Gaussian Process Regression-Based Traffic Prediction and Rate Coordination for Service Chain Congestion Mitigation

Zhi Wang, Bo Yi, *Member, IEEE*, Qiang He, XingWei Wang, YingPu Nian, XiaoShi Song, *Member, IEEE*, and Keqin Li, *Fellow, IEEE*

***Abstract*—With the rapid development of Internet technologies, the network traffic continues to grow exponentially, which leads to serious congestion and makes the service quality difficult to guarantee. Network function virtualization (NFV) has been proposed to ease such a challenge by sharing resource among different virtual network function (VNF). However, such resource sharing among VNFs would worsen traffic congestion and the waste of work due to the processing rates mismatch between upstream and downstream VNFs of the same service chain. Therefore, this work focuses on coordinating the VNF processing rates to jointly avoid resource waste and ease traffic congestion. Specifically, the rate balance process is implemented with two parts which are the rate prediction and matching, during which the backpressure and Gaussian process regression are applied to quickly locate the rate mismatch position and formulate the relationship between network resource and traffic rate respectively. Then, the future traffic rate is predicted and applied to adjust the allocated resource of VNFs in one service chain for mismatch rate coordination. The experimental results indicate that the proposed method eases the traffic congestion and resource waste efficiently and outperforms the other methods in terms of delay, backlog, throughput, etc.**

***Index Terms*—Backpressure, Gaussian process regression, rate coordination, traffic scheduling, virtualization of network functions.**

NOMENCLATURE

Variable	*Description*
ST_s	Start time of service chain s.
ET_s	Maximum allowed end time of service chain s.
κ_s	Number of packets in service chain s during $[\mathrm{ST}_s, \mathrm{ET}_s]$.
$\lambda_{s,p}$	Size of the pth packet in service chain s.
$\eta_{i,p}$	Completion time of the pth packet at VNF v_i.
c_i^v	Processing capacity of VNF v_i.
$b_{i,j}^l$	Link capacity between VNF instances v_i and v_j.
$x_i^{i'}$	Whether VNF v_i is deployed on node $n_{i'}$.
$y_{i,j}^{i',j'}$	Whether virtual link $l_{i,j}$ is mapped on physical link $e_{i',j'}$.
$z_p^{s,i}$	Whether VNF v_i in service chain s can process the pth packet.
$\delta_{v_i}^{v_{i-1}}$	Backpressure between adjacent VNFs v_{i-1} and v_i.
ξ_1^s	Bandwidth requirement of service chain s.
ξ_2^s	Traffic label of service chain s.
ξ_3^s	Average packet size of service chain s.
$\boldsymbol{\xi_4^s}$	VNF traffic rates for service chain s.
$\boldsymbol{\xi_5^s}$	VNF congestion condition for service chain s.
$\boldsymbol{\xi_6^s}$	VNF resource features for service chain s.

I. INTRODUCTION

THE Internet has entered a new phase of evolution with the emergence of novel network technologies, including B5G/6G [1], the Internet of Things [2], [3], [4], and blockchain [5]. These advancements offer significant benefits—for instance, B5G/6G has greatly reduced end-to-end delay and significantly enhanced bandwidth. Consequently, numerous new applications have emerged, such as virtual reality, augmented reality, and driverless cars. These technological improvements and applications collectively drive Internet development and enhance user experience [6].

However, the rapid adoption of these technologies has led to a massive influx of end terminals connecting to the Internet, with their numbers continuing to grow exponentially. This surge presents two major challenges for network infrastructure operation and maintenance: increased network congestion due to high traffic volumes and the difficulty of accommodating diverse user requirements. To mitigate these issues, network function virtualization (NFV) has been introduced as a solution, enabling more efficient resource utilization and congestion alleviation by transforming hardware-based network functions into software-based virtual network functions (VNFs) [7], [8].

Received 24 June 2024; revised 23 March 2025 and 6 August 2025; accepted 10 September 2025. Date of publication 20 October 2025; date of current version 14 January 2026. This work was supported in part by the National Natural Science Foundation of China under Grant 62472078, Grant U22A2004, and Grant 62032013, in part by the Fundamental Research Funds for the Central Universities of China under Grant N2316006 and Grant N25LPY047, in part by the Liaoning Provincial Science and Technology Joint Program under Grant 2023-MSBA-079, in part by the Open Research Fund from Guangdong Laboratory of Artificial Intelligence and Digital Economy under Grant GML-KF-24-31, in part by the State Grid Corporation of China Science and Technology Project Funding under Grant 2024YF-95, and in part by the ZTE Industry-University Cooperation Funds under Grant IA20240819011. *(Corresponding author: Bo Yi.)*

Zhi Wang, Bo Yi, XingWei Wang, YingPu Nian, and XiaoShi Song are with the College of Computer Science and Engineering, Northeastern University, Shenyang 110169, China (e-mail: wz13368275969@gmail.com; yiboboscar@gmail.com; wangxingweis@163.com; 1134697238@qq.com; songxiaoshi@cse.neu.edu.cn).

