Representation of QoS Conditions at Observation Nodes that Satisfy the Required QoE at User End Nodes

Ayumi Araragi, Akio Watanabe, Hiroki Ikeuchi, Yousuke Takahashi, Taichi Kawano and Masahiro Yokota
NTT Network Service Systems Laboratories, NTT Corporation, Tokyo, Japan
Email: {ayumi.araragi, akio.watanabe, hiroki.ikeuchi, yousuke.takahashi, taichi.kawano, masahiro.yokota}@ntt.com

***Abstract*— This paper proposes a method for concisely representing the Quality of Service (QoS) requirements necessary to achieve user-desired Quality of Experience (QoE), even when these requirements are distributed in a complex shape within a high-dimensional space, in a form that can be utilized by control systems. Conventional QoE–QoS relationship models have been based on observations at a single node and primarily used simple threshold-based condition expressions. However, in control systems utilizing multiple network observation points, more complex and flexible expressions are required. In this study, we propose a method to obtain a set of QoS conditions that satisfy QoE as a point cloud, approximate its outline using as few hyperplanes as possible, and represent it with multiple convex polytopes. The proposed method is based on divide-and-conquer and shows superior performance in both accuracy and representation simplicity compared to conventional convex hull-based methods. Moreover, the resulting polytope representation is effective for application to network control algorithms and provides practical design guidelines.**

***Index Terms* —QoS, QoE, mapping**

I. Introduction

This study addresses the challenge of concisely representing complex Quality of Service (QoS) conditions in a high-dimensional space that satisfy the Quality of Experience (QoE) required by users. Such representation is crucial for enabling automated control in practical network management.

Conventional QoE–QoS relationship models have primarily been analytical based on measurements at a single node and assuming application-specific characteristics [1][2]. However, from a network operation perspective, it is often necessary to derive control operations based on QoS information obtained from multiple observation nodes within the network to achieve the QoE required by end users. In this case, the relationship between QoS and QoE strongly depends on the network configuration and settings, making explicit analytical model difficult.

On the other hand, in recent years, methods that use machine learning models to predict QoE from QoS have been attracting attention [3][4]. By utilizing such models, it is also possible to consider an approach in which points that satisfy QoE conditions are sampled as a point cloud in QoS space.

However, to directly utilize the point cloud obtained from the learning model for control tasks, further approximation and representation of its shape are required. In particular, to inversely derive a set of QoS conditions that satisfy the QoE, the outer shape of the point cloud must be represented as a constraint region.

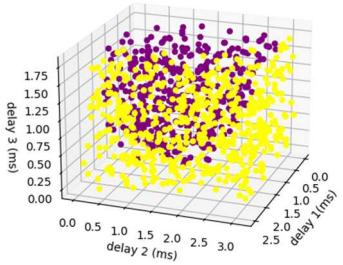

Fig. 1. Point cloud in QoS space (delays observed at three nodes)
Yellow: QoE satisfied Purple: QoE violated

As shown in Fig. 1, even in a simple case, the point cloud in QoS space that satisfies the QoE condition forms a complex shape. Such a shape cannot be adequately represented by hyperrectangular regions defined by per-axis QoS thresholds.

In network control, control operations to achieve QoE can be formulated as a problem of searching for an effective region within the QoS condition space. Many optimization-based control methods (e.g., Model Predictive Control, MPC) express such constraint regions as convex polytopes to ensure computational efficiency and theoretical analyzability [5]. Therefore, methods that approximate point clouds with simple polytopes with high accuracy can be an important component in control system design.

In this study, we propose a method that approximates the outline of a point cloud representing QoE-satisfying QoS conditions with as few hyperplanes as possible and constructs multiple convex polytopes based on these hyperplanes. This method follows a divide-and-conquer strategy to capture local structures of the point cloud, while merging hyperplanes as needed to achieve a balance between simplicity of representation and approximation accuracy.

The contributions of this study can be summarized in the following two points.

First, we propose a novel polytope construction method that accurately approximates the boundary of a set of QoS measurement points satisfying a required QoE, using a small number of hyperplanes.

Second, through comparative evaluation against representative existing approaches, we demonstrate that the proposed method achieves a balance between approximation accuracy and representational simplicity—an essential requirement for practical network control.

