Generative Adversarial Network for Simulation of Load Balancing Optimization in Mobile Networks

Fu Jie Tey[1], Tin-Yu Wu[2*], Yueh Wu[3], Jiann-Liang Chen[1]

[1] Department of Electrical Engineering, National Taiwan University of Science and Technology, Taiwan
[2] Department of Management Information Systems, National Pingtung University of Science and Technology, Taiwan
[3] Department of Computer Science and Information Engineering, National Ilan University, Taiwan
D10907001@mail.ntust.edu.tw, tyw@mail.npust.edu.tw, willy821230ptr@hotmail.com.tw, Lchen@mail.ntust.edu.tw

Abstract

The commercial operation of 5G networks is almost ready to be launched, but problems related to wireless environment, load balancing for example, remain. Many load balancing methods have been proposed, but they were implemented in simulation environments that greatly differ from 5G networks. Current load balancing algorithms, on the other hand, focus on the selection of appropriate Wi-Fi or macro & small cells for Device to Device (D2D) communications, but Wi-Fi facilities and small cells are not available all the time. For this reason, we propose to use the macro cells that provide large coverage to achieve load balancing. By combing Generative Adversarial Network (GAN) with the ns-3 network simulator, this paper uses neural networks in TensorFlow to optimize load balancing of mobile networks, increase the data throughput and reduce the packet loss rate. In addition, to discuss the load balancing problem, we take the data produced by the ns-3 network simulator as the real data for GAN.

Keywords: 5G, Generative Adversarial Network (GAN), Load balance, Neural network

1 Introduction

When commercial 5G networks are going to be widely deployed, there are still have problems with associated with wireless environment, example like load balancing, need to be solved [1]. Previous load balancing techniques were implemented in simulation environments which were different from 5G networks [5, 24]. On the other hand, present load balancing methods are focus on the selection of appropriate Wi-Fi or macro & small cells for D2D communications.

Normally, small cells are deployed at the edge of macro cells to improve signal quality and mitigate interference, but it could cost a lot of money [2]. Before the small cell deployments, we must consider if the existing macro cells are enough to achieve load balancing. Unfortunately, it is hard to tell whether the macro cell's load influences the selection of macro cells.

In recent years, the load balancing techniques for wired networks have been widely applied with the flourishing Software-Defined Networking (SDN). Those methods have been relatively mature, but they are not suitable for wireless networks.

Nodes in wired networks are all connected through wired connections. To achieve load balancing in SDN, for example, a centralized controller is responsible for collecting data and directing data flows to the selected target. As for wireless networks, high path loss and various kinds of interference among base station (BS) and devices make wireless communications more difficult. Consequently, those load balancing methods are designed for wired networks are not feasible for wireless networks. Therefore, we propose to use the learning ability of neural networks to deal with load balancing problems in wireless networks.

Network load balancing is an important challenge in wireless networks. Too many users together with high data throughput at the same time could lead to severe performance degradation of BSs. Neither former load balancing algorithms implemented in simulation environments nor those designed for wired networks are suitable for current network status. For this reason, novel load balancing algorithms for wireless networks must be proposed. BSs in 5G mobile network environment include at least E-UTRAN Node B (eNBs), Wi-Fi BSs and millimeter-wave (mmWave) BSs, among which eNBs provide the greater coverage. There are many Wi-Fi BSs and mmWave BSs, but not all of them are available when it comes to load balancing. Therefore, this study proposes to solve the load balancing problem of BSs in order to improve the throughput and minimize the packet loss. Simulation are usually conducted to test load balancing results and the returned results will be values, like transmission capacity measured as kbps, or packet loss rate given as a percentage. All values are meaningful to researchers because values obtained from different simulations will be analyzed for further parameter adjustment. Based on the results, we can gradually improve the simulation and prove the correctness and advantages of our proposed method. As long as the simulation design is appropriate, we believe that neural networks can be used to learn and reward and we are able to find suitable neural networks for simulation environment. Without making modifications in the simulation environment, we normally input parameters, analyze simulation results and again input or change parameters. We hope that neural networks can learn to input parameters as well as interpret the simulation results. In order to achieve the goal, Generative Adversarial Network (GAN) is a good choice because GAN comprises two parts: the generator network to produce parameters and the discriminator network to distinguish between real and fake samples. The aim of this study is to

*Corresponding Author: Tin-Yu Wu; E-mail: tyw@mail.npust.edu.tw
DOI: 10.53106/160792642022032302010

integrate GAN with the load-balancing simulation system to achieve load balancing.

