Non-parametric Decision-Making by Bayesian Attractor Model for Dynamic Slice Selection

Tatsuya Otoshi[¶§], Shin'ichi Arakawa[†§],Masayuki Murata[†§], Takeo Hosomi[‡§]
[¶] Graduate School of Economics, Osaka University
1-7 Machikaneyama-Cho, Toyonaka, Osaka 565-0043, Japan
[†]Graduate School of Information Science and Technology, Osaka University
1-5 Yamadaoka, Suita, Osaka 565-0871, Japan
[‡]Data Science Laboratories, NEC Corporation
1753 Shimonumabe, Nakahara-ku, Kawasaki, Kanagawa 211-8666, Japan
[§] NEC Brain Inspired Computing Research Alliance Laboratories, Osaka University
1-5 Yamadaoka, Suita, Osaka 565-0871, Japan

***Abstract*—In 5G, the network is divided into slices to provide communications with different characteristics, such as low latency and reliable communications (URRLC), multiple connections (MTC), and high speed and high capacity communications (eMBB), for different applications. Although the selection of network slices is often static, in practice, dynamic slice selection is required depending on the application situation. However, there are issues such as the slice change itself changing the application situation and the delay associated with the slice change. In this paper, we realize dynamic slice selection by recognizing the rough situation and the mapping between the recognized situation and the slice. The Bayesian Attractor Model (BAM) is used for recognition to achieve consistent recognition and is extended to the Dirichlet Process Mixture Model (DPMM) to achieve automatic attractor construction. The mapping between situations and slices is also automatically learned by using feedback. As an application of dynamic slice selection, we also show slice selection based on the video streaming situation. Through numerical examples, we show that our method can keep the quality of video streaming high while reducing slice changes.**

***Index Terms*—Bayesian Attractor Model, Dirichlet Process Mixture Model, MPEG-DASH**

I. Introduction

Virtualization of network resources and construction of virtual networks are becoming more and more popular to flexibly support diverse services [1], [2]. In 5G, the network is divided into slices to provide communication with different characteristics such as low latency and reliable communication(URRLC), multiple connections(MTC), and high speed and large capacity communication(eMBB) for different applications [3]. In general, slices are pre-built and the communication is mapped to the appropriate slice according to the characteristics of the application type [3]. This makes it possible to provide network quality for different communication requirements even on the same network.

Network slice selection is often static, but in practice, dynamic slice selection is required depending on the application situation. For example, in the case of the Tactile Internet [4], whether low latency is required depends on how fast the operator's hand moves. If the hand movement is slow, then some delay is acceptable. Providing low latency even in such a slow-moving case would impose excessive communication costs on the user. Dynamic slice selection is also useful for video streaming [5]. Also during streaming, if there are enough videos in the buffer to be read ahead of time, it will not affect the streaming quality even when the download is delayed. At such moments, high-capacity communication is unnecessary, and it is desirable to temporarily switch to a slice that provides low-cost communication.

However, there are several difficulties involved in dynamic slice selection. First, since the slice change itself changes the application situation, it is difficult to identify the application situation and the appropriate slice for that situation. It is difficult to assume the interaction between the slice change and the application in advance, and therefore the overall objective function cannot be defined in advance. This makes normal optimization methods unrealistic in dynamic slice selection. In addition, additional slices are expected to be added in future extension of 5G, and it is required to make appropriate choices when new slices are constructed. Second, there is a cost associated with changing slices. Since changing a slice involves changing the configuration of devices on the network, there is a delay until the change is completed [4]. In addition, changing the slice changes the route on the network, which may cause a decrease in the throughput of TCP [6]. For this reason, frequent slice changes are undesirable.