978-3-903176-75-1 ©2025 IFIP

This paper is organized as follows. Section II reviews related research and clarifies the position and novelty of the proposed method. Section III describes the proposed method's algorithm and parameters in detail. Section IV compares the proposed method with related methods in terms of performance evaluation and characteristics related to approximation accuracy and simplicity of expression.

II. Related Works

Research on the QoS–QoE relationship has mainly addressed forward mapping (QoS→QoE). Generic models [1] and learning-based predictors such as Pensieve [2] achieved accurate QoE estimation, but they remain application-specific or black-box in nature.

Recently, attention has shifted toward inverse or QoE-driven approaches. Al-Azzeh et al. [3] directly mapped QoE to QoS using spline approximation in cellular networks. Yan et al. [4] optimized resource allocation in semantic communication by treating QoE as the target, and Tang et al. [6] proposed traffic aggregation that adapts to heterogeneous QoE requirements. These works highlight the potential of QoE→QoS research, but they are either scenario-dependent or lack general, interpretable region modeling.

To represent feasible QoS conditions, hyperrectangular thresholds [7] and decision-tree partitioning [8] are widely used due to simplicity, yet they fail to achieve both accurate separation and concise representation. Greedy hyperplane approximation [9] improves flexibility but suffers from instability.

In control engineering, convex polytopes are standard in Model Predictive Control (MPC) [5], and recent works embed QoE into control, e.g., QoE-aware congestion control [10] or reinforcement learning-based resource management [11][12]. However, such studies rely on implicit mappings.

Our work differs by providing a general polytope-based representation of QoS regions satisfying user QoE, combining approximation accuracy with interpretability and enabling direct integration into automated network control.

III. Proposed Method

A. Overview of the Proposed Method

QoS measurement values from multiple observation nodes are combined into a single tuple. This QoS tuple space is uniformly distributed, and QoS values that achieve the target QoE and those that do not are classified using the mapping from the observation node QoS to the user-end QoE. The classified point clusters are referred to as QOS+ and QOS-, respectively. Next, multiple hyperplanes approximating the boundary of the QOS+ region are sought in this high-dimensional space. To simplify the representation, the number of hyperplanes is kept as small as possible.

The method for finding hyperplanes is as follows. First, the entire space is divided into small blocks. In each block, if QOS+ and QOS- are mixed, a hyperplane that separates QOS+ and QOS- is found using margin maximization. If the slopes of the hyperplanes in adjacent blocks are similar, the blocks are merged and the process is repeated to find separating hyperplanes for the merged blocks. This process is repeated to approximate the boundary using a small number of hyperplanes.

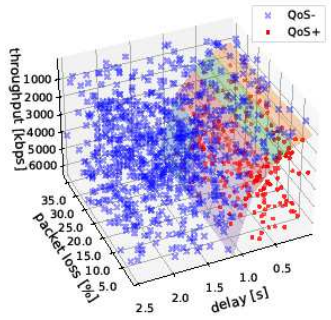

Fig. 2. An example of applying the proposed method

Finally, the first-order inequalities obtained from each boundary approximation hyperplane are combined into multiple simultaneous linear inequalities to represent QOS+. Fig. 2 shows an example of this method. This is a simple case where the observation node is at a single location, the data flow to be observed is a single stream, and the observed QoS is delay, packet loss, and throughput. It can be seen that QOS+ and QOS- are separated by five hyperplanes.

B. Details of the Proposed Method

Here, we describe the proposed method in detail, following its overall flow. First, in 1) and 2) below, we describe the QoS–QoE mapping assumed by the proposed method and the procedure for generating point clouds (QOS+) using it. These approaches are considered reasonable based on recent research trends and are not central to the originality of this study. On the other hand, in 3) and beyond, we propose a new boundary extraction method to precisely and concisely approximate polytopes for this point cloud, which constitutes the main contribution of this study.

1) Mapping that predicts QoE from the QoS of multiple observation nodes

This proposed method assumes that a mapping from QoS at observation nodes to QoE is given, where the QoE at the user end is estimated using the measured values of multiple types of QoS (delay, packet loss rate, throughput, etc.) for multiple services at multiple observation nodes as input. Using this mapping, we construct the mapping from QoE to QoS in the opposite direction. The input of the given mapping is expressed in the form of a tuple $(qos_1, \ldots, qos_n)$ that combines all measured QoS values. We call this tuple *observed QoS*. The output QoE is expressed as a tuple $\left(qoe_p^q\right)_{p,q}$ that combines all the QoE qoe_p^q expressed in Mean Opinion Score for each service q at each user end node p.