Qiang He is with the College of Medicine and Biological Information Engineering, Northeastern University, Shenyang 110169, China (e-mail: heqiangcai@gmail.com).

Keqin Li is with the Department of Computer Science, State University of New York, New York, NY 12561 USA (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/JSYST.2025.3610504

A VNF is a software instance that operates within a virtualized network to provide various network functions, such as routing, firewalling, load balancing, and network address translation. Service function chains (SFCs) are composed of VNFs and dynamically configured to meet diverse user requirements [9]. The virtualized nature of VNFs enables flexible and rapid adjustments based on network conditions, allowing SFCs to be reconfigured efficiently. Despite this, we deeply analyze two challenges that: 1) The mismatch in processing rates between upstream and downstream VNFs in SFC leads to significant resource wastage. When the upstream VNF has a much smaller processing rate than the downstream VNF, the latter's resources are underutilized. Conversely, if the upstream VNF has a higher processing rate than the downstream VNF, it will lead to congestion at the downstream VNF, causing packet loss. This issue is particularly prevalent in dynamic environments, where traffic patterns can fluctuate rapidly, making such mismatches occur more frequently and affecting the overall system performance. 2) Furthermore, sharing VNF instances among different services exacerbates downstream VNF congestion [10].

Taking the above analysis into consideration, this work proposes the Gaussian process regression (GPR)-based rate coordination mechanism to address the corresponding VNF rate mismatch problem. The mechanism operates in two stages: *1) Rate prediction based on the formulated relationship between processing rate and resource allocation. 2) Rate coordination between upstream and downstream VNFs.* First, achieving accurate processing rate prediction requires the prediction model to strike a balance between local and global generalization abilities. To achieve this equilibrium, we propose integrating the Matern and periodic covariance functions as the foundation for GPR [11]. Second, it is essential to recognize that rate coordination revolves around adjusting the resource allocation for different VNFs residing on the same node. In accordance with the prediction results, the resource allocation for each VNF is dynamically adjusted to meet the demands of rate coordination, considering both local and global perspectives. Lastly, once rate coordination is implemented, a globally balanced state is achieved. This state significantly alleviates resource wastage and greatly enhances the efficiency of service provisioning.

The main contributions are as follows.

1) We propose a VNF rate coordination algorithm based on GPR and optimization. The algorithm employs the GPR model to predict and discover the inherent relationship between VNF processing rates and resources, thereby enabling accurate rate coordination decisions based on these predictions.
2) We optimize the hyperparameters by utilizing sparse computation to streamline the matrix inversion process, resulting in a reduction of time complexity from $O(n_h^3)$ to $O(n_s^2 n_h)$ (n_h is the number of historical data samples and $n_s (\ll n_h)$ is the number of the sparse data samples).
3) The experimental results show that the proposed rate coordination algorithm achieves better performance in terms of the average completion time, average delay, packet loss rate, throughput and backlog compared with the other state-of-the-art methods.

The rest of this article is organized as follows. Section II introduces the related work. Section III designs the system framework and model. Section IV mainly discusses the proposed mechanism. Section V analyzes the experimental results. Finally, Section VI concludes this article.

II. Related Work

VNF mapping involves deploying VNF software onto standardized hardware based on user requirements. These VNFs are then arranged sequentially to form the SFC, ensuring complete and reliable service delivery. Traditional service provisioning relies on static network function mapping, which requires predeployment before use. To overcome this limitation, various approaches have been proposed, including single-hop VNF scheduling and mapping [9], mapping VNFs along the shortest path [10], online mapping and scheduling strategies [11], and dynamic priority adjustment methods [12]. However, these approaches overlook abrupt traffic fluctuations and the stochastic nature of user demands, which may lead to network congestion. To address this issue, both static and dynamic VNF mapping techniques have been introduced [13]. While static VNF mapping lacks the flexibility to adapt to real-time network changes, dynamic VNF mapping provides better adaptability. However, this flexibility comes at the cost of higher resource consumption and potential queuing issues.

To balance cost and performance, static and dynamic mapping are integrated. For instance, Zhu et al. [13] proposed a hybrid approach that initially deploys VNFs statically and later adjusts placements dynamically, achieving a tradeoff between cost and performance. Similarly, Yao et al. [14] tackled the mapping and scheduling problem by formulating an integer linear programming (ILP) model to minimize cost and delay, employing Tabu search for an optimal solution, and subsequently introducing an online preemption algorithm to dynamically assign VNFs to appropriate physical nodes. However, this strategy remains complex and inherits limitations from both static and dynamic approaches.