In this paper, dynamic slice selection is achieved by learning to recognize a rough situation and the mapping between the recognized situation and the slice. Our research group is investigating a method for network control using the Bayesian Attractor Model (BAM), which makes decisions in noisy observation information. By using BAM, it is possible to make choices that avoid unnecessary slice changes even in a fluctuating environment. However, BAM needs to be given the assumed situation as an attractor in advance. Previous research [7], [8] has proposed methods for adding and updating attractors, but in all cases, the number of attractors and the conditions for adding them must be given by the designer. It is difficult to assume every situation in advance because the application situation varies with time and slice selection. Therefore, it is necessary to build attractors automatically based on the data without assuming any prior assumptions. In the field of clustering, there is a non-parametric method called Dirichlet Process Mixture Model (DPMM) [9], which does not assume the number of classes or class features in advance. Thus, we extend the BAM with DPMM, called *extended BAM* to automatically estimate attractors, including the number of attractors and automatically construct attractors from observed data. Then, for the constructed attractor, extended BAM learns the mapping to the appropriate slice by feedback.

As an application example to the proposed dynamic slice selection using extended BAM, we also study dynamic slice selection in video streaming. On the video player, Adaptive Bitrate (ABR) is usually running, and the bitrate of the video changes according to the streaming situation [10]. The ABR also reacts during slice selection, and the situation changes in a complex manner. By using extended BAM, appropriate slice selection is performed according to the video streaming situation. Through evaluation using numerical examples, we show that extended BAM is capable of appropriately recognizing situations without knowing the algorithm of the ABR running on the video player. We also show that extended BAM can maintain high bitrate streaming while suppressing excessive slice changes by automatically acquiring the granularity of the situation to be recognized.

The structure of this paper thereafter is as follows. In Section II, we summarize related slice selection research. In Section III, we propose a dynamic slice selection method using DPMM+BAM. In Section IV, we describe the application of the dynamic slice selection method to streaming. In Section V, we evaluate the behavior of the proposed method during streaming through numerical examples. Section VI presents the conclusion of this paper.

II. Related Work

In 5G, functional requirements are defined according to the application, and slices are built in advance according to each requirement. Slice selection is expected to be provided as a function of the base station when the device is accessed.

Slice selection methods considered in existing research include (1) rule-based slice selection [3], (2) slice selection based on the status of the slice at the start of the session [11], [12], and (3) dynamic slice selection based on the application status [5].

In rule-based slice selection, the performance requirements for a slice are defined in advance, and the slice is selected fixedly according to the type of application. Specifically, at the start of a session, the device sends a Service Description Document (SDD) describing the type of service, such as video, voice, eMBB (high speed and large capacity), URLLC (high reliability and low latency), MTC (multiple connections), etc., and assigns slices according to the SDD [3].

Reference [11] proposes a method for allocating an appropriate slice at the start of a session based on the performance requirements of the slice and the actual performance of the slice. In this method, the metrics (latency, throughput, etc.) of each slice are evaluated by weighting them, and the slice with the best evaluation is assigned. In [12], slice selection including the selection of the base station to which the device connects is done by solving an optimization problem. In the optimization problem, the allocation is done by maximizing the degree to which the transmission rate requested by the user is satisfied. Since this is a combinatorial optimization problem, a heuristic solution method using a genetic algorithm is used.

All of the above assumes that a slice is selected at the beginning of a session and the same slice is used until the session ends. In reference [5], a method to dynamically select and switch slices depending on the video streaming status is studied. Specifically, two types of slices with different characteristics are constructed: one is a fast slice with low latency but a small number of concurrent connections, and the other is a normal slice with high latency but a large number of concurrent connections. The high-speed slice is used at the start of video loading, and a method is proposed to shorten the video loading time and achieve a high response. The video streaming status is monitored by receiving notifications from the device. When the video is loaded sufficiently and the buffer is filled, the slice is returned to the normal slice.

III. Dynamic Network Slice Selection with extended BAM

In order to select appropriate slices while reducing slice changes, we propose a method that roughly recognizes the situation and learns the appropriate slice mapping for the recognized situation. In this method, we use extended BAM, which is an extension of BAM, to automatically construct attractors from observed information and recognize the situation. By clustering the situations, consistent control is possible even when noise is included. One may think that a moving average can deal with noise problem, but if it fluctuates around a control threshold, the control will oscillate. Also, discretizing similar situations together increases the affinity with learning using feedback.