Observed QoS that satisfies the target QoE is defined as follows. Suppose that the target value of QoE v_p^q for node p and service q is given. Then, the observed QoS that provides

an output qoe_p^q that satisfies ($qoe_p^q \geq v_p^q$) for all p and q is called *'satisfying the target QoE'*.

As an example of the construction of the underlying mapping, the following is possible under certain assumptions. First, assume that an existing mapping from QoS to QoE is given on the same node, and further assume that QoS measurement is possible at not only the observation nodes but also the user end nodes. In this case, we create a mapping from observation-node QoS to user-end QoS by simultaneous measurement. We can also relate the user-end QoS to QoE using the existing mapping. Then we can obtain the data of the relation between the observation-node QoS and QoE. With these data, we can construct the desired mapping by machine learning.

2) Using the learning model to express the set of observed QoS values that satisfy the target QoE as point cloud QOS+ (Monte Carlo method)

First, a point cloud is generated uniformly at random in the n-dimensional observation QoS space. Each point is input into the mapping from observed QoS to QoE, and the target QoE is judged to be satisfied or not based on the output results. The sets of all points that satisfy and do not satisfy the target QoE are classified as QOS+ and QOS-, respectively.

3) Derivation of multiple hyperplanes that approximate the boundary of QOS+

Here, we show a method for approximating the boundary of QOS+ with as few hyperplanes as possible based on the principles of margin maximization and divide-and-conquer. The basic procedure is as described in the overview. The pseudo-code for the algorithm is shown in Algorithm 1. Before executing the pseudo-code, the space of observed QoS is divided into small blocks.

a) QoS space partitioning and symbol definitions

For each tuple, which is a point in the observation QoS space, its elements are assumed to be represented by real values. First, the observation QoS space $\mathbb{R}^n$ is divided into a number of hyperrectangles with the same number of division for each axis. Each range that can be taken by each qos_i ($i = 1, 2, \dots, n$) is divided into m parts, and then $\mathbb{R}^n$ is divided into m^n parts. Each of these hyperrectangles is called a *block*. Let $[s_i, e_i]$ (where s_i and e_i are the minimum and maximum possible values of qos_i respectively) be the interval corresponding to each block, and let the size of each side of the block be $d_j = (e_j - s_j)/m$. Then, each block is defined by index $j_1, \dots, j_n$ (j_i is an integer, $0 \leq j_i \leq m-1$) as follows.

$b_{j_1,\dots,j_n} = \{(x_1, \dots, x_n) \in \mathbb{R}^n | \; d_i \, j_i \leq x_i \leq d_i \, (j_i + 1) \}$

The set of all blocks is denoted by $\mathcal{B}$. The sets of QOS+ and QOS- points contained in block b are written as $B(b)^+$ and $B(b)^-$, respectively. For a set of blocks B, the set of all QOS+ points contained in B is written as $B+ = \bigcup_{b \in B} B(b)^+$, and the set of all QOS- points is written as $B- = \bigcup_{b \in B} B(b)^-$.

b) Explanation of the behavior of the pseudo-code

The following describes the details of each step in the pseudo-code for Algorithm 1. The algorithm uses the hyperparameters shown in Table I. Boundaries that were finely divided are gradually integrated during the execution of the algorithm.

Algorithm 1 Algorithm for Separating the Boundaries of QOS+ in the Observation QoS Space

Step1 Let $\mathcal{B}$ be the set of all blocks, and let S be the set of all boundaries. The initial state of S is defined as follows.
$S = \{\{b\} | \; b \in \mathcal{B}, p < |B(b)^+|/(|B(b)^+| + |B(b)^-|) < 1 - p\}$

Step 2 For each boundary $\{b\}$ in S, we find a hyperplane that separates $B(b)^+$ and $B(b)^-$ based on the idea of maximizing the margin, and call this the hyperplane associated with that boundary.

Step 3 Choose two adjacent boundaries Bd_1 and Bd_2 from S. Let n_1 and n_2 be the unit normal vectors of the hyperplanes associated with Bd_1 and Bd_2, respectively.

Step 4 **If** the inner product $n_1 \cdot n_2 > c$ **then**
    find a hyperplane separating
    $Bd_1^+ \cup Bd_2^+$ and $Bd_1^- \cup Bd_2^-$ and call it H.
    If the separation degree of $H > r$ **then**
        replace $Bd_1 \cup Bd_2$ with Bd_1, Bd_2 as the new boundary and return to step 3.
    else
        return to step 3.
else if there is an adjacent boundary that was not selected in step 3 **then**
    return to step 3.
else
    end processing.