The rest of this paper is organized as follows: Section II reviews researches on load-balancing methods and background. Section III details how about system architecture and proposed method. In Section IV, we perform the experiment and simulate the parameters with our proposed models. Finally, we summarize our findings and results.

2 Background and Related Works

Load balancing approaches, Generative Adversarial Network (GAN) and SeqGAN are reviewed in Section II.

2.1 Load Balancing Approaches

To achieve load balancing in wired networks, the centralized controller monitors the loads of the inter-connected computers and routers and commands to coordinate traffic flow. In wire-line networks, a centralized controller is responsible for monitoring computers and routers in the network, and managing the data flow to achieve load balancing.

2.1.1 Least Load First

The "least load first" idea is the key point of load balancing and also the basis for weighted and standard deviation-based algorithms mentioned below. To achieve load balancing, the least load first algorithm finds the node with least load (normally a router or a base station) and routes traffic or users to this node.

2.1.2 Least Load First Standard-Deviation-Based

In the load balancing algorithm based on standard deviation, the centralized controller is responsible for gathering all routers' data and calculating each router's standard deviation. Based on the standard deviation that measures the spread of the data about the mean value, the algorithm can find the router with a lighter load, balance link loads in the network and reroute flows to avoid links with large loads as well as resolve congested paths [3, 18].

2.1.3 Weight-Based

The weight-based load balancing algorithm assigns a weight to each router and adjusts the weight based on the loading status of the present BS and other criteria [4, 15-16].

In addition, there are other ways to achieve load balancing, such as considers the operator utility and the user utility [17] or using the association to achieve adaptive load-balancing [4].

2.1.4 Mobile Networks and Wireless Networks

As discussed above, the load balancing methods designed for wired networks are not suitable for wireless networks, but still some have been presented to perform load balancing, including using a load-aware small-cell management mechanism to manage mobile devices in all small cells and transfer redundant mobile devices to other cells to alleviate congestion [5]; employing Device-to-Device (D2D) communications so that the data traffic can be effectively offloaded from a congested small cell to other underutilized small cells [6]; and utilizing a fog computing based architecture [7]. However, D2D is only applicable under certain conditions [6, 25]. To achieve load balancing, other methods proposed to select the access point based on Wi-Fi [4, 8, 14, 19], but Wi-Fi and mobile networks are very different in terms of signal strength, interference and other factors. Load balancing mechanisms for LTE systems have been presented much earlier [9, 23, 26], but current LTE simulation structures cannot fit the upcoming 5G networks.

2.2 Generative Adversarial Network (GAN)

The idea of GANs was introduced in 2014 [10] and a GAN consists of two neural networks: a generator network and a discriminator network, as shown in Figure 1. The generator network generates plausible data and the discriminator network distinguishes real and generated data. By competing against each other, both the two neural networks get better and better. For example, the generator network generates samples that have a high probability of being real samples while the discriminator network can more accurately distinguishes the generator's fake data from real samples. Nevertheless, the discriminator training can fail if the generator training is too quick; the generator training can fail if the discriminator is too good. GANs are hard to train because it is difficult to reach a convergence in the game [20]. For this reason, we must adjust parameters and use only part of real data. GAN later also has various variants like SeqGAN, DCGANs [13] and WGAN [12].

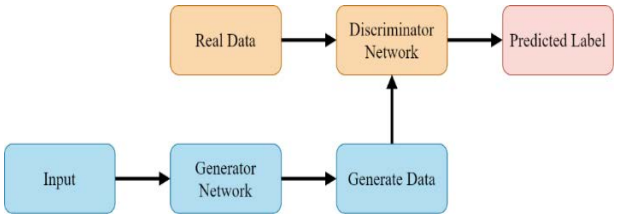

Figure 1. Generative Adversarial Network

2.3 SeqGAN

Sequence GAN (SeqGAN) was presented in 2016 [11] to help GAN handle discrete outputs and the illustration of SeqGAN is displayed in Figure 2. GANs can be used to deal with sequential data but not discrete data like languages or words. In order to cope with the problem, SeqGAN adds the Monte Carlo Tree Search (MCTS) to the discriminator network so that the reward from the discriminator on a complete sequence can be returned to the generator network as shown in Figure 2.

Note that SeqGAN is usually used for poetry generation and discrimination. Real data is generated by a trained neural network. Since this study integrates with the simulation network system, we will use the simulation data as reference.