Then, the system learns the correspondence between the recognized situation and the slice based on the control feedback. The system then learns the correspondence between the recognized situation and the slice based on the control feedback [7], [8]. This allows the system to automatically acquire recognition and control at an appropriate granularity.

A. Dynamic Network Slice Selection

Network slicing is a technology for building a flexible virtual network to meet the different performance requirements of each application. In most cases, slices are statically assigned for each application [3], but dynamic slice selection is required to respond to changing application conditions [4], [5] In the literature [4], they target the Tactile Internet and dynamically switch slices to ensure communication with appropriate latency depending on the speed of hand movements during teleoperation.

In 5G, it is assumed that slices will be constructed for each use case, such as eMBB (high speed and large capacity), URLLC (high reliability and low latency), and MTC (multiple connections), and that slices will be allocated according to the performance requirements of each service. On the other hand, it may be necessary to dynamically switch slices depending on the application situation, for example, using multiple connections daily and switching to high-speed, high-capacity communication depending on the situation.

Slice selection is performed by a slice controller located at the edge. In the case of 5G, the slice controller is deployed as a function of the base station.

In the case of static slice selection, slice selection is done by the user device only at the start of communication. The user device sends the application information to the slice controller at the start of communication and the slice controller assigns the appropriate slice to the user device [3].

In the case of dynamic slice selection, the user device sequentially notifies the slice controller of the application status, and the slice controller switches slices according to the status [4], [5]. On the user device, the requirements for communication vary depending on the application and the user's behavior. For example, in the case of Tactile internet [4], if the user's hand is moving quickly, low latency is required, while some latency is acceptable if the hand is moving slowly. In the case of video streaming, the bitrate of the video can be dynamically switched by the Adaptive Bitrate (ABR) algorithm or manual operation on the user device. On the other hand, frequent switching is undesirable because slice switching causes delays [4] and, communication instability due to route switching. Therefore, it is necessary to switch slices according to the situation while absorbing minor variations by capturing similar situations in a somewhat rough manner. Thus, we use BAM, which is tolerant to fluctuation of observations, to recognize the situation.

B. BAM with Variable Attractors

In the original BAM, the number of attractors must be determined in advance and the representative values of the attractors must be given. DPMM [9] is a model that also estimates the number of classes and the representative values of the classes from the data, and by incorporating this mechanism in BAM, the number of attractors and the representative values of the attractors can be estimated from the data.

Similar to DPMM, we extend BAM to have a variable number of attractors by the following model, where the discriminative label generation model is a Chinese restaurant process.

$$\boldsymbol{x_t} = M_t\sigma(\boldsymbol{z_t}) + \boldsymbol{u_t} \tag{1}$$

$$\boldsymbol{z_t} = f(\boldsymbol{z_{t-1}}) + \boldsymbol{v_t} \tag{2}$$

$$p(\phi_t = k) = \begin{cases} \frac{n_k}{n-1+\alpha} \; (k \le K_t) \\ \frac{\alpha}{n-1+\alpha} \; (k > K_t) \end{cases} \tag{3}$$

where $M_t = (\mu_t^1, \mu_t^2, \cdots, \mu_t^{K_t})$ is a matrix of representative values, $\boldsymbol{u_t}, \boldsymbol{v_t}$ is the noise term for the observed and state values, $f(z_t)$ is the Hopfield dynamics, σ is the sigmoid function, n_k is the number of data classified into attractors k at time t, K_t is the estimated number of attractors at time t, n_k is the number of data classified into attractor k, $n = \sum_k n_k$ is the total number of data, and α is the parameter of the Chinese restaurant process.

C. Merge and Delete Attractors

The Chinese restaurant process does not include any operations to reduce the number of attractors since the new data can be classified either into existing attractors or into new attractors. When classifying data offline in a batch-processing manner, the likelihood is evaluated by looking at the entire data, so the appropriate number of attractors is set even in such a case. However, in the case of online estimation, the only decision is whether to maintain or increase the number of attractors at the time, so the number of attractors will continue to increase over time and unnecessary attractors will remain.