To maximize resource utilization, Yi et al. [15] proposed sharing VNFs across service chains to mitigate resource fragmentation from deploying numerous instances. However, limited queue lengths may cause packet queuing, loss, and degraded performance [16]. Efficient resource allocation is, thus, essential; Herrera et al. [17] introduced a three-stage method employing shunt operations to optimize allocation and minimize delay, enhancing flow rates, and supporting multiple traffic paths while preserving VNF dependencies [18]. A major challenge in VNF sharing is network congestion when multiple services share one instance. To tackle this, Elias et al. [19] analyzed VNF sharing in NFV infrastructures to establish congestion performance bounds. Similarly, Zeng et al. [20] proposed a fair VNF scheduling method using a bias-parameterized utility function and an online distributed scheduling algorithm based on a discrete time-slot queuing model to address real-time traffic requirements.

Service chain provisioning in mobile edge computing has been widely studied. For example, Xie et al. [21] employed a deep learning approach for dynamic VNF scheduling, yet such methods face high complexity and real-time limitations. To address congestion from VNF sharing, Yang et al. [22] developed a dynamic resource allocation framework using online improved slope methods. Similarly, Chen et al. [23] proposed a service chain model with two improved slope algorithms to minimize delays, though these methods degrade under resource constraints and complex VNF multiplexing.

Overall, existing works fail to achieve satisfactory performance due to their neglect of VNF rate mismatches, leading to resource underutilization or network congestion, especially in dynamic environments. In contrast, this work leverages GPR-based rate prediction and backpressure-driven dynamic coordination to effectively balance loads and mitigate congestion, while employing sparse matrix optimization to reduce computational overhead, thereby significantly enhancing service chain performance.

III. Problem Model

The underlying physical network is modeled as an undirected connected graph $G = (N, E)$, where $N = \{n_i \mid i \in [1, |N|]\}$ represents the set of physical nodes, and $E = \{e_{i,j} \mid i, j \in [1, |E|]\}$ represents the set of physical links. Each physical node n_i has a processing capacity c_i^n, while each physical link $e_{i,j}$ has a bandwidth resource denoted by $b_{i,j}^e$. In addition, the VNF-composed forwarding graph is represented as $G' = (V, L)$, where $V = \{v_i \mid i \in [1, |V|]\}$ denotes the set of VNFs, and $L = \{l_{i,j} \mid i, j \in [1, |L|]\}$ represents the set of virtual links. Given a service set S and a VNF category set F, each service chain $s \in S$ requires a subset of VNFs $F_s \subseteq F$. The bandwidth requirement for a service chain is denoted by $b_{i,j}^l$, while the node capacity requirement for the ith required VNF instance ($\in F_s$) is represented as c_i^v.

Nomenclatue summarizes the variables used in our model. In brief, a service chain s starts at time ST_s and must be completed by ET_s. During the interval $[\mathrm{ST}_s, \mathrm{ET}_s]$, κ_s packets are transmitted, each with a size of $\lambda_{s,p}$. The completion time of the p-th packet processed by VNF v_i is denoted as $\eta_{i,p}$. Packet processing and propagation times are given by $\frac{\lambda_{s,p}}{c_i^v}$ and $\frac{\lambda_{s,p}}{b_{i,j}^l}$, respectively. Detailed definitions of these and other variables are provided in Nomenclatue.

Now, we need to build the relationship between service chain and the physical network. $\forall v_i \in V, n_{i'} \in N$, a decision variable is defined as

$$x_i^{i'} = \begin{cases} 1, & \text{if } v_i \text{ is deployed on } n_{i'} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Similarly, the virtual link is also mapped on several physical links. $\forall l_{i,j} \in L, e_{i',j'} \in E$, we define another decision variable as

$$y_{i,j}^{i',j'} = \begin{cases} 1, & \text{if } l_{i,j} \text{ is mapped on } e_{i',j'} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