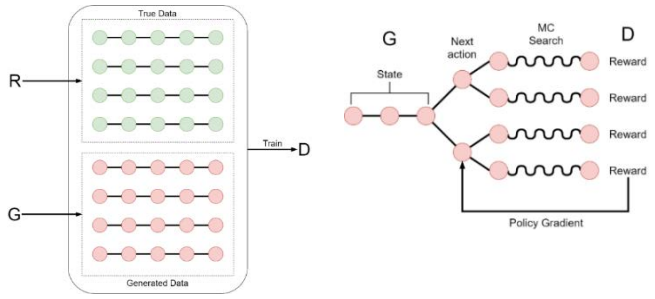

Figure 2. Diagram of SeqGAN

3 System Architecture and Proposed Method

Section 3 introduces how we combine our proposed load balancing system with GAN, which includes a generator network and a discriminator network. Figure 3 shows the system architecture. The reason we use GAN is that GAN generates results by competing the identification network with the generative network, so this means that there is almost no data before the load balancing, so GAN can be trained with a small amount of data to achieve our desired goal [21-22].

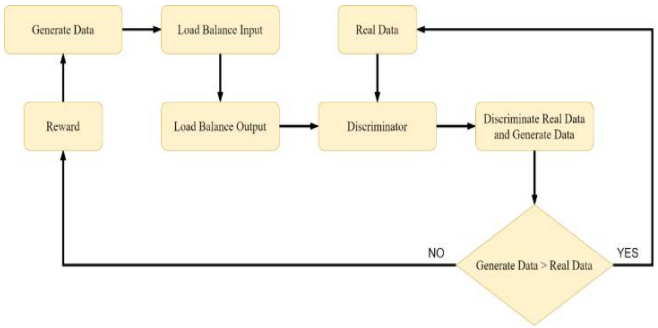

Figure 3. System architecture

3.1 Load Balancing Simulation Environment

In the load balancing simulation environment, the first step is to set and adjust the parameters, including number of users, number of BSs, simulation time and weighting parameters generated by GAN. The load balancing simulation environment can be created using self-motivated software or other specific network environment simulators. The common way to test network load balancing is to make the nodes fully loaded and then observe the performance of the load balancing method. In this study, we use the same way. Moreover, to prevent performance degradation on the system when the load balancing mode is on while not needed, our proposed method tests network traffic in the normal load mode to ensure the normal operation of the system. The results of the study, including throughput and packet loss rate, will be output after the simulation.

3.2 Integrating Load-Balancing Simulation Test into GAN

To integrate the load balancing simulation test into GAN, we correlate the results generated by the generator network to the parameters set in the load balancing simulation environment. Then, for output discrimination, the result of the load balancing simulation test will be sent back to the neural network.

3.2.1 Corresponding Parameters

To combine the load balancing simulation test with GAN, we need to correlate the results generated by the generator network to the parameters set in the load balancing simulation environment. Next, we will return the result of the load balancing simulation test to the neural network as feedback. Our method, based on SeqGAN, generates a one-dimensional array that contains twenty elements and uses them as the weighting parameters for selecting 20 BSs in the load balancing simulation environment. Once the simulation is completed, the result will be sent to the neural network. Instead of estimating packet loss rate, this study estimates throughput. Packet loss is calculated as a percentage and we may encounter the situation that the packet loss rate is similar but the throughput greatly differs. So, higher throughput signifies better performance of the load balancing simulation test. We then send the result back to the neural network.

3.2.2 Adjusting the SeqGAN

SeqGAN's generator network is exactly what our method needs: a one-dimensional array that contains twenty normally distributed random elements. The discriminator network is responsible for distinguishing true data from generated data. If the discriminator network tells that the generated data and true data are the same, it means that the data generated by the generator network have been very close to true data. True data for SeqGAN are poetry data from a trained RNN; however, the content does not meet our needs. For this reason, we will use the generated data that bring better result in the load balancing simulation as true data.

$$BN_{original} = \max\left(\{N_1, ..., N_n\}\right) \quad \textbf{(1)}$$

$$BN_{weight+\mathrm{RSRQ}} = \max\left(\{N_1 + W_1, ..., N_n + W_n\}\right) \quad \textbf{(2)}$$

3.3 How Weighting Parameters Affect BS Selection

To know how weighting parameters affect selection of BSs, we originally chose the BS with higher weights but soon found the method inefficient. The user may connect to a remote BS that has relatively low signal strength. Therefore, to select the best BS, we use the signal strength and the weight values generated by the generator network. The flowchart is displayed in Figure 4. First, the user verifies the present Reference Signal Received Quality (RSRQ), shown in (3), and evaluates whether to handover when the High Threshold (HT) is higher than the RSRQ. The N is the resource blocks.