Step 1) For each block $b \in \mathcal{B}$, if $B(b)^+$ and $B(b)^-$ contained therein are included in nearly equal parts within an error of |1/2 - p|, as specified by condition formula S, the block is called a boundary block between QOS+ and QOS−. Each set consisting of one boundary block is defined as the first set of boundaries, S. Note that each boundary is a set of blocks.

Step 2) For each boundary block b, we find the hyperplane that separates $B(b)^+$ and $B(b)^-$ based on the idea of maximizing the margin. We call this hyperplane the *hyperplane associated with the boundary block*.

Step 3) To express the boundary with fewer hyperplanes, we merge boundaries. We select a pair of adjacent boundaries (Bd_1, Bd_2). Here, two boundaries Bd_1, Bd_2 are said to be adjacent if there are two blocks $b_1 \in Bd_1$ and $b_2 \in Bd_2$ such that b_1 and b_2 are adjacent. Two blocks, each with index $b_{i_1,\dots,i_n}$ and $b_{i\prime_1,\dots,i\prime_n}$ are said to be adjacent if, for some j, $|\, i_j - i\prime_j| = 1$ and $i_k = i\prime_k$ (for all k where $k \neq j$).

Step 4) Consider the merging of the hyperplanes associated with the two adjacent boundaries (Bd_1, Bd_2), i.e., the merging of the boundaries, as follows.

Boundaries are merged when (i) the cosine of the angle between the two hyperplanes is greater than the threshold c, and (ii) the separation degree is greater than the threshold r. (i) is determined by whether the inner product of the unit normal vectors n_1 and n_2 of each hyperplane is greater than c. If the cosine value of the angle is determined to be greater than c, the two boundaries are tentatively merged. On (ii), if (i) is

satisfied, find a hyperplane that separates $Bd_1^+ \cup Bd_2^+$ and $Bd_1^- \cup Bd_2^-$ for the tentatively merged boundary. If the merged boundary can be separated by the hyperplane with a threshold r or more, which is predetermined as the hyperparameter, the two boundaries merging is determined. The hyperplane is considered to be associated with the boundary.

By continuing the above merging operation until no new merging of boundaries occurs, we obtain a small number of hyperplanes that approximate the boundaries.

c) Parameter description

The proposed method has four hyperparameters: partition density *m*, mixture threshold *p*, plane merge threshold *c*, and post-merge separation *r*.

Partition density *m*: Divides each axis into *m* parts, yielding m^d blocks in *d*-dimensional space. Larger *m* improves boundary resolution but increases cost.

Mixture threshold *p*: Identifies boundary blocks by the ratio *r* of QOS+ to QOS−. A block is on the boundary if $p \leq r \leq 1-p$. Larger *p* enforces stricter detection but risks boundary defects.

Merge threshold *c*: Neighboring hyperplanes with cosine similarity above *c* are merged. Larger *c* demands stricter alignment, improving accuracy but increasing plane count.

Separation *r*: Finalizes a merge only if the separation between merged planes exceeds *r*, controlling how smoothly curved boundaries are approximated.

These parameters govern the trade-off between accuracy and simplicity (number of hyperplanes) and should be tuned to the point-cloud structure and control requirements.

4) Expression of the set of observation QoSs that satisfy the target QoE using linear inequalities

For the hyperplanes $l_1, l_2, \ldots, l_m$ obtained in 3), and the hyperplanes $l_{m+1}, l_{m+2}, \ldots, l_{m+2n}$ defined by the lower and upper bounds of each qos_i, we denote the two linear inequalities derived from each l_i, with opposite directions as l_i+ and l_i-. For each *i*, either l_i+ or l_i- is selected, and a system of $m + 2n$ linear inequalities is constructed. If the ratio of QOS+ points contained in the region defined by this system exceeds a predefined threshold, the system is accepted. This process is repeated for all possible combinations of selections, and the set of accepted systems is used as the representation of the set of observed QoS conditions that satisfy the target QoE.

Each system of linear inequalities defines a convex polytope, and the resulting representation is a set of such convex polytopes.

