Dynamic Multi-tier Resource Allocation Framework for Metaverse

Nam H. Chu, Hieu Q. Nguyen, Diep N. Nguyen, Dinh Thai Hoang, Khoa T. Phan, Eryk Dutkiewicz, Dusit Niyato, and Tao Shu

***Abstract*—Since Metaverse requires enormous resources that have never been seen before, resource management is one of the biggest challenges hindering Metaverse deployment. Thus, this paper introduces a novel framework that can effectively and smartly manage various types of resources in different network layers to guarantee strict requirements of Metaverse. In particular, this framework is built based on two innovative techniques: (i) MetaSlice decomposition providing a flexible and effective solution in deploying, managing, and updating Metaverse applications, and (ii) MetaInstance that can maximize resource utilization by exploiting similarities among Metaverse applications. Moreover, to address the dynamic, uncertain, and real-time resource demand in Metaverse, we develop an intelligent algorithm that can quickly find the optimal resource allocation for the system. The key idea of this algorithm is to automatically learn the optimal policy through interactions the environment without requiring complete information. The simulation results show that the proposed framework can not only improve the long-term revenue for the Metaverse provider up to 1.8 times but also enhance user experience near 1.5 times compared with other baseline schemes.**

***Index Terms*—Metaverse, resource management, semi-Markov decision process, machine learning.**

I. Introduction

Thanks to the recent blossom of emerging technologies, e.g., artificial intelligence, hardware virtualization, and virtual reality, Metaverse, a concept introduced in 1992 [1], is not fiction anymore. Specifically, Metaverse is built on two pillars. First, it is a blended environment of physical and virtual worlds. Currently, virtual reality techniques, e.g., Extended Reality (XR), can achieve the integration of physical and virtual environments by allowing users to interact seamlessly with both physical and virtual objects simultaneously. In addition, Metaverse allows users not only to freely create their virtual objects but also bring their physical objects (e.g., painting) to Metaverse by digitalizing technologies (e.g., digital twins [2]). Moreover, users can share and trade their assets (e.g., virtual paintings and outfits) with others in Metaverse. Second, unlike conventional virtual worlds, where each is created for a particular application (e.g., entertainment or education), Metaverse is the seamless integration of various applications. As such, a user only needs to maintain one account to enjoy all applications (i.e., virtual worlds), thereby relieving the headache of managing multiple accounts for different applications. Moreover, analogous to our real lives, Metaverse allows users to bring/share their virtual assets across virtual worlds without losing their values. Thus, Metaverse is a revolution changing not only how we immerse ourselves in entertainment but also many aspects of our daily lives, e.g., industry, education, and healthcare.

However, Metaverse is still only at its infancy, and thus a lot of effort is needed towards its success. Among obstacles hindering the development of Metaverse, resource management is one of the most important problems. It is stemmed from the fact that Metaverse applications require mountainous resources of various types (e.g., computing, storage, and networking) with strict requirements. First, reality technologies (e.g., XR) blending digital objects into the physical environment require not only intensive computing resources for rendering 3D objects but also strict delay requirements in both computing and networking to maintain Quality-of-Experience (QoE) for users. Second, in Metaverse, interactions between physical and virtual environments are bi-direction. For example, gestures and body movements can be used to control virtual objects, while a virtual remote can be used to control household devices. Thus, Metaverse applications require high-speed connections with low latency for both directions. Third, due to the integration of multiple virtual worlds in Metaverse, a huge number of users is expected to join Metaverse simultaneously, leading to tremendous resource demand. As such, data forwarded over networks is expected to increase about 20 times thanks to Metaverse operation [3]. Moreover, users can enter and leave anytime, leading to high uncertainty and dynamic resource demand in Metaverse. Therefore, the Metaverse's deployment calls for an innovative solution to address the above challenges effectively.

To address the aforementioned challenges, this paper introduces a novel resource management framework for deploying Metaverse applications. In particular, the proposed framework is established based on two innovative techniques to flexibly and effectively manage resources by exploiting similarities among applications. In addition, a deep reinforcement learning algorithm is developed to address the dynamic, uncertainty, and real-time resource demand in Metaverse. Simulation results demonstrate that the proposed solution can improve

Nam H. Chu, Hieu Q. Nguyen, Diep N. Nguyen, Dinh Thai Hoang, and Eryk Dutkiewicz are with the School of Electrical and Data Engineering, University of Technology Sydney, Australia (e-mails: namhoai.chu@student.uts.edu.au, hieu.nguyen-1@student.uts.edu.au, diep.nguyen@uts.edu.au, hoang.dinh@uts.edu.au, and eryk.dutkiewicz@uts.edu.au).