$$\mathrm{RSRQ} = \mathrm{N} \times \mathrm{RSRP} / \mathrm{RSSI} \quad \textbf{(3)}$$

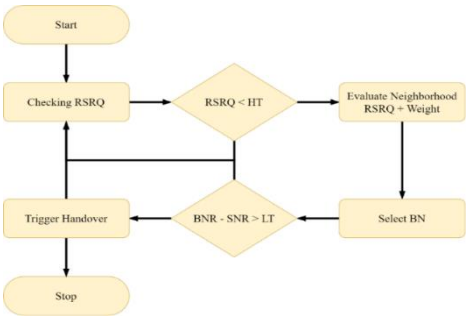

Figure 4. BN selection based on weights

On the original handover algorithm in (1) will evaluate all neighborhood to find the best RSRQ as Best Neighborhood (BN) for the handover BS. Based on the weights and the RSRQ also will evaluate the RSRQ but will include the weight that train with GAN to find the BN, the equation shown in (3). In case that a cell phone switches frequently between two BSs with similar signal strength, the handover will be triggered when the Serving Neighborhood RSRQ (SNR) minus the Best Neighborhood RSRQ (BNR) is larger than the Low Threshold (LT).

4 Results and Analysis

Section 4 describes how Tensorflow combines with ns-3, analyzes the training process of GAN, and runs the load balancing test in both normal and high load for performance evaluation. For SeqGAN simulation, we use GTX 1080Ti as the GPU to attain neural network acceleration.

4.1 Combining Tensorflow with ns-3

Figure 5. Combing TensorFlow with ns-3

This study includes two parts: the neural network TensorFlow and the ns-3 network simulator. To combine TensorFlow with ns-3, we have to considerate their structures. As an open-source Python library for machine learning, TensorFlow supports Python. The simulator ns-3, written in the C++ language, makes use of the waf build system with optional Python bindings. It would be easier to combine TensorFlow with ns-3 if both of them adopt Python. However, most mobile network scenarios in the ns-3 network simulator are written in C++ and so is the structure of ns-3. If the system structure must be modified, the programming language will be mainly C++. Therefore, we combine the neural network in Python with the network simulation system in C++. Since the network simulation system must be inserted into the neural network, the neural network in Python will connect to the ns-3. The system first uses Python to modify the target parameters in ns-3 and, second, uses Python to run the ns-3 script. Finally, the simulation result will be saved. Figure 5 shows the flowchart to combine TensorFlow with ns-3.

4.2 Results of GAN Training

It is difficult for a GAN to converge because of the interplay between the generator and the discriminator that may lead to unexpected convergence. Although the SeqGAN used in this study is a trained neural network that ensures convergence, we still evaluate its convergence due to the modification of the target parameters. To know if a neural network converges, we usually check the test loss, which is also an index to examine SeqGAN convergence. Table 1 and Table 2 show the SeqGAN generator and the discriminator parameter which use in the training.

Table 1. Generator parameter

Embedding Dimension	16
Hidden Layer	32
Sequence Length	20
Batch Size	64

Table 2. Discriminator parameter

Embedding Dimension	20
Dropout Probability	20%
Batch Size	64

Table 3. Basic parameters for load-balancing simulation test

Load Mode	Normal Load	High Load
UE amount	20	40
BS amount	20	20
BS dBm	46 dBm	46 dBm
Simulation Time (sec.)	10s	10s

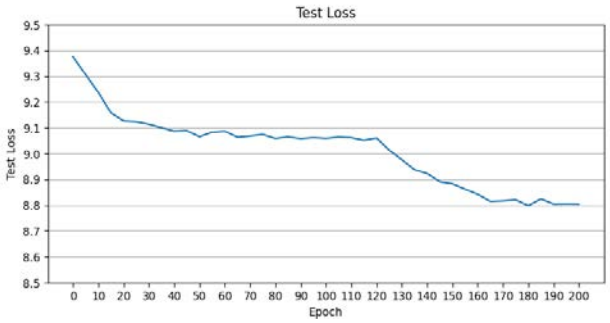

Figure 6. Test loss

If the test loss keeps steadily decreasing during the training process, the neural network is still learning. If the test loss remains flat, there might be overfitting or bottleneck problems and the neural network must be adjusted again. If the test loss keeps increasing, there must be something wrong with neural network design or parameter settings. In Figure 6, the test loss curve that refers to the convergence in the training of neural network may go up and down but decreases steadily, denoting that the neural network is learning successfully and

converging. Based on the SeqGAN design, the number of Epoch is set to 200.