As for the p-th packet of s, a binary variable is also defined as

$$z_p^{s,i} = \begin{cases} 1, & \text{if } v_i \in F_s \text{ can process the } p\text{-th packet} \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Considering the aforementioned definition, it is crucial to account for the service chain constraint, which mandates that data packets traverse the deployed VNFs in a predefined order. Based on this constraint, the overall flow completion time is computed as shown in (4), incorporating both VNF processing delay and link transmission delay. The former depends on the computational capacity of VNFs and traffic allocation strategies, while the latter is determined by link bandwidth and routing decisions. By jointly optimizing these factors, the objective function aims to enhance VNF processing efficiency and overall resource utilization

$$T = \sum_{s \in S} \left(\sum_{i=1}^{|V|} \sum_{i'=1}^{|N|} \sum_{p \in \kappa_s} \frac{\lambda_{s,p}}{c_i^v} x_i^{i'} z_p^{s,i} + \sum_{i,j=1}^{|V|} \sum_{i',j'=1}^{|N|} \sum_{p \in \kappa_s} \frac{\lambda_{s,p}}{b_{i,j}^l} y_{i,j}^{i',j'} z_p^{s,i} \right). \tag{4}$$

Combined with (4), we formulate the overall objective as follows:

$$\begin{aligned}
&\text{Minimize:} \quad T \\
&\text{s.t.} \quad \sum_{i' \in [1,|N|]} x_i^{i'} = 1 \quad \forall i \in [1, |V|] \\
&\sum_{i',j' \in [1,|E|]} y_{i,j}^{i',j'} \geq 1 \quad \forall i, j \in [1, |V|] \\
&\sum_{i \in [1,|V|]} x_i^{i'} c_i^v \leq c_{i'}^n \quad \forall i' \in [1, |N|] \\
&\sum_{i,j \in [1,|L|]} y_{i,j}^{i',j'} b_{i,j}^l \leq b_{i',j'}^e \quad \forall i', j' \in [1, |N|] \\
&\sum_{i=1}^{|V|} \sum_{i'=1}^{|N|} \sum_{p \in \kappa_s} \frac{\lambda_{s,p}}{c_i^v} x_i^{i'} z_p^{s,i} \\
&+ \sum_{i,j=1}^{|V|} \sum_{i',j'=1}^{|N|} \sum_{p \in \kappa_s} \frac{\lambda_{s,p}}{b_{i,j}^l} y_{i,j}^{i',j'} z_p^{s,i} \leq (\mathrm{ET}_s - \mathrm{ST}_s) \\
&\eta_{i,p} - \eta_{i,p-1} - \sum_{p=1}^{\kappa} z_p^{s,i} \frac{\lambda_{s,p}}{c_i^v} \geq 0 \quad \forall i \in [1, |F_s|].
\end{aligned} \tag{5}$$

This optimization problem is NP-hard, which is proved in the supplementary materials. Our goal is to find an approximate solution that tackles traffic congestion caused by VNF rate mismatches in service chains while minimizing flow completion time. By reducing flow completion time, we can accommodate more traffic flow within the same time constraints, thereby enhancing overall network efficiency. The flow completion time, denoted as T, is determined by several constraints. The first ensures each VNF instance is deployed on a single physical node, while the second ensures each virtual link is mapped to at least one physical link. In addition, the third and fourth constraints ensure that allocated resources do not exceed physical capacity. The fifth constraint ensures that the total flow completion time stays within the maximum allowable time window, denoted as $\mathrm{ET}_s - \mathrm{ST}_s$. This helps to guarantee that the system meets real-time requirements. The last constraint ensures that packets within a given flow are processed in sequence, as they pass through the VNFs. To address this optimization problem as shown in (5), we employ a GPR-based rate prediction and coordination mechanism to optimize resource allocation, thereby minimizing end-to-end processing latency.

IV. GPR-Based Traffic Prediction and Rate Coordination Mechanism

The above section formulates an ILP-based objective function to optimize VNF resource allocation and minimize SFC end-to-end latency. However, the NP-hard nature of this optimization problem poses significant challenges for large-scale solutions. To address this issue and enhance adaptability in dynamic networks, we propose a GPR-based rate prediction and coordination mechanism. The dynamic rate prediction method anticipates service congestion caused by processing rate imbalances by establishing relationships between processing rate and VNF resources. Leveraging this information, it proactively identifies congestion risks. Based on these predictions, we design a rate coordination algorithm that dynamically adjusts VNF resource allocation to achieve rate and load balancing. When congestion is anticipated, the system triggers reoptimization to ensure real-time adaptation of the resource allocation strategy.

A. GPR-Based Traffic Prediction

In this section, we present the GPR-based traffic prediction framework that supports proactive rate coordination. The process includes the following four key stages:

1) establishing a prediction model that captures the relationship between VNF processing rates and resource allocation;
2) selecting a kernel function that reflects the nonlinear and periodic nature of network traffic;
3) optimizing hyperparameters to improve accuracy and efficiency;
4) forecasting future traffic using historical and real-time data.