Therefore, the number of attractors should be adjusted appropriately in the online estimation by adding operations to reduce the number of existing attractors. The operations to reduce the number of attractors include merging several similar attractors into one and deleting unnecessary attractors, and each operation is described in detail below.

1) Merge of Attractors: If there are attractors with similar representative values, it is appropriate to classify them as the same attractor without distinguishing them. In this case, the representative values of each attractor and the number of classified data in the Chinese restaurant process can be inherited, so that the data of each attractor can be reused. The following procedure is used to integrate the attractors.

1) Calculate the distance of the representative value between attractors
2) Perform integration in the following steps when there is an attractor pair a, b ($a < b$) whose distance is less than the threshold
3) Merge attractor a into attractor b. The merged attractor will inherit the following values
    - $n_{a+b} = n_a + n_b$
    - $\mu_{a+b} = \frac{n_a\mu_a + n_b\mu_b}{n_a+n_b}$
4) Replace attractor b with attractor K_t.
5) Delete attractor K_t and decrement K_t.

2) Deletion of Attractors: When an attractor is newly created for an outlier, it may remain as an unnecessary attractor with little or no data classified in it. When such unclassified attractors exist, they can be deleted to avoid the number of attractors continuing to increase. However, since the deletion of an attractor has a large impact on the loss of information on the data classified in that attractor, the frequency of the operation should be adjusted probabilistically so that it is not performed frequently. The procedure for deleting an attractor is as follows

1) Execute with a certain probability for each time step
2) Perform deletion if there is an attractor d whose number of classifieds n_d is less than the threshold.
3) Replace attractor d with attractor K_t.
4) Delete the attractor K_t and decrement K_t.

Unlike merging attractors, deleting an attractor does not reuse the previous data, so it is desirable to reduce the number of attractors by merging attractors as much as possible. Therefore, when merging attractors and deleting attractors at the same time, priority should be given to merging attractors.

IV. Application of Slice Selection in Video Streaming

Since it is easy to build an evaluation environment by using the existing configuration of the application, we consider the application scenario of dynamic slice selection. We will use video streaming as the subject of our application, as in the literature [5].

A. Adaptive Bitrate

In a streaming, a mechanism called adaptive bitrate to change the bitrate according to the communication status and continue streaming is common. In MPEG-DASH [10], [13], which is one of adaptive bitrate streaming method, the video delivery server holds video files that have been divided into segments at fixed time intervals in advance, with multiple profiles of different audio and video rates. The video player on the user's device can change the bitrate dynamically during video streaming by specifying a specific video profile and downloading the video segment from the server. The video profile may be selected manually by the user according to his/her preference, but in most cases, Adaptive Bitrate (ABR) is used, in which the video player automatically selects the bitrate based on throughput and other measurement information.

In the event of congestion on the communication channel, ABR can automatically change the bitrate to a lower bitrate to allow video streaming to continue. However, if the throughput drops below the minimum bitrate profile, even if the bitrate is changed by ABR, the video cannot be avoided from stopping. It is more desirable to be able to avoid congestion and maintain the bitrate by switching slices.

In this way, there is a limit to the number of situations that can be handled by the ABR alone, and it is necessary to switch slices according to the streaming situation. In this case, it is necessary to recognize the situation including the behavior of the ABR to select an appropriate slice, but since the behavior of the ABR depends on the implementation of the player, it is difficult to assume the behavior pattern in advance. Therefore, it is desirable to use extended BAM to dynamically estimate the number of attractors and their representative values.

B. Slice Selection by extended BAM

To select a slice according to the streaming situation, the streaming situation is recognized by extended BAM. The observed values and attractors of extended BAM are mapped as follows.

a) Input/Observation: When selecting the bitrate for ABR, the input is often either the length of the read buffer or the throughput or both. In addition to the buffer length and throughput, the input to extended BAM is the bitrate selected by ABR and the residual bandwidth per slice. That is, we use the following variables as input of extended BAM.