4.3 Load Balancing System

The ns-3 simulator is used to conduct the simulation. We know that it will be easier to use self-motivated software for GAN, but the design of simulation environment may be wrong. For this reason, the ns-3 simulator that has been extensively recognized is adopted to prove the feasibility to combine GAN and simulation environment. The load balancing simulation test can be conducted in two ways: in normal load or high load. Table 3 shows the basic parameters for the load balancing simulation test. According to different load modes, the number of user equipment (UE amount) is adjusted: 20 in normal load and 40 in high load, in case the simulation is too long. The time to run the simulation is set to 10 seconds only. There are 20 BSs in both modes because we want to present a large load balancing environment so that UE can be distributed in a large area in normal load. A large simulation environment also provides more choices to UEs to reflect performance variability of different modes. Packet size BS dBm is decided based on the ns-3 simulator default setting.

Figure 7 shows the simulation scenario, in which BSs are deployed based on triangles in a region of 2750m x 1800m. In normal load mode, users are distributed randomly in the environment. In high load mode, 40 users are gathered near one single BS and weights are adjusted to achieve load balancing.

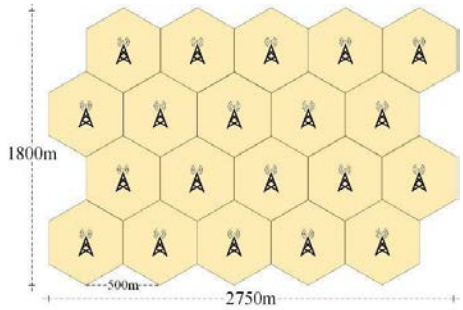

Figure 7. BS deployments in the simulation environment

Next, we briefly explain why the simulation includes 40 UEs. The upper load threshold of BS may be influenced by many factors. Figure 8 shows the signal strength-based BS throughput test conducted on 10 to 40 users. When 20 UEs connect to the same BS at the same time, the BS is unable to process more messages. Even more UEs join, the throughput cannot increase, causing the packet loss increase incessantly, as displayed in Figure 9. Therefore, to run the load-balancing simulation, there must be at least 20 UEs. The number of UE amount is set to 40 in case the simulation takes too much time. Moreover, our simulation data is generated by the ns-3 simulator when all UEs are keeping sending packets. Not all users must be connected to the BS to reach the upper load threshold for the simulation.

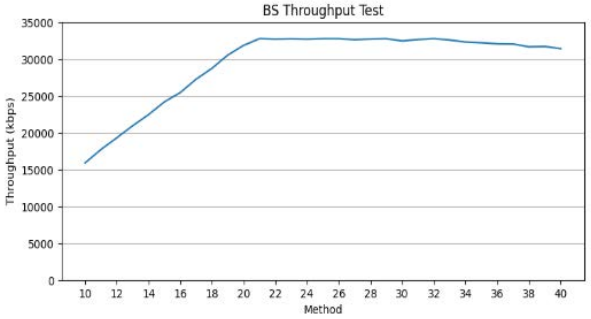

Figure 8. BS throughput test

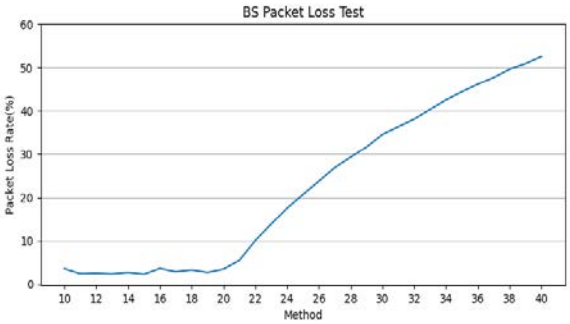

Figure 9. BS packet loss test

4.4 Load Balancing Simulation

To evaluate the performance of the load balancing simulation test, we examine whether load balancing works efficiently. If load balancing works, the network bottleneck due to high load will be postponed, the throughput will increase and the packet loss will be reduced.