Khoa T. Phan is with School of Engineering and Mathematical Sciences, La Trobe University, Melbourne, Australia (e-mail: K.Phan@latrobe.edu.au).

Dusit Niyato is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798 (e-mail: dniyato@ntu.edu.sg).

Tao Shu is with the Department of Computer Science and Software Engineering, Auburn University, Auburn, AL 36849. (e-mail: tshu@auburn.edu).

Metaverse provider's long-term revenue up to 1.8 times and at the same time enhance service availability for users near 1.5 times. The main contributions of this paper are summarized as follows:

- We discuss current resource allocation approaches and then introduce a novel framework that can dynamically, effectively, and intelligently manage high volume of resources in various types to maximize the performance of Metaverse applications.
- We propose two innovative methods, i.e., Metaverse application decomposition and MetaInstance, that can exploit similarities among applications to maximize resource utilization.
- We develop an intelligent algorithm together with the underlying model formulated by the semi-Markov decision process to help the system quickly learn the optimal resource allocation policy under the uncertainty, dynamic, and real-time environment.

II. Resource Management for Metaverse

Figure 1 illustrates the network architecture of a Metaverse system when Metaverse applications are deployed. In practice, creating a Metaverse application to serve a large number of users requires a huge amount of different types of resources. Specifically, it usually requires a lot of computing resources to create a virtual world to be able to serve many users to join simultaneously. For example, according to [4], Fortnite, a game hosting over 15.3 millions of concurrent users on live virtual concerts, requires tens of thousands of instances equipped by AWS Graviton processors whose computing capacity is equivalent to Intel Xeon Platinum 8259CL. More importantly, Metaverse applications are much more complicated than those of existing virtual worlds due to the fully blending between digital and physical environments. For instance, Metaverse allow users to create virtual objects by digitalizing real objects (e.g., paintings). Then, they can interact with these objects (e.g., touching, modifying, and coloring) as well as share them with others in Metaverse. Digitalizing real objects from the real world to the Metaverse environment requires not only intensive computing resources but also a huge amount of data to process. Furthermore, users generally join Metaverse on their portable devices with limited resources, e.g., computing and energy, and thus streaming services is promising approach to alleviate mountainous resource demand of Metaverse application. However, unlike conventional streaming services, Metaverse applications require interactive VR 360^0 streaming services in which users can immerse in Metaverse spaces with 3D view and at the same time interact with all surrounding objects and other users in a such virtual environment. This will require not only enormous radio resources (including both uplink and downlink) but also various computing types at the edges, e.g., to preprocess video streaming. Therefore, this demands an innovative resource management solution that can effectively address tremendous demands in various resource types from Metaverse applications.

In the literature, a few attempts have been conducted to address resource allocation problems in Metaverse. Most studies

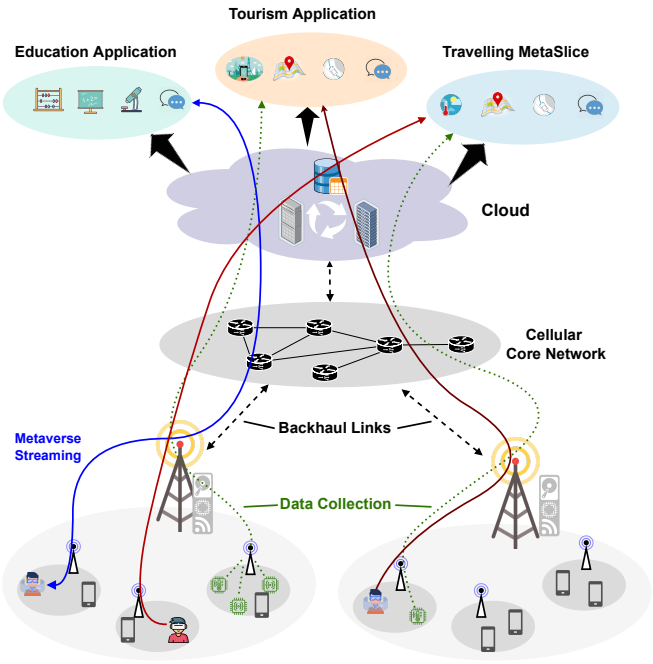


Fig. 1: The network architecture of Metaverse system.

consider computing resource allocation at the edge [5]–[7]. In [5], the authors consider a task offloading problem in a Metaverse vehicular application. Particularly, since intensive Metaverse computing tasks cannot be efficiently processed locally at a vehicle, the authors propose to use edge computing to execute these tasks. First, the Metaverse provide will select qualified vehicles to execute computation tasks based on their reputation values. To alleviate the straggler effects in the offloading task problem, the authors then adopt a Coded Distributed Computing (CDC) approach for computing these tasks. After that, a Stackelberg game model is proposed to investigate the reliable and sustainable CDC scheme in this vehicular Metaverse application.