1) Prediction Model Establishment: GPR requires a substantial amount of historical data collected from ongoing services. However, using all service chain features can lead to high training costs and convergence issues. To address this, we select six representative features: data flow tag, bandwidth, packet size, data rate, VNF congestion status, and VNF resource profile. These features capture essential aspects of traffic dynamics and VNF performance, enabling efficient and accurate congestion prediction for service chain flows. Given a service chain s, we have

$$\begin{aligned}\boldsymbol{\xi^s} &= \{\xi_1^s, \xi_2^s, \xi_3^s, \boldsymbol{\xi_4^s}, \boldsymbol{\xi_5^s}, \boldsymbol{\xi_6^s}\}\\ \text{where } \xi_1^s &\in \mathbf{R}_+\\ \xi_2^s &\in \mathbf{N}_+\\ \xi_3^s &\in \mathbf{R}_+\\ \boldsymbol{\xi_4^s} &= \{r_i^v| \quad \forall v_i \in F_s\}\\ \boldsymbol{\xi_5^s} &= \left\{\frac{\delta_{v_i}^{v_{i-1}}}{q_{i-1}^v}| \quad \forall v_i, v_{i-1} \in F_s\right\}\\ \boldsymbol{\xi_6^s} &= \{c_i^v| \quad \forall v_i \in F_s\}.\end{aligned} \tag{6}$$

In (6), ξ_1^s indicates the bandwidth resource requirement of s; ξ_2^s indicates the traffic label of s; ξ_3^s indicates the average packet size of s; $\boldsymbol{\xi_4^s}$ indicates the VNF traffic rates of s; $\boldsymbol{\xi_5^s}$ indicates the VNF congestion condition of s; $\boldsymbol{\xi_6^s}$ indicates the VNF resource feature of s, encompassing factors such as CPU, memory, and storage requirements.

For each feature dimension, we first formulate the definition of its probability density function, as follows.

Given any one of them, we have the following.

Definition 1: Given any dimension of the feature set of s, let us say $\xi_i^s, \forall i \in [1, n]$, its probability density function is formulated according to the univariate Gaussian distribution, as follows:

$$f(\xi_i^s) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(\xi_i^s - \mu)^2}{2\sigma^2}\right) \tag{7}$$

where μ and σ are the mean and variance of the feature ξ_i^s; $n = 6$ is the number of feature dimensions.

However, there are six kinds of features of s. Apparently, these features are independent of each other. In this way, (8) is expanded and generalized as follows:

$$\begin{aligned}f(\xi_1^s, \ldots, \xi_n^s) &= \prod_{i=1}^{n} f(\xi_i^s)\\ &= \frac{\exp\left(-\frac{1}{2}\left[\frac{(\xi_1^s-\mu_1)^2}{\sigma_1^2} + \cdots + \frac{(\xi_n^s-\mu_1)^2}{\sigma_n^2}\right]\right)}{(2\pi)^{\frac{n}{2}}\sigma_1 \cdots \sigma_n}\end{aligned} \tag{8}$$

where $\mu_1, \ldots, \mu_n$ and $\sigma_1, \ldots, \sigma_n$ are the mean and variance of different dimensional feature samples, respectively.

Let

$$\begin{aligned}\boldsymbol{\xi^s} - \boldsymbol{\mu} &= [\xi_i^s - \mu_i]_{(1\times n)}^T \quad \forall i \in [1, n]\\ K &= \begin{bmatrix}\sigma_1^2 & 0 & \cdots & 0\\ 0 & \sigma_2^2 & \cdots & 0\\ \vdots & \vdots & \ddots & 0\\ 0 & 0 & 0 & \sigma_n^2\end{bmatrix}\end{aligned} \tag{9}$$

we deduce that

$$\begin{aligned}\sigma_1\sigma_2 \cdots \sigma_n &= |K|^{\frac{1}{2}}\\ \sum_{i=1}^{n} \frac{(\xi_i^s - \mu_i)^2}{\sigma_i^2} &= (\boldsymbol{\xi^s} - \boldsymbol{\mu})^T K^{-1} (\boldsymbol{\xi^s} - \boldsymbol{\mu}).\end{aligned} \tag{10}$$

Substituting (10) into (8), it follows that

$$\begin{aligned}f(\boldsymbol{\xi^s}) &= \frac{\exp\left(-\frac{1}{2}(\boldsymbol{\xi^s} - \boldsymbol{\mu})^T K^{-1} (\boldsymbol{\xi^s} - \boldsymbol{\mu})\right)}{(2\pi)^{\frac{n}{2}}|K|^{\frac{1}{2}}}\\ &= (2\pi)^{-3}|K|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{\xi^s} - \boldsymbol{\mu})^T K^{-1} (\boldsymbol{\xi^s} - \boldsymbol{\mu})\right)\end{aligned} \tag{11}$$

where $\boldsymbol{\mu} \in \mathbf{R}^n$ is the mean vector; $K \in \mathbf{R}^{n\times n}$ is the covariance function matrix.