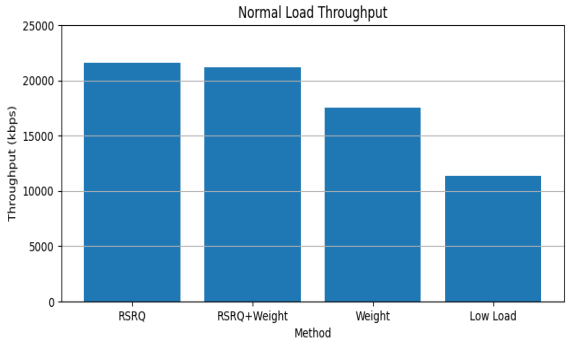

Figure 10. Normal load throughput

During the simulation test, we compare four methods, including RSRQ, RSRQ+Weight, Weight and Low Load, in both normal load and high load. Weight and RSRQ Weight are based on the GAN generator. As shown in Table 1, in the normal load mode, 20 users are randomly distributed in the map. Figure 10 displays that in terms of data throughput, RSRQ is similar to RSRQ+Weight. After neural network training, the Weight method is able to achieve basic throughput performance; however, due to the strong impact of randomness, it fails to find a better solution. The Low Load method is the worst-performing because it easily connects to the BS that provides worse signal strength than the current one.

Figure 11 shows that in terms of packet loss rate, the RSRQ and RSRQ+Weight methods are also similar, approximately 34 to 35%, and the Weight method is unable to

yield better result as well. The Low Load method easily connects to the farther BSs, making the packet loss rate even worse.

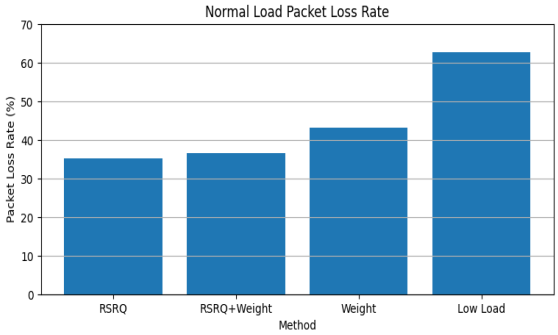

Figure 11. Normal load packet loss rate

In the normal load mode, the packet loss rate is comparatively higher. In our simulation, the ns-3 simulator uses the UDP protocol to deliver 100 packets/per second to each user, but the outcome must be determined by the quality of the connection between users and BSs. Those packets that cannot be delivered in time will be categorized as packet loss. The result reveals that in the ns-3 system, BSs all have their upper limit thresholds in delivering packets. In addition, when the UE is located between two BSs and the RSRQ is bad, high QAM levels cannot be achieved for packet delivery. The better the RSRQ is, the higher the QAM level is and the higher the data throughput will be. So, if 256-QAM cannot be currently maintained, the system will change to 640-QAM, making it difficult to reach higher throughput. Such a situation often occurs at the fringe of the BS coverage area with higher interference level and therefore higher packet loss rate. This is the reason why the maximum throughput in the test is different from the fore-mentioned BS throughput.

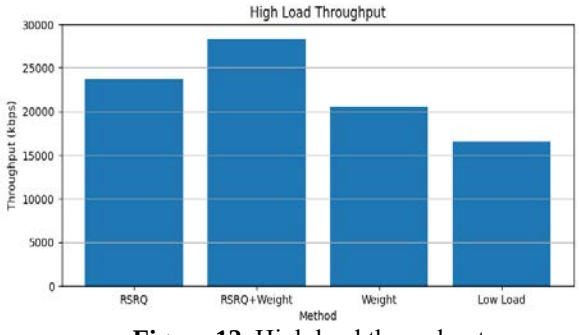

Figure 12. High load throughput

In the high load mode, according to Figure 12, the RSRQ+Weight method achieves the highest throughput. The RSRQ and Weight methods are getting close but still not stable. Although the Low Load method yields good performances in some scenarios, it is deeply affected by vacant BSs with poor signal strength and is unable to get better result.

As for the packet loss rate in the high load mode, the RSRQ+Weight method performs best because of the lowest packet loss rate, and the RSRQ is the second. The RSRQ and Weight methods are getting close as well but both fail to find better results. The Low Load method remains the worst and its packet loss rate is even worse than that in the normal load mode because of too long distances and interferences as shown in Figure 13.

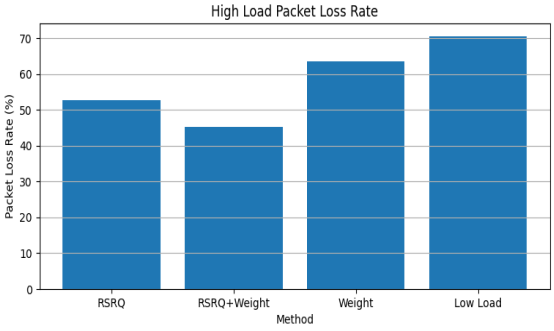

Figure 13. High load packet loss rate

To use the original SeqGAN for the training in our simulation takes approximately 4 hours. The simulation time for the ns-3 simulation, on the other hand, is decided according to the number of users and the preset simulation time. Since we do not use GPU to speed up computation, more simulation time and more users will certainly increase the time to finish the ns-3 simulation.