The edge-powered Metaverse is further investigated by studies in [6] and [7], which considers more types of resources, i.e., computation and communication. The authors in [6] study the scenario in which VR service providers offer edge-computing resources for rendering the VR to Metaverse users. They propose a framework that maps and prices virtual reality services between users and the service provider dynamically, so that the seamless experiences of users are guaranteed. The objective of this work is to maximize users' experiences and perception. For that, an incentive mechanism based on deep reinforcement learning and double Dutch auction is designed to achieve effective performance in terms of social welfare and information exchange cost. In [7], the authors propose a unified resource allocation framework to tackle stochastic user demand in the Metaverse education application. In this work, a stochastic integer programming method is utilized to minimize the cost of a service provider, given the stochastic demand of the cyber and physical resources from users. Simulation results show that the proposed framework achieves better performance than other baselines in terms of cost for the provider of virtual

education. Unlike the above works, the authors in [8] consider components in Metaverse (e.g., infrastructures and software) as services, similar to the concept of everything-as-a-service (XaaS) in cloud systems. For managing and orchestrating Metaverse services, the authors propose a framework that can customise various service models with different types of resources. Simulation results show that the proposed framework can handle resource mapping problems in a travelling application, given the network's dynamic rendering capacity and data rates.

It can be observed that all of the above works focus on a single-tier resource architecture (i.e., edge computing) for Metaverse applications, which may lead to a point-of-congestion problem. The reason is that in edge computing, users typically are allocated resources that near their locations. Thus, when a large number of users experience Metaverse application simultaneously in a small area, their nearby resources are inadequate to meet users' QoE requirements. In this context, multi-tier resource allocation architecture is a promising solution because it can alleviate a point-of-congestion by distributing resources along the path from users to the cloud. In addition, it also offers a flexible solution to deploy Metaverse applications. In particular, Metaverse applications can be allocated resources at different tiers according to their requirements. Moreover, none of the above works and others in the literature can exploit the similarities among Metaverse applications to improve resource utilization. It is observed that Metaverse applications may share some same functions. For example, a digital map likely belongs to the travel and vehicular Metaverse applications. If this function is shared among applications, system resource usage efficiency is greater than that of the scenario in which a digital map is separately created for each application. In the next section, we will introduce a novel solution for effectively managing the resources in Metaverse based on a multi-tier resource allocation architecture.

III. Dynamic Multi-tier Resource Management Framework for Metaverse

A. The Multi-tier Resource Allocation Architecture for Metaverse Applications

Due to the stringent delay requirement and extremely high resource demands in various types, we propose a novel multi-tier resource allocation architecture for implementing Metaverse applications. Specifically, resources (e.g., computing, storage, and networking) can be allocated along the way from end-user to the cloud to form a multi-tier resource allocation architecture. Suppose that an user enters a Metaverse application via a 5G network so that the first resource tier can be mapped to 5G small cells (e.g., femtocells, microcells, and microcell), the second tier can be mapped to 5G macrocells, and so on, as illustrated in Figs. 1 and 2. If this application requires a low latency, it can be allocated resources near end-users, e.g., tier-1. In contrast, if application needs more intensive resources (e.g., computing), resources at a higher tier should be used. As a result, the proposed multi-tier resource allocation architecture offers a highly-effective and flexible solution for Metaverse applications.

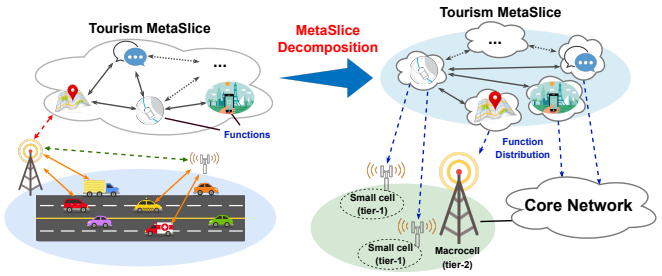


Fig. 2: An example of the MetaSlice Decomposition, where a tourism MetaSlice is decomposed into multiple functions (e.g., recommendation and digital map) that can be created and operated independently at different tiers of the multi-tier resource architecture.