IV. Performance Evaluation and Analysis

A. Performance Evaluation of the Proposed Method

1) Comparison methods

To evaluate the relative accuracy of the proposed method, we compared it with the three existing approaches mentioned in the introduction. (i) separation using a single hyper-rectangle, (ii) separation using multiple hyperrectangles, and (iii) separation using multiple hyperplanes by the greedy method.

TABLE I
List of hyperparameters used

m	Number of partitions for each QoS
p	Ratio of QOS+ and QOS- of blocks
c	Maximum value of the inner product of unit normal vectors of two hyperplanes that are judged to be close in angle
r	Minimum value of the separation degree of QOS+ and QOS- of hyperplanes that allow merging

The polytope approximation procedure for QoS point clouds in each method is as follows:

Single hyperrectangle method: Each axis of the d-dimensional space in which the point cloud exists is discretely divided, and the entire space is approximated with a single hyperrectangle. Lower and upper limits for each dimension are explored by grid search to maximize the F1 score.

Multiple hyperrectangle method: We used DecisionTreeClassifier from sklearn.tree to partition the space into multiple hyperrectangular regions based on split conditions at each node. This method can model nonlinear and complex boundaries but is prone to overfitting.

Hyperplane separation using the greedy method: First, a single hyperplane is used to separate QoS+ and QoS- with a linear classifier. New hyperplanes are iteratively added to regions with large errors, forming a polytope-based separation region. We used scikit-learn's LogisticRegression for implementation.

Each of these methods has different characteristics. A single hyperrectangle is effective when the target point set has a simple shape, but it has limitations in representing complex boundaries. Multiple hyperrectangles allow for flexible representation, but there is a risk of model complexity due to overfitting. The greedy method is prone to accuracy variability due to the order of hyperplane addition and local optima.

From these perspectives, we quantitatively evaluate the proposed method in comparison with the others in terms of various aspects.

2) Generation of test data

The accuracy and behavior of the proposed and comparison methods heavily depend on the characteristics of the input QoS point cloud. Therefore, it is important to use evaluation data that appropriately reflect those characteristics when comparing the methods. In this study, test data were generated through simulation using the following procedure.

Network configuration: The network configuration to be evaluated is shown in Fig. 3, consisting of nine intermediate nodes (*n1* to *n9*), two server nodes (*S1* and *S2*), and two user end nodes (*E1* and *E2*). Each node is connected by links shown in the figure, and relay nodes and user nodes are assigned fixed-size receive buffers, while links are assigned fixed bandwidths.

Traffic model and simulation: Network utilization varies depending on the amount of data transmitted from the server

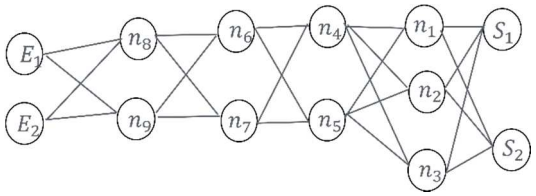

Fig.3. Structure of target network

TABLE II
OBSERVATION DESCRIPTION OF EACH OBSERVATION SUITE

Observation name	Observation node (observation data flow)
Obs.1	$n_1(S_1 \to E_1, S_1 \to E_2), n_9(S_2 \to E_2),$ $n_6(S_1 \to E_1, S_2 \to E_1, S_2 \to E_2)$
Obs.2	$n_4(S_1 \to E_1,\ S_2 \to E_1\ , S_2 \to E_2),$ $n_9(S_1 \to E_2, S_2 \to E_1\ , S_2 \to E_2)$
Obs.3	$n_1(S_1 \to E_1,\ S_1 \to E_2), n_2(S_2 \to E_1),$ $n_9(S_1 \to E_2, S_2 \to E_1\ , S_2 \to E_2)$
Obs.4	$n_4(S_1 \to E_1,\ S_2 \to E_1\ , S_2 \to E_2),$ $n_6(S_1 \to E_1,\ S_2 \to E_1\ , S_2 \to E_2)$
Obs.5	$n_4(S_2 \to E_1, S_2 \to E_2), n_5(S_1 \to E_2),$ $n_6(S_1 \to E_1, S_2 \to E_1\ , S_2 \to E_2)$
Obs.6	$n_6(S_1 \to E_1, S_2 \to E_1, S_2 \to E_2),$ $n_7(S_1 \to E_2), n_8(S_1 \to E_1), n_9(S_2 \to E_1\ )$
Obs.7	$n_1(S_1 \to E_2), n_2(S_2 \to E_1), n_3(S_2 \to E_2\ ),$ $n_4(S_1 \to E_1, S_2 \to E_1, S_2 \to E_2)$
Obs.8	$n_6(S_2 \to E_2), n_7(S_1 \to E_2), n_8(S_1 \to E_1\ ),$ $n_9(S_1 \to E_2, S_2 \to E_1\ , S_2 \to E_2)$

to the end nodes and the routing. Traffic is modeled based on the following probability distributions:

- Number of session occurrences: Poisson distribution
- Session duration: Weibull distribution
- Session start time: Uniform distribution

To reflect these variations, 20 types of user usage patterns were prepared. The routing was fixed throughout the experiment.

QoS measurement and observation patterns: QoS was measured using a custom event-driven simulator. This simulator is a simplified version of standard simulators such as ns-3. In the evaluation, we used eight observation patterns shown in Table II. In each pattern, we measured the delay for multiple data flows (e.g., *S1→E1*) at specific nodes. The QoS vector at each observation point consists of six dimensions.

Construction of QoS point clouds: In total, 160 scenarios were generated by combining each observation pattern and user usage pattern. In each case, a Multi-Layer Perceptron (MLP) was trained using QoS and corresponding QoE data to construct a model for predicting QoE. Subsequently, uniform sampling was performed in the QoS space using the Monte Carlo method, and the MLP was used to determine whether QoE was satisfied, thereby generating the required QoS point cloud.

Here, QoE is defined as "satisfied" when each service provided at each user end node meets the specified QoS criteria. The prediction accuracy of the MLP in each case is shown in Table III, and we confirmed that the prediction accuracy was high in all cases at the selected observation points. In general, low prediction accuracy indicates that the data flow affecting QoE is not being properly observed, suggesting that the selection of measurement locations and observation targets is extremely important.

Note that various distribution parameters were empirically determined to ensure diversity in user variation.

3) Performance comparison

Based on the experimental setup described above, we evaluated the performance of the proposed method and three comparison methods. The evaluation results for each method are shown in Tables IV–VII. For each of the 160 cases, the F1 score, precision, and recall were calculated, and their mean, variance, maximum, and minimum values were summarized. The number of hyperplanes used to evaluate the complexity of the representation is also shown in the same way.

These results show that the proposed method demonstrates the highest point cloud classification performance while keeping the number of hyperplanes low owing to the merge, thereby achieving an accurate and simple separation boundary.

The following are our findings based on the analysis of experimental results for the comparison methods.

Single hyperrectangle method: Precision was low, and there was a tendency to cover a wide range of point clouds. As a result, the F1 score was not high.

Multiple hyperrectangle method: A high F1 score was obtained, although it was still lower than that of the proposed method. In addition, the separation boundary became much more complex due to the use of many hyperplanes, which indicates a tendency toward overfitting.

Greedy method: As expected, performance varied significantly, and overall stability was not achieved. In this experiment, the number of hyperplanes k was fixed at $k = 3$, but increasing k did not necessarily lead to improved accuracy. Detailed result for $k = 10$ is shown in the last row of Table VII.

For reference, we also attempted an approximation using simplicial complexes based on the persistent homology framework, separately from the comparison methods. This method is flexible in terms of shape, but it requires a very large number of simplices ($O(10^5)$) and the obtained F1 score was only around 0.3. Therefore, it is difficult to use this method as a practical comparison method in this study, and some improvement is needed to reduce the number of simplices.

4) Hyperparameter characteristics of the proposed method

Tables VIII–XI summarize the robustness of the proposed method's hyperparameters p, c, r, and m in three cases.

Asterisks (*) denote baseline values, and other results vary a single parameter while fixing the rest. On the evaluation machine, the execution became infeasible when the number of hyperplanes exceeded approximately 25, so some high-precision settings were terminated.