4 Conclusion

In this paper, we use GAN and the ns-3 simulator to present a load balancing optimization simulation method, in which the generator network is responsible for generating weighting parameters for the load balancing system and the discriminator network is responsible for discriminating the results. Moreover, the better load balancing data is adjusted and taken as real GAN data for load balancing optimization. We also bring up the possibility to integrate SeqGAN with the load balancing system and to use GAN "reward" to replace real data. Our proposed method performs well in normal load, and also increases throughout about 4000kbps and reduces about 6% packet loss in high load.

Although we successfully combine GAN with load balancing simulation test, the simulation may take too much time. To speed up the simulation, we can build our own simulation software according to our needs and improve CPU usage for optimized performance. At present, the existing SeqGAN is the most suitable one for our research goal. To sum up, three major contributions of this paper include: (1) using GAN in the network simulation system, (2) integrating Tensorflow with ns-3, and (3) increasing 3141kbps throughput in the load balancing test.

Acknowledgment

Thanks to the Ministry of Science and Technology for providing industry-university cooperative research projects to complete the paper: MOST 110-2221-E-020-023, MOST 107-2221-E-197-007 -MY3 and MOST 108-2321-B-197-004.

References

[1] B. B. Sánchez, Á. Sánchez-Picot, D. S. De Rivera, Using 5G technologies in the Internet of Things:

Handovers, Problems and Challenges, *9th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, Santa Catarina, Brazil, 2015, pp. 364-369.

[2] I. P. Chochliouros, A. Kostopoulos, I. Giannoulakis, A. S. Spiliopoulo, M. Belesioti, E. Sfakianakis, A. Kourtis, E. Kafetzakis, Using Small Cells for Enhancing 5G Network Facilities, *IEEE NFV-SDN - Fourth Workshop on Network Function Virtualization and Software Defined Networks*, Berlin, Germany, 2017, pp. 264-269.

[3] Y.-L. Lan, K. Wang, Y.-H. Hsu, Dynamic load-balanced path optimization in SDN-based data center networks, *10th International Symposium on Communication Systems, Networks and Digital Signal Processing*, Prague, Czech Republic, 2016, pp. 1-6.

[4] C.-Y. Lin, W.-P. Tsai, M.-H. Tsai, Y.-Z. Cai, Adaptive Load-balancing Scheme Through Wireless SDN-based Association Control, *IEEE 31st International Conference on Advanced Information Networking and Applications*, Taipei, Taiwan, 2017, pp. 546-553.

[5] Y.-C. Wang, K.-C. Chien, A load-aware small-cell management mechanism to support green communications in 5G networks, *The 27th Wireless and Optical Communications Conference*, Hualien, Taiwan, 2018, pp. 1-5.

[6] H. Zhang, L. Song, Y. J. Zhang, Load Balancing for 5G Ultra-Dense Networks Using Device-to-Device Communications, *IEEE Transactions on Wireless Communications*, Vol. 17, No. 6, pp. 4039-4050, June, 2018.

[7] J. Jijin, B.-C. Seet, P. H. J. Chong, H. Jarrah, Service Load Balancing in Fog-based 5G Radio Access Networks, *IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications*, Montreal, QC, Canada, 2017, pp. 1-5.

[8] N. Kiran, C. Yin, Z. Akram, AP Load Balance Based Handover in Software Defined WiFi Systems, *5th International Conference on Network Infrastructure and Digital Content*, Beijing, China, 2016, pp. 6-11.

[9] Y. Yang, P. Li, X. Chen, W. Wang, A High-efficient Algorithm of Mobile Load Balancing in LTE System, *IEEE Vehicular Technology Conference*, Quebec City, QC, Canada, pp. 1-5.

[10] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, *Generative Adversarial Networks*, arXiv:1406.2661v1, June, 2014.

[11] L. Yu, W. Zhang, J. Wang, Y. Yu, SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient, *The Thirty-First AAAI Conference on Artificial Intelligence*, San Francisco, CA, USA, 2017, pp. 2852-2858.

[12] H. Liu, X. Gu, D. Samaras, Wasserstein GAN With Quadratic Transport Cost, *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea, 2019, pp. 4831-4840.