Moreover, our proposed architecture can offer a number of benefits to Metaverse providers and users. First, each Metaverse application can be allocated resources at different tiers, so the point-of-failure and network congestion problems of the conventional cloud can be alleviated. Second, the multi-tier architecture brings computing resources nearer to users, and thus it reduces data transmission delay, which is crucial in Metaverse to maintain QoE for users. Third, this architecture can distribute different resource types (e.g., computing, radio, and storage) over networks, making it more resilient and flexible than that of the centralized resource allocation architecture.

B. MetaSlicing Framework

This subsection presents our novel resource management framework, i.e., MetaSlicing, that can effectively manage Metaverse applications built on multi-tier resource architecture. The MetaSlicing consists of two innovative techniques, i.e., MetaSlice decomposition and MetaInstance, that can provide flexible solutions in managing and allocating different types of resources for different Metaverse applications based on their actual demands.

1) MetaSlice Decomposition: As discussed in Section III-A, a multi-tier resource architecture is a very promising solution for Metaverse application deployment. However, it still poses several challenges. First, most Metaverse applications demand low delay on processing and connection, so they are likely to be created at the edge, i.e., tier-1. Thus, this leads to overload problems at low tiers, whose resources are typically lower than those of higher tiers, resulting in a high delay in computing and transmission or even service disruption. Note that delay is a crucial factor determining QoE of users in Metaverse. Second, if a user moves to an area far from the previous one, the QoE of its ongoing Metaverse applications may not be guaranteed. A possible solution for this problem is migrating these applications to a site near the user's new location. Nevertheless, the migration leads to unavoidable high delay due to the re-initialization of Metaverse application.

To address these challenges, we propose that a Metaverse application, i.e., MetaSlice, can be decomposed into different

functions that can be initialized and operated independently. For example, a tourism MetaSlice likely has recommendation and navigation functions. The recommendation function suggests nearby sightseeing according to a user's location and preferences, and then the navigation function takes action to give the best route to the chosen place as illustrated in Fig. 2. Indeed, these functions can operate independently. As such, a tourism MetaSlice can be developed by a modular design whose functions are initialized and operated separately. These independent functions can be viewed as a complete application so that they can be connected to others via some interfaces, similar to how current applications connect to existing online service platforms, e.g., Google Maps API.

The MetaSlice decomposition can bring numerous benefits to Metaverse's deployment and operation. First, it can make more convenient in managing MetaSlices. For example, if a function is failed, we can use another equivalent one without interrupting users and services. Second, since each function in a MetaSlice is an independent entity, it can be upgraded separately, resulting in faster evolutions of functions and MetaSlices. Third, this technique can help MetaSlice developer to focus more on their unique functions (e.g., a recommendation function in tourism MetaSlice) rather than putting effort on other functions (e.g., a digital map) that can be developed more efficiently by other parties with more experiences in this area. Fourth, the MetaSlice decomposition, together with a multi-tier resource allocation architect, provides a flexible solution for implementing Metaverse since each function of a MetaSlice can be dynamically created at a different tier according to its requirements. For example, suppose that a travelling MetaSlice has digital map and driving assistance functions. Since the driving assistance requires a low delay, it should be created and placed near users, e.g., tier-1. Whereas the digital map, which occasionally needs an update, should be placed on a higher tier, e.g., the cloud. Finally, the application migration delay problem, as discussed above, is also alleviated by MetaSlice decomposition. Specifically, instead of migrating entire a MetaSlice, only some functions with strict delay requirement will be migrated, leading to a significant decrease of migration delay. To support the MetaSlice decomposition, a MetaSlice can be initialized based on a record describing the configuration and workflow for creating and managing this MetaSlice throughout its life-cycle.

2) MetaInstance: To further exploit the benefits of MetaSlice decomposition, we introduce the MetaInstance to maximize resource utilization. Specifically, we observe that different MetaSlices may have some common functions. For example, MetaSlices for education, tourism, and industry may have functions for Metaverse users to communicate (e.g., instant message and video call). Moreover, one MetaSlice type may have multiple variants created and managed by different parties. Thus, there is a high probability that ongoing MetaSlices have some common functions. As a result, system resources can be further optimized if these MetaSlices can share the same function instead of creating one for each MetaSlice. In this way, the service provider can get more revenue by better optimizing resource utilization and at the same time enhance QoE for Metaverse users.