- b_t : Length of video in buffer
- r_t : Throughput
- R_t : Bitrate selected by ABR
- B_t^i : Residual bandwidth of slice i.

Let $\boldsymbol{x_t} = (R_t, b_t, r_t, B_t^1, \cdots, B_t^S)$ be the observed values at time t of extended BAM.

b) Attractors: An attractor holds a representative value of the observed value, $\boldsymbol{\mu^i}$, and the combination of these values represents a particular streaming situation. In practice, according to the extended BAM model, the number of attractors and their representative values are estimated from the data.

C. Mapping streaming status to slice selection

If the current streaming situation is found by extended BAM to be the situation corresponding to attractor i, then If the current streaming situation is found to correspond to attractor i by extended BAM, the appropriate slice is selected according to the situation. In this case, it is necessary to map which slice is appropriate for which streaming situation.

One may think a rule-based mapping is possible to map slices and situations. For example, if the buffer length and bitrate are both small, the congestion cannot be handled by ABR, and the slice is switched to a faster slice. However, due to the lag of the slice switching itself, or the time required to download the segment even after switching, the video may not be played in time and will stop. In addition, since the bitrate can be changed by ABR by triggering the slice switching, it is not necessary to change the bitrate immediately after the current status is recognized. The bitrate can be changed by ABR by triggering a network slice switch, which may cause an immediate change from the perceived current state.

Therefore, it is necessary to modify the mapping so that it can be based on rule-based mappings, but can perform appropriate switching while predicting the outcome of slice switching. This can be achieved by learning by feeding back the results of slice selection for a recognized situation as a reward [8].

In this streaming case, the reward F_t is defined as follows.

$$F_t = R'_t - C(s_t) \tag{4}$$

where R'_t is the min-max normalized bitrate R_t and $C(s_t)$ is the min-max normalized communication cost [4] of slice s_t which is selected by slice selection. When the buffer is empty and streaming stops, R_t equals to 0. In this case, switching the slice and resuming streaming immediately will increase the reward per unit time. On the other hand, if there are enough buffers and switching slices does not affect the streaming bitrate, then selecting a slice with lower communication cost will result in a higher reward. Note here that the cost of slice switching is not included in the reward. The effects caused by slice switching include the effects with other controls such as TCP, but it is difficult to quantify these interactions. However, in the proposed method, the number of switching can be reduced by the consistency of the perception of the situation.

V. Evaluation

We verified the operation of a method for recognizing the status of streaming using extended BAM in streaming distribution through numerical examples.

A. Environment

We use extended BAM to recognize and classify the streaming situation on a player with continuously varying throughput and automatic bitrate selection by ABR.

For streaming settings, we assumed that the video had seven profiles (1, 3, 5, 10, 15, 30, and 50 Mbps) ranging from 1 Mbps to 50 Mbps and that it was divided into segments of 4 seconds each. It was also assumed that the maximum buffer length of the video player was 30 seconds of video.

To study the behavior during congestion and recovery from congestion, the streaming starts with sufficient throughput (1 Gbps), gradually decreases the throughput (1 Mbps), and then gradually recovers the original throughput.

The ABR is running on the video player, and automatically changes the bitrate of the video player in response to changes in throughput. The following ABR is used.

a) Throughput-based ABR: When the length of the loaded buffer of the video player is below/above the threshold value (5 sec. / 25 sec.), it shall be set to the maximum bitrate among the profiles that have a bitrate less than or equal to the throughput at that time.

B. Recognition Results

Figure 1 shows the classification results of extended BAM for each observation. The figure plots the scatter plot of the observations with the bitrate as the vertical axis and the read buffer length as the horizontal axis. The classified attractors are indicated by the color of the plotted points, and in the case of Figure 1, the attractors are classified into four types: black, yellow, green, and red. The dotted lines in the figure connect consecutive observations. The results of the classification of DPMM alone are shown in Figure 2.