2) Kernel Function Design: The covariance function matrix $K \in \mathbf{R}^{n\times n}$ plays a crucial role in GPR, as it defines the kernel function in GPR. Different kernel functions can significantly impact the performance of the prediction model. Therefore, selecting an appropriate kernel function is essential for the established prediction model.

A commonly used kernel function in GPR is the radial basis function (RBF) [24], known for its flexibility and smoothness in fitting nonlinear data with strong local descriptive capability. Given these advantages, we adopt the RBF as the fundamental kernel function for the proposed prediction model, formulated as follows:

$$K(\xi_i^s, \xi_j^s) = \sigma^2 \exp\left(-\frac{(\xi_i^s - \xi_j^s)^2}{2l^2}\right). \tag{12}$$

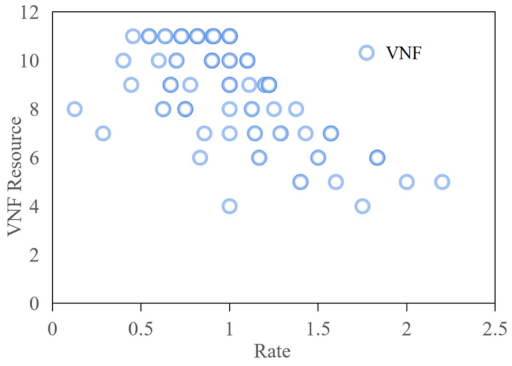


Fig. 1. Example of relationship between VNF resource and rate.

The Euclidean distance calculation of $K(\xi_i^s, \xi_j^s)$ represents the difference between adjacent VNFs, where greater distances indicate larger disparities in processing capacity, increasing the likelihood of congestion. The parameter l determines the feature length. However, the strong local descriptive ability of the RBF kernel makes it prone to local optima, leading to overfitting. To address this, the kernel function should also incorporate global generalization capabilities. Moreover, in service chain provisioning, the resource allocation and processing rates of a VNF instance in s are closely related to those of its upstream and downstream VNFs. For instance, Fig. 1 illustrates a simple case demonstrating this relationship. Notably, the resource allocation of different VNF instances does not follow a linear pattern when the rate is fixed but instead exhibits a periodic trend.

Taking the above two conditions into consideration, the periodic covariance function is also introduced and combined in the kernel function of (12), so that the kernel function is transformed as

$$\begin{aligned} K(\xi_i^s, \xi_j^s) &= \sigma^2 \exp\left(-\frac{(\xi_i^s - \xi_j^s)^2}{2l^2}\right) \\ &\quad + \sigma^2 \exp\left(-\frac{2\sin^2(\frac{\pi(\xi_i^s - \xi_j^s)^2}{o})}{l^2}\right) \\ &= \sigma^2 \exp\left(-\frac{(\xi_i^s - \xi_j^s)^2}{2l^2} - \frac{2\sin^2(\frac{\pi(\xi_i^s - \xi_j^s)^2}{o})}{l^2}\right) \\ &= \sigma^2 \exp\left(-\frac{(\xi_i^s - \xi_j^s)^2 + 4\sin^2(\frac{\pi(\xi_i^s - \xi_j^s)^2}{o})}{2l^2}\right) \end{aligned} \tag{13}$$

where o indicates the cycle.

Based on the above definitions, we obtain the prior distribution of $f(\boldsymbol{\xi^s})$ as follows:

$$p(f \mid \boldsymbol{\xi^s}, \theta) = \mathcal{N}(\mu, K) \tag{14}$$

where θ is the hyperparameter set including σ, l etc.

3) Hyperparameter Optimization: Generally, the prediction process of GPR suffers from a high training complexity (i.e., $O(n^3)$), which may delay the service chain provision process and even cause severe service congestion. In this regard, we need to reduce the training complexity. Based on the prior distribution function in (14), we begin the optimization from the noisy observations (i.e., chaos data samples) denoted by $\gamma = f(\boldsymbol{\xi^s}) + \epsilon$. Assuming that the noise ϵ satisfies the same independent Gaussian distribution, that is, $p(\epsilon) = \mathcal{N}(\mu, \sigma^2 I)$, then, we obtain a likelihood function as follows:

$$p(\gamma \mid f) = \mathcal{N}\left(\mu, \sigma^2 I\right) \tag{15}$$

where I represents the unit matrix.