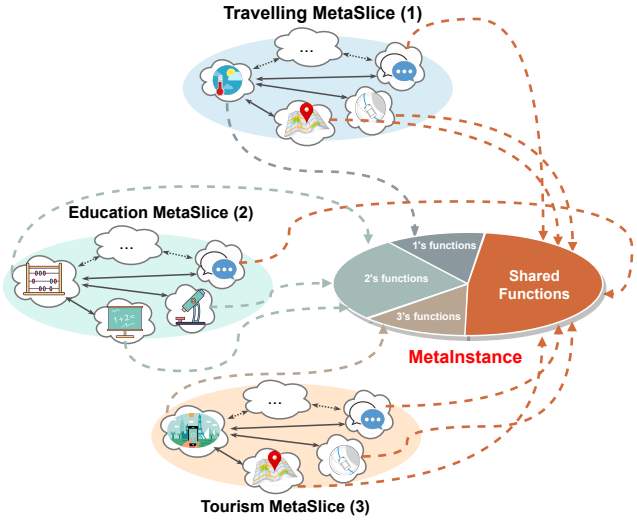


Fig. 3: An example of a MetaInstance that consists of multiple MetaSlices sharing some same functions (e.g., driving assistant, digital map, and messenger).

Given the above, we propose to group MetaSlices into multiple groups, named MetaInstances, based on their function similarities. Specifically, a MetaInstance contains multiple MetaSlices that share some same functions. As such, in a MetaInstance, there are two function types, including dedicated functions (which belong to particular MetaSlices) and shared functions serving multiple MetaSlices, as illustrated in Fig. 3. Note that a function in a MetaInstance can only be shared by MetaSlices in this group. To support the creation and management of MetaInstance, a MetaInstance maintains a configuration record describing its functions and interactions between them.

Note that the proposed MetaSlicing mechanism is different from network slicing mechanism in 5G networks [9]. Specifically, network slicing aims to provide various virtual networks (e.g., network slices) over a physical network to address different communication types for different businesses. For example, manufacturing customers may require ultra-reliable connections, while entertainment users usually need low-latency communications. Thus, network slicing mainly focuses on the communication aspect from users to MetaSlices, without capturing specific requirements of functions in MetaSlices. In contrast, our proposed MetaSlicing aims to offer a comprehensive solution for the deployment of Metaverse applications established on the underlying multi-tier resource allocation architecture. In particular, MetaSlicing first decomposes a MetaSlice into independent functions. Then, these functions are distributed across different tiers in the multi-tier resource architecture according to their requirements. Moreover, MetaSlicing groups MetaSlices into multiple MetaInstances, in which some function are shared among MetaSlices to improve the system resource utilization.

Based on the above analyses, the MetaSlicing, on the one hand, can help Metaverse providers to maximize their resource utilization while minimizing the initialization time and deployment cost for MetaSlices. On the other hand, it

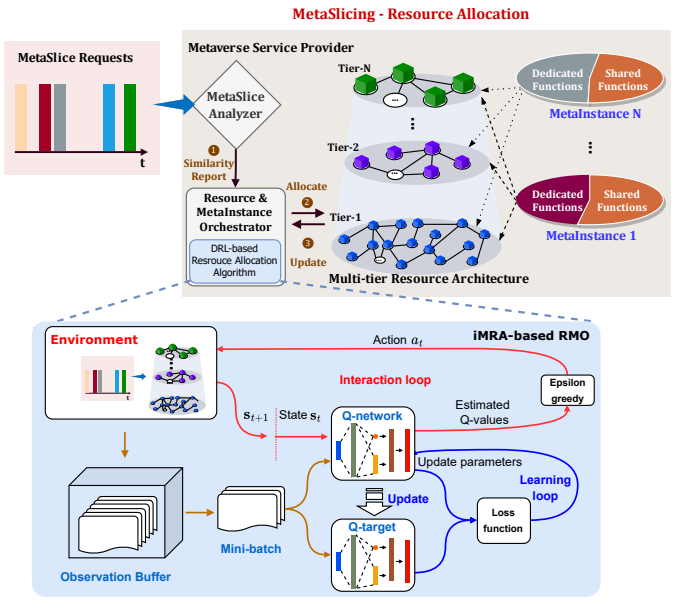


Fig. 4: MetaSlicing framework.

also benefits Metaverse users by improving QoE, e.g., high reliability and lower delay. To reach these goals, resource allocation plays a key role. For example, allocating resources for a MetaSlice sharing some functions with ongoing MetaSlices can save more resources than others with fewer or no shared functions. Therefore, in Section IV, we will introduce our propose algorithm that can effectively address this problem.