The DPMM does not classify well, but the extended BAM classifies in a rough but meaningful way. Specifically, in extended BAM, black attractors indicate the best streaming conditions with the highest bitrate. As the throughput decreases, the buffer length decreases, and the ABR switches to streaming at a lower bitrate, which is classified as a yellow attractor. When throughput drops further, the lowest bitrate is used, which is classified as a green attractor. When the throughput starts to recover, the bitrate is gradually increased by ABR, and this recovery state is classified as a red attractor.

The reason why DPMM alone does not classify well is thought to be that it does not work consistently to classify successive inputs in the same way. In extended BAM, the BAM state model produces a somewhat consistent classification for successive inputs, whereas DPMM lacks such consistency. In Fig. 2, we can also see that the classification sometimes changes in detail for consecutive inputs connected by dotted lines. In this way, even if the data is classified into a new class, the data classified into that class will not be accumulated. As a result, the largest class that already exists becomes dominant, and most of them are considered to be classified as the same class.

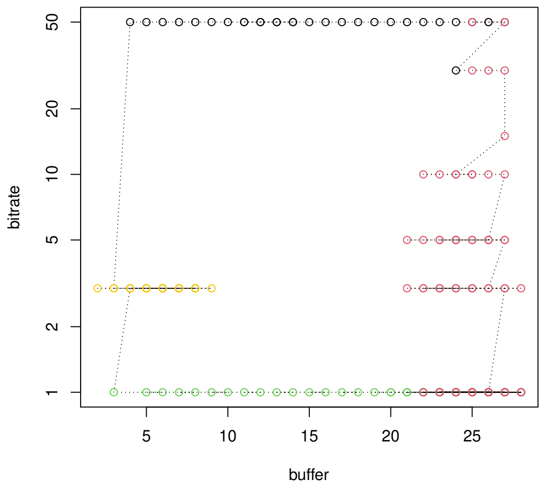

Fig. 1. Classification result of extended BAM(ABR: throughput-based)

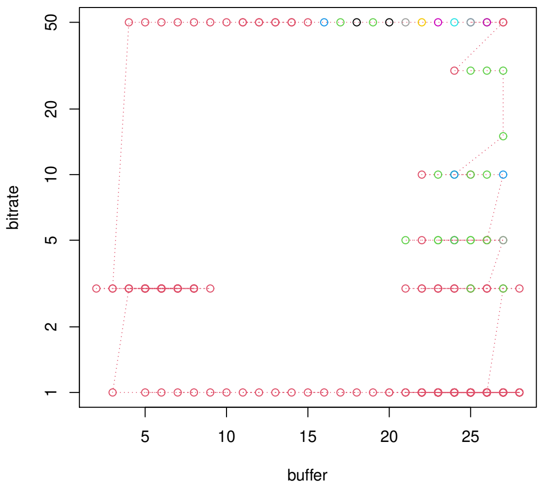

Fig. 2. Classification Result of DPMM（(ABR: throughput-based)

C. Slice Selection Results

We also evaluate the dynamic slice selection with recognized results to confirm that slice selection based on extended BAM can preserve the bitrate of the video while suppressing slice changes.

Figure 3 represents the result of not performing dynamic slice selection and continuing to use the slow slice. In the figure, the throughput of the selected slice at each time, the downloaded buffer length, and the bitrate selected by ABR are plotted. There are two types of slices: low-speed slices and high-speed slices. The throughput of the low-speed slices varies with time, while the high-speed slices can communicate at a constant high throughput. However, high-speed slices are more cost-expensive, and it is preferable to use low-speed slices unless it affects the quality of video streaming. The mapping between the perceived situation and the slice is learned by using the video bitrate and cost as feedback. When a slow slice is continuously used, ABR switches to low bitrate streaming when the throughput drops. In order for the bitrate to recover, it waits for the throughput to recover on the slice. Therefore, the streaming at low bit rate continues for a long time.