TABLE III
PREDICTION ACCURACY OF THE LEARNING MODEL IN EACH OBSERVATION SUITE

Observation name	Obs.1	Obs.2	Obs.3	Obs.4	Obs.5	Obs.6	Obs.7	Obs.8
Accuracy	0.932	0.939	0.941	0.928	0.934	0.931	0.903	0.935

TABLE IV
EVALUATION OF THE PROPOSED METHOD

	F1	precision	recall	complexity
Ave	0.96	0.98	0.92	7.2
Var	0.001	0.006	0.009	24.6
Max	1.0	1.0	1.0	19.0
Min	0.80	0.10	0.04	1.0

TABLE V
EVALUATION OF THE SINGLE HYPERRECTANGLE METHOD

	F1	precision	recall	complexity
Ave	0.82	0.77	0.87	12.0
Var	0.001	0.006	0.004	0.0
Max	0.92	0.90	0.99	12.0
Min	0.73	0.08	0.72	12.0

TABLE VI
EVALUATION OF THE MULTIPLE HYPERRECTANGLE METHOD

	F1	precision	recall	complexity
Ave	0.91	0.90	0.91	448.1
Var	0.002	0.0	0.0	3095.7
Max	0.98	0.98	0.99	588.0
Min	0.79	0.76	0.75	314.0

TABLE VII
EVALUATION OF GREEDY METHOD

	F1	precision	recall	complexity
Ave	0.82	0.73	0.96	3
Var	0.02	0.04	0.0	0
Max	0.99	0.97	1.0	3
Min	0.41	0.26	0.88	3
Ave	0.77	0.67	0.98	10

The characteristics of each hyperparameter are summarized below.

Mixed threshold *p*: At $p = 0.5$, some boundary blocks were excluded, increasing the number of hyperplanes. A smaller p is preferable.

Merge threshold *c*: Results were stable for $c \in [0.2,0.8]$, indicating robustness to this parameter.

Separation *r*: Merging remained stable without requiring very strict settings (e.g., $r \geq 0.95$).

Partition density *m*: Larger m improves resolution but exponentially increases cost (e.g., $m = 6 \rightarrow 6^6$ blocks). High F1 was already achieved with $m = 3$, suggesting moderate values are sufficient.

5) Scalability

To evaluate the performance of the proposed and comparison methods in high-dimensional spaces, experiments were conducted by increasing the number of observed items and expanding the QoS space to 10 and 16 dimensions. Table XII shows the F1 scores, number of hyperplanes, and execution times for two runs of each method.

The results show the proposed method maintains high discrimination performance (F1 score) even as dimension increases. On the other hand, as the dimension increases, the computation time increases significantly, and the proposed method tends to have a higher execution cost than the comparison methods. In particular, in 16 dimensions, the execution time per case exceeded 120 minutes, and it was judged to be unsuitable for real-time processing applications.

In contrast, the greedy method had shorter execution times than other methods, and the increase in computation time with increasing dimension was relatively gradual. This is thought to be because the greedy method does not perform dimension-specific partitioning or neighborhood determination, but instead performs low-dimensional operations such as sequential hyperplane addition.

This experiment shows that while the proposed method enables high-accuracy approximation, execution cost in high dimensions is an issue. In future studies, it will be necessary to investigate the possibility of speeding up the method by parallelizing the processing or combining it with dimension reduction methods.

V. FUTURE WORK

In this study, the performance evaluation was conducted under a fixed network topology while varying user dynamics. As future work, we plan to investigate how performance is affected when network topology itself changes. Such changes are expected to alter the geometric properties of the corresponding QoS point clouds. Our preliminary experiments using persistent homology indicated that geometric characteristics, such as cluster and hole structures, can influence the behavior of different approximation methods. A systematic analysis of these effects, particularly under diverse network topologies, remains an important direction for future research.

Furthermore, addressing the challenges of scalability and conducting evaluations with real-world data remain important avenues for future research.

TABLE VIII
PARAMETER P

	Case1		Case2		Case3	
p-value	F1	number of planes	F1	number of planes	F1	number of planes
0.02	0.98	2	0.94	17	0.95	7
0.08*	0.97	2	0.94	14	0.95	9
0.2	0.98	11	N/A	27	0.96	20
0.3	N/A	46	N/A	46	N/A	28

TABLE IX
PARAMETER C

	Case1		Case2		Case3	
c-value	F1	number of planes	F1	number of planes	F1	number of planes
0.2	0.98	2	0.94	10	0.95	10
0.7*	0.98	2	0.94	14	0.95	9
0.8	0.99	3	0.95	17	0.93	16
0.9	0.99	6	0.95	17	N/A	30

TABLE X
PARAMETER R

	Case1		Case2		Case3	
r-value	F1	number of planes	F1	number of planes	F1	number of planes
0.7	0.96	1	0.89	6	0.91	8
0.8	0.96	1	0.92	8	0.92	8
0.9*	0.98	2	0.94	14	0.95	9
0.95	0.98	4	N/A	65	0.92	8