IV. Intelligent Resource Management in the MetaSlicing

Upon receiving a MetaSlice request, a MetaSlice Analyzer will be used to determine similarities between the incoming request and current MetaSlices running in the system, as shown in Fig. 4. Then, this analysis will be used as one of the key factors to assist the Resource and MetaInstance Orchestrator (RMO) in deciding whether to allocate resources for the request or not. If the requested MetaSlice is accepted, it will be assigned to a MetaInstance with the highest similarity. Then, resources are allocated to initiate the MetaSlice's dedicated functions, and the selected MetaInstance's function configuration record will be updated accordingly. If the accepted MetaSlice does not share any function with ongoing MetaInstances, a new MetaInstance is created for this one. When a MetaSlice departs, its occupied resources are released, and the function configuration record of its MetaInstance is updated accordingly. The following subsections discuss the MetaSlice analysis procedure and our proposed Deep Reinforcement Learning (DRL)-based resource allocation in the MetaSlicing.

A. MetaSlice Analysis

This subsection presents our proposed MetaSlice analysis that can determine the similarities between the requested MetaSlice and ongoing MetaInstances. Note that the similarity index is a key factor in the MetaSlicing. First, it is important information for the system to decide whether a MetaSlice should be allocated resources or not. Second, suppose that a MetaSlice is accepted to be allocated resources. Its similarity index guides the MetaSlicing to be assigned to a new MetaInstance or an existing one.

To determine similarities among MetaSlices, we propose using the configuration records obtained from the requested MetaSlice and MetaInstance. The reason is that a configuration record consists of configurations for all functions in a MetaSlice/MetaInstance. As such, we can determine a similarity index between the requested MetaSlice and an ongoing MetaInstance by comparing their configuration records. First, the similarity score for each function of the requested MetaSlice is calculated. Suppose that a function configuration can be represented by a binary vector so that we can leverage existing methods, e.g., Cosine and Jaccard [10], to output the similarity score of this function. Specifically, these methods determine the similarity between two configuration vectors, i.e., one from the requested MetaSlice and another from the compared MetaInstance. Note that if a function is not required by a MetaInstance/MetaSlice, the corresponding vector in the configuration record is all zeros, and thus the similarity score is zero for this function. Second, the similarity index of the requested MetaSlice is defined as an average of the similarity scores of its functions. In the next subsection, we will present our proposed resource allocation approach based on DRL that can leverage the information from MetaSlice analysis to maximize the system performance.

B. iMRA: A Deep Reinforcement Learning-based Resource Allocation Approach for Metaverse

As discussed in the previous sections, Metaverse application may experience a large number of users to join simultaneously. Moreover, users can come and leave at any time, leading to the high uncertainty and dynamic of resource demands. In addition, due to strict delay requirements of Metaverse applications, real-time resource allocation is a critical issue in Metaverse. Note that optimization theory-based methods cannot address effectively these challenges due to dynamic and unavailability of system parameters, e.g., users' demands. To that end, we propose to use the Semi-Markov Decision Process (SMDP) framework to address the resource allocation problem in Metaverse. Then, we develop a DRL-based algorithm that can automatically learn an optimal policy to maximize the system performance (e.g., services availability) and revenue for the MSP without requiring complete environment information in advance.

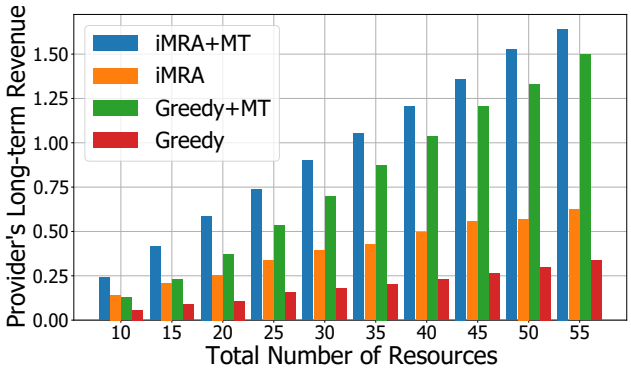
1) Real-time Resource Allocation: Our proposed SMDP framework is especially effective to allocate resources in Metaverse in a real-time manner due to the following reasons. First, it allows the system to automatically make the best action (e.g., whether to allocate resources or not) according to real-time observations of demands and current available resources [11]. In addition, unlike the conventional Markov Decision Process that takes an action at each equal time slot, SMDP takes action whenever an event occurs (e.g., resource demand arrival), making SMDP more suitable for