Figure 4 shows the time series of the video streaming status when slice selection is performed based on the recognition by extended BAM. To see the results of learning, the third round of repeating the same situation is plotted in the figure. Also, Figure 5 shows the results when DPMM is used for classification. The ABR was throughput-based.

From the figure, it can be seen that when extended BAM is used, the time for the bitrate to decrease is kept to a short time, while the number of slice switching is also reduced. In the case of using extended BAM, the slice is switched to a slice with higher throughput at the time when the buffer length is shortened and the bitrate switching occurs due to ABR. In DPMM, the situation is not well classified, resulting in frequent slice switching. Also, in DPMM, the delay in slice switching results in a period of low bitrate that lasts for about ten seconds. In extended BAM, there is also a momentary drop in bitrate, but this is quickly addressed by switching slices, thereby minimizing video quality degradation. Ideally, the bitrate drop should be predicted in advance and the slices should be switched in advance. One of the future work is to realize slice selection based on the prediction.

VI. Conclusion

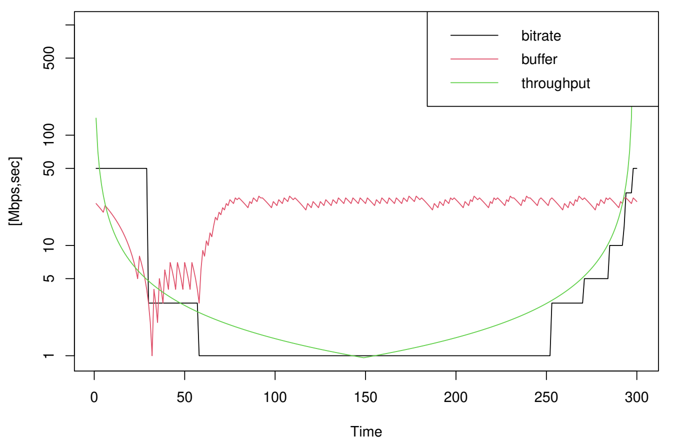

Fig. 3. Time series of player status with slice selection without slice selection (ABR:throughput-based)

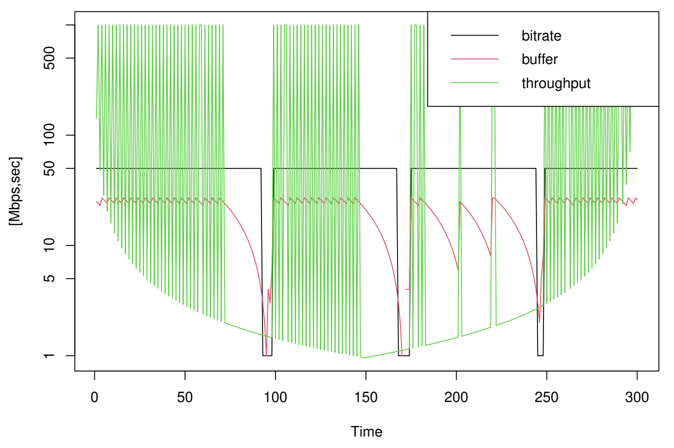

Fig. 5. Time series of player status with slice selection using DPMM (ABR:throughput-based)

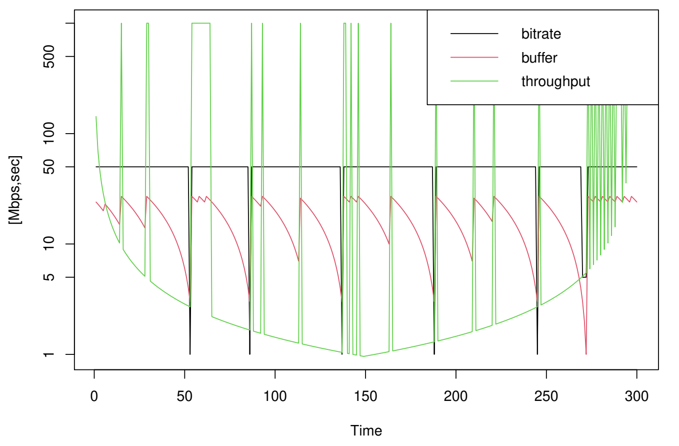

Fig. 4. Time series of player status with slice selection using extended BAM (ABR:throughput-based)