TABLE XI
PARAMETER M

	Case1		Case2		Case3	
m-value	F1	number of planes	F1	number of planes	F1	number of planes
1	0.96	1	0.76	1	0.83	1
3	0.98	3	0.94	7	0.94	10
4*	0.98	2	0.94	14	0.95	9
6	0.99	2	N/A	35	N/A	28

TABLE XII
SCALABILITY

Method	dim=10			dim=16		
	F1	number of planes	time (min)	F1	number of planes	time (min)
Proposed (m =2)	0.94	4	2	0.95	2	120
Multiple	0.87	2278	1	0.90	3673	70
Greedy	0.86	10	0.2	0.88	10	7

VI. Conclusion

In this paper, we proposed a novel method to represent the set of measured QoS values satisfying the target QoE at end nodes as polytopes defined by multiple hyperplanes in a multi-node network measurement scenario. This concise representation enables effective control to achieve the target QoE, as well as reliable prediction of future QoE fulfillment or violation. Furthermore, by analyzing the volume and topological features of the polytopes, we can identify critical nodes and metrics for efficient QoE control and prediction.

References

[1] T. Hoßfeld, P. E. Heegaard, M. Varela, M. Sebastian, "QoE beyond the MOS: an in-depth look at QoE via better metrics and their relation to MOS," *Quality and User Experience*, vol. 1, no.1, pp. 1-23, 2016.

[2] H. Mao, R. Netravali, and M. Alizadeh. "Neural adaptive video streaming with Pensieve," *ACM Special Interest Group on Data Communication (SIGCOMM)*, pp. 197-210, 2017.

[3] J.S. Al-Azzeh, R. Odarchenko, A. Abakumova, S. Bondar, "Method for QOE monitoring and increasing in cellular networks based on QOE-to-QOS mapping using spline approximation," *EURASIP Journal on Wireless Communications and Networking,* vol. 2022, no.1, 2022.

[4] L. Yan, Z. Qin, R. Zhang, Y. Li and G. Y. Li, "QoE-Aware Resource Allocation for Semantic Communication Networks," *IEEE Global Communications Conference (GLOBECOM),* pp. 3272-3277, 2022.

[5] A. Bemporad, and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica,* vol. 35, no. 3, pp. 407-427, 1999.

[6] P. Tang, Y. Dong, Y. Chen, S. Mao and S. Halgamuge, "QoE-Aware Traffic Aggregation Using Preference Logic for Edge Intelligence," *IEEE Transactions on Wireless Communications*, vol. 20, no. 9, pp. 6093-6106, 2021.

[7] S. Balachandran, D. Dasgupta, F. Nino and D. Garrett, "A Framework for Evolving Multi-Shaped Detectors in Negative Selection," *IEEE Symposium on Foundations of Computational Intelligence (FOCI)*, pp. 401-408, 2007.

[8] K. B. Ajeyprasaath, P. Vetrivelan, "Machine Learning Based Classifiers for QoE Prediction Framework in Video Streaming over 5G Wireless Networks," *Computers, Materials and Continua*, vol. 75, no. 1, pp. 1919-1939, 2023.

[9] A. Ahmad, A. B. Mansoor, A. A. Barakabitze, A. Hines, L. Atzori and R. Walshe, "Supervised-learning-Based QoE Prediction of Video Streaming in Future Networks: A Tutorial with Comparative Study," *IEEE Communications Magazine*, vol. 59, no. 11, pp. 88-94, 2021.

[10] J. Zhang, et al. "Bridging the gap between QoE and QoS in congestion control: A large-scale mobile web service perspective," *USENIX Annual Technical Conference (USENIX ATC)*, pp. 553-569, 2023.

[11] G. Kougioumtzidis, V. K. Poulkov, P. I. Lazaridis and Z. D. Zaharis, "Deep Reinforcement Learning-Based Resource Allocation for QoE Enhancement in Wireless VR Communications," *IEEE Access*, vol. 13, pp. 25045-25058, 2025.

[12] L. Yan, Z. Qin, C. Li, R. Zhang, Y. Li and X. Tao, "QoE-Based Semantic-Aware Resource Allocation for Multi-Task Networks," *IEEE Transactions on Wireless Communications*, vol. 23, no. 9, pp. 11958-11971, 2024.