real-time tasks. In the SMDP, a state is the system observation (e.g., available resources, resource demand, and the requested MetaSlice's similarity index) that gives information to the RMO to make actions. After executing an action, the RMO receives an immediate reward that indicates system performance (e.g., service availability and revenue for the services provider) according to the selected action. In the next section, we will discuss our proposed resource allocation algorithm to help the RMO automatically learn an optimal policy through interactions with the environment.

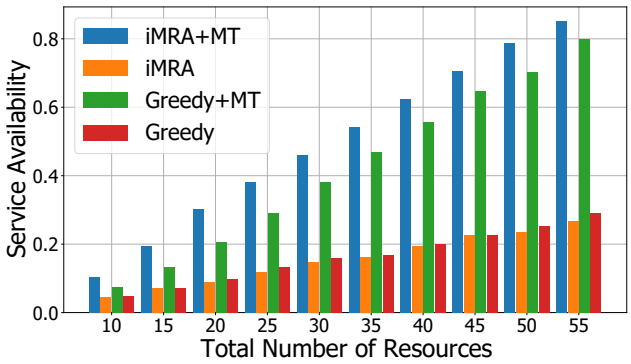
2) iMRA: A Deep Reinforcement Learning-based Resource Allocation Approach: In machine learning, DRL is a special branch leveraging Deep Neural Networks (DNN) to address consecutive decision-making problems under uncertainty and dynamic environments, often formulated as an MDP. In DRL, an agent often starts with an arbitrary policy (e.g., random policy), i.e., a mapping from the state space to the action space. Then, the agent gradually improves its policy based on interactions with the surrounding environment (e.g., states, actions, and immediate rewards). Eventually, it can learn an optimal policy that can maximize a long-term average reward function. This paper develops a DRL algorithm, namely iMRA, adopting recent advances in DRL. First, the iMRA adopts the history replay mechanism to break the correlation between consecutive observation, thus stabilizing the training process of DNN that is very sensitive to correlated data [12]. Second, the iMRA uses the dueling architecture [13], where the state value and advantage functions are estimated separately and simultaneously, making the learning process more stable. Third, the iMRA employs two DNNs, i.e., Q-network for estimating the value of performing an action and Q-target for action selection as illustrated in Fig. 4. The Q-network is updated at every time step, whereas the Q-target is only updated by copying parameters from the Q-network at certain time steps (e.g., 1000 time steps) to improve the stability of the learning process [14].

3) Simulation Results: In this section, simulations are conducted to evaluate the proposed solution, namely iMRA+MT. The DNN's hyper-parameters of iMRA are set similar to those in [12], [13], e.g., the learning rate is 10^{-3}. Note that our proposed iMRA+MT has two main components, i.e., the iMRA algorithm and MetaInstance. Therefore, we select three baseline methods, i.e., (i) iMRA, (ii) Greedy policy [15] that always allocates resources when the available resources are sufficient for the request, and (iii) Greedy policy with MetaInstance, namely Greedy+MT.

Here, we study the performance of our approach in various scenarios, each with different system resources. In particular, the total number of functions supported by the system is increased from 10 to 55. As shown in Fig. 5(a), the average rewards of all methods increase as the system resources increase. The reason is that a system with more resources can host more MetaSlices simultaneously, leading to a greater average reward. Fig. 5(a) also shows that the proposed iMRA+MT always gets the highest average rewards, up to 1.8 times compared to that of the Greedy+MT, the second-best approach. Similarly, Fig. 5(b) demonstrates that iMRA+MT brings the highest system availability, up to 1.47 times, compared to



(a) Provider's Long-term Revenue



(b) Service Availability

Fig. 5: Evaluation of the proposed MetaSlicing framework in various schemes with different total system resources.

that of the second-best approach, i.e., Greedy+MT. Moreover, it also clearly shows that the MetaInstance can increase the average reward and service availability of both the iMRA and Greedy by up to 2.68 and 4.6 times, respectively. Thus, the above results clearly demonstrate the superiority of our proposed MetaInstance and the iMRA algorithm. While the MetaInstance exploits function similarities among MetaSlice to increase resource usage, the iMRA helps the RMO to find an optimal resource allocation policy without requiring complete information about the environment in advance.