Combining (14) and (15), we obtain the marginal likelihood distribution as follows:

$$\begin{aligned} p(\gamma \mid \boldsymbol{\xi^s}, \theta) &= \int p(\gamma \mid f) p(f \mid \boldsymbol{\xi^s}, \theta) df \\ &= \mathcal{N}\left(\mu, K + \sigma^2 I\right). \end{aligned} \tag{16}$$

According to the Gaussian process, the joint probability distribution function is formulated as follows:

$$\begin{aligned} &\begin{pmatrix} \gamma \\ f_* \end{pmatrix} \sim \mathcal{N}\left(\mu, \begin{matrix} K + \sigma^2 I & K_* \\ K_*^T & K_{**} \end{matrix}\right) \\ &\text{where } K_*^T = \left\{K\left(\xi_1^s, \xi_*^s\right), \ldots, K\left(\xi_{|F_s|}^s, \xi_*^s\right)\right\} \\ &K_{**} = K\left(\xi_*^s, \xi_*^s\right). \end{aligned} \tag{17}$$

In (17), ξ_*^s is the prediction input and f_* is the prediction output. Transforming the joint probability distribution of (18) to the marginal distribution, it follows that

$$\begin{aligned} p\left(f_* \mid \boldsymbol{\xi^s}, \gamma, \xi_*^s, \sigma^2\right) &= \mathcal{N}\left(f_* \mid \overline{f_*}, \operatorname{cov}\left(f_*\right)\right) \\ \bar{f}_* &= K_*^T \left[K + \sigma^2 I\right]^{-1} \gamma \\ \operatorname{cov}\left(f_*\right) &= K_{**} - K_*^T \left[K + \sigma^2 I\right]^{-1} K_* \end{aligned} \tag{18}$$

where $\bar{f}_*$ represents the average value of f_* and $\operatorname{cov}(f_*)$ represents the predicted covariance.

Jointly observing (17) and (18), we discover that $(K + \sigma^2 I)$ is the main reason leading to such high complexity in GPR, since the matrix inversion is extremely time-consuming. Therefore, to improve the calculation speed and reduce the impact on prediction, a sparse method is adopted to reduce the rank of the kernel matrix [25]. Assuming that K is an $n \times n$ matrix and $K = K' K'^T$, where K' is an $n \times m (n > m)$ matrix, then, according to the inversion theorem of matrix [26], we have

$$\left(K + \sigma^2 I\right)^{-1} = \sigma^{-2} I - \sigma^{-2} K' \left(\sigma^2 I + K'^T K'\right)^{-1} K'^T. \tag{19}$$

Based on (19), the inversion operation of the $n \times n$ matrix (i.e., $K + \sigma^2 I$) becomes the inversion operation of the $m \times m$ matrix (i.e., $(K'^T K')$), where $m < n$, so that we get the following theorem.

Theorem 1: Given the kernel matrix K, there exists a matrix $\widetilde{K}$ of rank m satisfying

$$\widetilde{K} = K_{n,m} K_{m,m}^{-1} K_{m,n} \tag{20}$$

where $(K_{m,m})_{i,j} = K(\xi_i^s, \xi_j^s), \forall i, j \in [1, m]$ is invertible.

Proof: Let X be an $n \times n$ matrix and $K = XX^T, C = X^T X$, applying the singular-value decomposition to the three matrices, so that we have

$$\begin{aligned} X &= U \Sigma^{\frac{1}{2}} V^T \\ K &= U \Sigma U^T \\ C &= V \Sigma V^T \end{aligned} \tag{21}$$

where U is the unitary matrix; Σ is the positive semidefinite diagonal matrix; V^T is the conjugate transpose matrix.

Similarly, let X_m be the matrix consisting of the first m rows of X, so that $K_m = X_m X_m^T, C = X_m^T X_m$. Applying the singular-value decomposition to them, we have

$$\begin{aligned} X_m &= U_m \Sigma_m^{\frac{1}{2}} V_m^T \\ K_m &= U_m \Sigma_m U_m^T \\ C_m &= V_m \Sigma_m V_m^T. \end{aligned} \tag{22}$$

It is aware that U, V, U_m, V_m are orthogonal matrices, it follows that

$$\begin{aligned} U &= X V \Sigma^{-\frac{1}{2}} \\ V_m &= X_m^T U_m \Sigma_m^{-\frac{1}{2}}. \end{aligned} \tag{23}$$