[13] A. Radford, L. Metz, S. Chintala, *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*, arXiv:1511.06434, January, 2016.

[14] W. K. Soo, T.-C. Ling, A. H. Maw, S. T. Win, Survey on Load-Balancing Methods in 802.11 Infrastructure Mode Wireless Networks for Improving Quality of Service, *ACM Computing Surveys*, Vol. 51, No. 2, pp. 1-21, March, 2019.

[15] K. Addali, M. Kadoch, Enhanced Mobility Load Balancing Algorithm for 5G Small Cell Networks, *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, Edmonton, AB, Canada, 2019, pp. 1-5.

[16] C. Yi, X. Zhang, W. Cao, Dynamic Weight Based Load Balancing for Microservice Cluster, *The 2nd International Conference on Computer Science and Application Engineering – CSAE18*, Hohhot, China, 2018, pp. 1-7.

[17] K. M. Addali, S. Y. B. Melhem, Y. Khamayseh, Z. Zhang, M. Kadoch, Dynamic Mobility Load Balancing for 5G Small-Cell Networks Based on Utility Functions, *IEEE Access*, Vol. 7, pp. 126998-127011, September, 2019.

[18] N. Hassan, X. Fernando, An Optimum User Association Algorithm in Heterogeneous 5G Networks Using Standard Deviation of the Load, *Electronics*, Vol. 9, No. 9, Article No. 1495, pp. 1-20, September 2020.

[19] D. Qin, P. Ji, S. Yang, T. M. Berhane, An efficient data collection and load balance algorithm in wireless sensor networks, *Wireless Networks*, Vol. 25, No. 7, pp. 3703-3714, October, 2019.

[20] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, A. A. Bharath, Generative Adversarial Networks: An Overview, *IEEE Signal Processing Magazine*, Vol. 35, No. 1, p. 53-65, January, 2018.

[21] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, T. Aila, *Training Generative Adversarial Networks with Limited Data*, arXiv:2006.06676, October, 2020.

[22] S. Shao, P. Wang, R. Yan, Generative adversarial networks for data augmentation in machine fault diagnosis, *Computers in Industry*, Vol. 106, pp. 85-93, April, 2019.

[23] T.-Y. Wu, W.-F. Weng, Reducing handoff delay of wireless access in vehicular environments by artificial neural network-based geographical fingerprint, *IET Communications*, Vol. 5, No. 4, pp. 542-553, March, 2011.

[24] T.-Y. Wu, T. Chang, Interference Reduction by Millimeter Wave Technology for 5G-based Green Communications, *IEEE Access*, Vol. 4, pp. 10228-10234, August, 2016.

[25] W. K. Lai, Y. U. Chen, T.-Y. Wu, Analysis and Evaluation of Random-Based Message Propagation Models on the Social Networks, *Computer Networks*, Vol. 170, Article No. 107047, April, 2020.

[26] T.-Y. Wu, W. Chen, W.-T. Lee, F.-H. Liu, Y.-P. Huang, SeMIPv6: Secure and Efficient Roaming Mechanism Based on IEEE 802.11r and MIH, *Journal of Internet Technology*, Vol. 11, No. 7, pp. 909-916, December, 2010.

Biographies

Fu Jie Tey completed his master 's degree at National Ilan University‘s Department of Computer Science and Information Engineering. He is a PhD student in the Department of Electrical Engineering at National Taiwan University of Science and Technology. His research interests are in machine learning applications, mobile networks, Internet of Things, information security, and programming. His favourite activity is coding in spare time.

Tin-Yu Wu currently works as a Full Professor in the Department of Management Information Systems, National Pingtung University of Science and Technology, Taiwan. He received his M.S. and Ph.D. degrees in the Department of Electrical Engineering, National Dong Hwa University, Hualien, Taiwan in 2000 and 2007 respectively. His research interests focus on the big data analytics, cloud computing and mobile computing.

Yueh Wu received his M.S. degrees in the National Ilan University‘s Department of Computer Science and Information Engineering. Yilan. His research interests focus on the mobile computing, 5G communication and IoTs.

Jiann-Liang Chen was born in Taiwan on December 15, 1963. He received the Ph.D. degree in Electrical Engineering from National Taiwan University, Taipei, Taiwan in 1989. Since August 2008, he has been with the Department of Electrical Engineering of National Taiwan University of Science and Technology, where he is a distinguished professor now. His current research interests are directed at cellular management, IoT system design and cyber-security.