V. Conclusion

In this paper, we have proposed a novel resource management framework for Metaverse established on the multi-tier resource allocation architecture. This framework is built based on two innovative techniques, i.e., the MetaSlice decomposition and MetaInstance, that not only provide a flexible solution in Metaverse deployment but also improve resource usage by exploiting the similarities among MetaSlices. To deal with the high dynamic, real-time, and uncertainty of resource demand, we have formulated the problem as an SMDP and then developed the DRL algorithm to find the optimal resource allocation policy without requiring complete environment information in advance. The simulation results then clearly show the outperformance (in terms of resource allocation and revenue for the service provider) compared with other baseline approaches.

REFERENCES

[1] N. Stephenson, *Snow Crash*. New York: Bantam Books, 1992.

[2] Y. Wang *et al.*, "A survey on metaverse: Fundamentals, security, and privacy," *IEEE Communications Surveys & Tutorials*, Sep. 2022.

[3] "Metaverse: A guide to the next-gen internet," https://www.credit-suisse.com/media/assets/corporate/docs/about-us/media/media-release/2022/03/metaverse-14032022.pdf, (accessed: June 01, 2022).

[4] "Epic games on AWS," https://aws.amazon.com/solutions/case-studies/innovators/epic-games/, (accessed: May 01, 2022).

[5] Y. Jiang *et al.*, "Reliable distributed computing for metaverse: A hierarchical game-theoretic approach," *IEEE Transactions on Vehicular Technology*, pp. 1–16, Sep. 2022.

[6] M. Xu *et al.*, "Wireless edge-empowered metaverse: A learning-based incentive mechanism for virtual reality," in *IEEE International Conference on Communications*, 2022, pp. 5220–5225.

[7] W. C. Ng *et al.*, "Unified resource allocation framework for the edge intelligence-enabled metaverse," in *IEEE International Conference on Communications*, 2022, pp. 5214–5219.

[8] Y.-J. Liu, H. Du, D. Niyato, G. Feng, J. Kang, and Z. Xiong, "Slicing4meta: An intelligent integration framework with multi-dimensional network resources for metaverse-as-a-service in web 3.0," *arXiv preprint arXiv:2208.06081*, 2022.

[9] "Description of network slicing concept," NGMN Alliance. Accessed: Apr. 24, 2022. [Online]. Available: https://www.ngmn.org/wp-content/uploads/Publications/2016/161010_NGMN_Network_Slicing_framework_v1.0.8.pdf

[10] P. Jaccard, "The distribution of the flora in the alpine zone," *The New Phytologist*, vol. 11, no. 2, pp. 37–50, Feb. 1912.

[11] H. C. Tijms, *A First Course in Stochastic Models*. Wiley, 2003.

[12] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[13] 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.

[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] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press, 2018.

Nam H. Chu is currently a Ph.D. student at the University of Technology Sydney, Australia. His research interests include applying machine learning and optimization methods for wireless communication networks.

Nguyen Quang Hieu is currently a Ph.D. student at the University of Technology Sydney, Australia. His research interests include optimizing wireless network performance with advanced machine learning and deep (reinforcement) learning techniques.

Diep N. Nguyen is currently a faculty member with the Faculty of Engineering and Information Technology, University of Technology Sydney. His research interests include computer networking, wireless communications, and machine learning application, with emphasis on systems' performance and security/privacy.

Dinh Thai Hoang is currently a faculty member at the School of Electrical and Data Engineering, University of Technology Sydney, Australia. His research interests include emerging topics in wireless communications and networking such as machine learning, IoT, and 5G/6G networks.

Khoa T. Phan is currently a Senior Lecturer at the Department of Computer Science and Information Technology, La Trobe University, Victoria, Australia. His research interests are broadly design, control, optimization, and security of next-generation communications networks.

Eryk Dutkiewicz is currently the Head of School of Electrical and Data Engineering at the University of Technology Sydney, Australia. He also holds a professorial appointment at Hokkaido University in Japan. His current research interests cover 5G/6G and IoT networks.

Dusit Niyato is currently a professor in the School of Computer Science and Engineering and, by courtesy, School of Physical and Mathematical Sciences, at the Nanyang Technological University, Singapore. He is a Fellow of IEEE.

Tao Shu is currently an associate professor in the Department of Computer Science and Software Engineering at Auburn University. Before joining Auburn, he worked as an assistant professor in the Computer Science and Engineering Department of Oakland University in Rochester, Michigan.