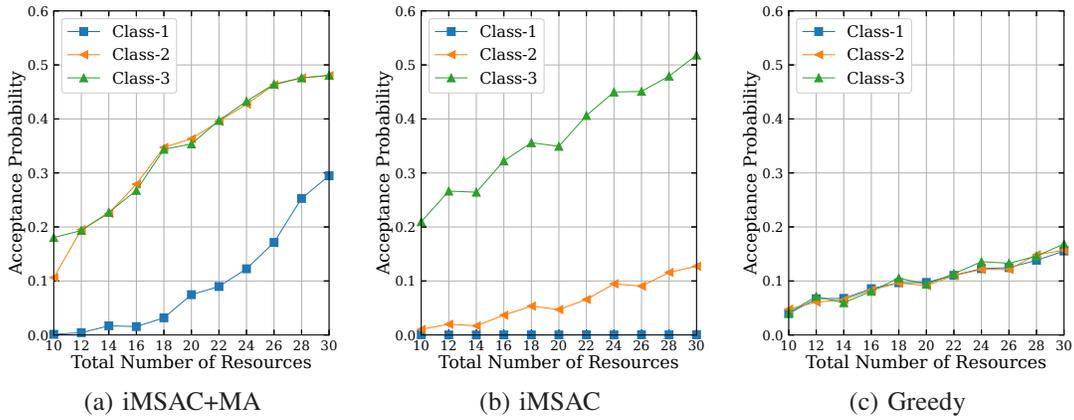


Fig. 4: The acceptance probability per class when varying total number of resources.

GB, and 1200 GFLOPS/s, respectively. In other words, the maximum number of functions simultaneously running on the system is varied from 10 to 30. The policies learned by the iMSAC and iMSAC+MA are acquired after 3.75×10^5 iterations. In this experience, we use average reward and acceptance probability metrics because they clearly demonstrate the system performance in terms of provider's long-term average revenue and the users' QoS, i.e., service availability. Fig. 3 demonstrates that the average rewards and acceptance probabilities obtained by all solutions increase when the system resource quantity increases. It is stemmed from the fact that with more resources, the system can host a greater number of MetaSlices, and thus it can accept more requests to get higher revenue than those of the system with a lower amount of resources. As the amount of system resources increases, iMSAC+MA always obtains the best results in terms of average reward and acceptance probability, which are up to 120% and 178.9% greater than those of the iMSAC, respectively, as shown in Fig. 3. Interestingly, even though the acceptance probability of the iMSAC is slightly lower than that of the Greedy, the iMSAC consistently achieves higher average rewards (e.g., up to 182%) than the Greedy does, as shown in Fig. 3(a). The reason can be observed in Fig. 4 where we look deeper at the acceptance probability per class. Specifically, in the iMSAC+MA and iMSAC, the acceptance probability of class-3, which has the highest immediate reward (i.e., 4), is much higher than that of class-1, which has the lowest immediate reward, i.e., 1. In contrast, in the Greedy, the acceptance probabilities of all classes are similar. The above results clearly demonstrate the effectiveness and robustness of our proposed solution.

VI. CONCLUSION

This paper has proposed the novel resource management framework to dynamically allocate appropriate resources from different tiers for effectively addressing the massive resource demands of Metaverse's applications, thereby maximizing the provider's long-term revenue. To capture real time, dynamic and uncertainty of request arrival and MetaSlice departure processes, we have developed the sMDP-based framework and

the intelligent algorithm that can gradually learn the optimal admission policy without requiring complete information about these processes. The simulation results clearly show the effectiveness and robustness of our proposed solution.

REFERENCES

- [1] M. Xu *et al.*, "A full dive into realizing the edge-enabled Metaverse: Visions, enabling technologies, and challenges," *arXiv preprint arXiv:2203.05471*, 2022.
- [2] <https://www.credit-suisse.com/media/assets/corporate/docs/aboutus/media/media-release/2022/03/metaverse-14032022.pdf>, (accessed: May 01, 2022).
- [3] X. Wang *et al.*, "Characterizing the gaming traffic of World of Warcraft: From game scenarios to network access technologies," *IEEE Network*, vol. 26, no. 1, pp. 27-34, Jan. 2012.
- [4] J. Zhao, R. S. Allison, M. Vinnikov, and S. Jennings, "Estimating the motion-to-photon latency in head mounted displays," in *Proceedings of the 2017 IEEE Virtual Reality (VR)*, 2017, pp. 313-314.
- [5] Y. Jiang *et al.*, "Reliable coded distributed computing for Metaverse services: Coalition formation and incentive mechanism design," *arXiv preprint arXiv:2111.10548*, 2021.
- [6] M. Xu *et al.*, "Wireless edge-empowered Metaverse: A learning based incentive mechanism for virtual reality," *arXiv preprint arXiv:2111.03776*, 2021.
- [7] W. C. Ng *et al.*, "Unified resource allocation framework for the edge intelligence-enabled Metaverse," *arXiv preprint arXiv:2110.14325*, 2021.
- [8] Y. Han *et al.*, "A dynamic resource allocation framework for synchronizing Metaverse with IoT service and data," *arXiv preprint arXiv:2111.00431*, 2021.
- [9] <https://analyticsindiamag.com/google-maps-api-is-the-new-imperative-to-mobile-gaming/>, (accessed: May 01, 2022).
- [10] B. Alturki *et al.*, "Exploring the effectiveness of service decomposition in fog computing architecture for the internet of things," *IEEE Transactions on Sustainable Computing*, vol. 7, no. 2, pp. 299-312, Mar. 2019.
- [11] <https://www.ngmn.org/wp-content/uploads/Publications/2016/161010NGMNNetworkSlicingframeworkv1.0.8.pdf>, (accessed: May 01, 2022).
- [12] P. Jaccard, "The distribution of the flora in the alpine zone," *The New Phytologist*, vol. 11, no. 2, pp. 37-50, Feb. 1912.
- [13] H. C. Tijms, *A First Course in Stochastic Models*. Wiley, 2003.
- [14] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016, pp. 2094-2100.
- [15] Z. Wang *et al.*, "Dueling network architectures for deep reinforcement learning," in *Proceedings of the 33rd International Conference on Machine Learning*, 2016, pp. 1995-2003.
- [16] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, pp. 400-407, Sep. 1951.
- [17] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529-533, Feb. 2015.

- [18] G. Dandachi *et al.*, “An artificial intelligence framework for slice deployment and orchestration in 5G networks,” *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 2, pp. 858–871, Nov. 2020.