In this paper, we proposed a method for dynamic slice selection by learning the recognition of a rough situation and the mapping between the recognized situation and the slice. While the use of BAM enables consistent recognition, the extension to DPMM enables automatic attractor construction. In addition, we added the integration and deletion of attractors to maintain only the necessary number of attractors. The system then learns the appropriate slice for the perceived situation using feedback. Evaluation using numerical examples showed that extended BAM can be used to reduce the number of slice changes while reducing the quality degradation of video streaming.

Future work includes the realization of slice selection that considers the control lag by incorporating prediction mechanism.

References

[1] I. Alam, K. Sharif, F. Li, Z. Latif, M. Karim, S. Biswas, B. Nour, and Y. Wang, “A survey of network virtualization techniques for Internet of Things using SDN and NFV,” *ACM Computing Surveys (CSUR)*, vol. 53, no. 2, pp. 1–40, 2020.

[2] H. Cao, H. Hu, Z. Qu, and L. Yang, “Heuristic solutions of virtual network embedding: A survey,” *China Communications*, vol. 15, no. 3, pp. 186–219, 2018.

[3] V. K. Choyi, A. Abdel-Hamid, Y. Shah, S. Ferdi, and A. Brusilovsky, “Network slice selection, assignment and routing within 5G networks,” in *2016 IEEE Conference on Standards for Communications and Networking (CSCN)*. IEEE, 2016, pp. 1–7.

[4] K. Polachan, B. Turkovic, T. Prabhakar, C. Singh, and F. A. Kuipers, “Dynamic network slicing for the tactile Internet,” in *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2020, pp. 129–140.

[5] M. Yamamoto and S. Nishimura, “Latest trends in network slicing utilization technology (translated),” *NHK R&D*, no. 178, pp. 4–11, 2019.

[6] R. Carpa, M. D. de AssunçÃo, O. Glück, L. Lefèvre, and J.-C. Mignot, “Evaluating the impact of SDN-induced frequent route changes on TCP flows,” in *2017 13th International Conference on Network and Service Management (CNSM)*. IEEE, 2017, pp. 1–9.

[7] T. Otoshi, S. Arakawa, M. Murata, K. Wang, T. Hosomi, and T. Kanoh, “Acquiring new categories by self data gathering with Bayesian attractor model,” in *2020 IEEE International Conference on Human-Machine Systems (ICHMS)*. IEEE, 2020, pp. 1–4.

[8] Tatsuya Otoshi and Shin’ichi Arakawa and Masayuki Murata and Kai Wang and Takeo Hosomi and Toshiyuki Kanoh, “Method for flexible updating of attractors in virtual network topology control with Bayesian attractor model,” in *Proceedings of IEEE International Conference on Communications(to be presented)*, online, June 2021.

[9] M. Liu, L. Wang, and R. Siegwart, “DP-Fusion: A generic framework for online multi sensor recognition,” in *2012 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE, 2012, pp. 7–12.

[10] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, “A control-theoretic approach for dynamic adaptive video streaming over HTTP,” in *Proceedings of SIGCOMM*, 2015, pp. 325–338.

[11] G. Zhao, S. Qin, G. Feng, and Y. Sun, “Network slice selection in softwarization-based mobile networks,” *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 1, p. e3617, 2020.

[12] B. Bakmaz, Z. Bojkovic, and M. Bakmaz, “Topsis-based approach for network slice selection in 5G mobile systems,” *International Journal of Communication Systems*, vol. 33, no. 11, p. e4395, 2020.

[13] I. Sodagar, “The mpeg-dash standard for multimedia streaming over the internet,” *IEEE multimedia*, vol. 18, no. 4, pp. 62–67, 2011.