Replacing V, Σ by V_m, Σ_m, an approximation for U (i.e., $\widetilde{U}$) is obtained as follows:

$$\widetilde{U} = X X_m^T U_m \Sigma_m^{-1}. \tag{24}$$

Now, defining the approximation for K by $\widetilde{K} = \widetilde{U} \Sigma_m \widetilde{U}^T$, it follows that

$$\begin{aligned} \widetilde{K} &= \widetilde{U} \Sigma_m \widetilde{U}^T \\ &= X X_m^T U_m \Sigma_m^{-1} \Sigma_m \left(X X_m^T U_m \Sigma_m^{-1} \right)^T \\ &= X X_m^T \left\{ U_m (\Sigma_m^{-1})^T U_m^T \right\} (X X_m^T)^T. \end{aligned} \tag{25}$$

Then, according to the features of the orthogonal matrix, the equation $(U_m)^T = (U_m)^{-1}$ holds. In addition, the inverse of matrix multiplication follows the principle that $(AB)^{-1} = B^{-1} A^{-1}$, where A, B are matrices. Taking the above two situations into consideration, the expression in (25) is transformed as follows:

$$\begin{aligned} U_m \left(\Sigma_m^{-1} \right)^T U_m^T &= \left(U_m \Sigma_m^T U_m^T \right)^{-1} \\ &= (K)^{-1}. \end{aligned} \tag{26}$$

Substituting (26) into (25), we have

$$\widetilde{K} = \left(X X_m^T \right) (K)^{-1} \left(X X_m^T \right)^T. \tag{27}$$

Let $K_{n,m} = X X_m^T$, then this theorem is proved.

Based on Theorem 1, we quickly calculate the approximate kernel matrix $\widetilde{K}$ by transforming the high-rank matrix inversion into low-rank matrix inversion. Replacing K in (18) with $\widetilde{K}$, it follows that

$$\begin{aligned} \bar{f}_* &= K_*^T \left[\widetilde{K} + \sigma^2 I \right]^{-1} \gamma \\ \operatorname{cov}(f_*) &= K_{**} - K_*^T \left[\widetilde{K} + \sigma^2 I \right]^{-1} K_* \end{aligned} \tag{28}$$

where K_*^T and K_{**} are calculated according to (17). The use of $\widetilde{K}$ reflects the core principle of sparse GPR, where a low-rank approximation is employed to reduce the cubic complexity of full kernel inversion. Although this introduces a slight deviation from the globally optimal solution, it enables a favorable balance between inference accuracy and computational efficiency, which is essential for real-time large-scale traffic prediction.

The posterior distribution of the corresponding hyperparameter is calculated according to the Bayesian theory [27], as follows:

$$p(\theta \mid \gamma, \boldsymbol{\xi^s}) = \frac{p(\gamma \mid \theta, \boldsymbol{\xi^s}) p(\theta \mid \boldsymbol{\xi^s})}{p(\gamma \mid \boldsymbol{\xi^s})}. \tag{29}$$

Observing (29), we discover that

1) the probability value of $p(\theta \mid \boldsymbol{\xi^s}) = p(\theta)$ does not have correlation with the data samples, so that it is usually set to a constant;
2) $p(\gamma \mid \boldsymbol{\xi^s})$ is a marginal likelihood function that has no relation with θ, so that it is also set to a constant;
3) $p(\gamma \mid \theta, \boldsymbol{\xi^s})$ is the likelihood function of observations.

Then, according to the case that $p(\gamma \mid \theta, \boldsymbol{\xi^s})) = \mathcal{N}(\gamma \mid \mu, K + \sigma^2 I)$, we conclude that $p(\theta \mid \gamma, \boldsymbol{\xi^s})$ is proportional to $\mathcal{N}(\gamma \mid \mu, K + \sigma^2 I)$, which follows the Gaussian process that

$$p(\theta \mid \gamma, \boldsymbol{\xi^s}) \propto \frac{1}{\sqrt{|2\pi(K + \sigma^2 I)|}} \exp\left(-\frac{1}{2} (\boldsymbol{\xi^s} - \mu) \right). \tag{30}$$

Then, the optimal hyperparameters are determined by finding the maximum value of (30). In this way, the logarithm operation is executed on both sides of (30) to achieve the logarithmic likelihood function, as follows:

$$\begin{aligned} \log(p) = \frac{2}{n} \log(2\pi) - \frac{1}{2} \log \left| \widetilde{K} + \sigma^2 I \right| \\ - \frac{1}{2} (\gamma - \mu)^T \left(\widetilde{K} + \sigma^2 I \right)^{-1} (\gamma - \mu) + \log(c) \end{aligned} \tag{31}$$

where n is the dimension of K; $\frac{2}{n} \log(2\pi)$ and $\log(c)$ are constants; $\frac{1}{2} \log |\widetilde{K} + \sigma^2 I|$ is the complexity penalty; $\frac{1}{2} (\gamma - \mu)^T (\widetilde{K} + \sigma^2 I)^{-1} (\gamma - \mu)$ is the fitting degree of hyperparameters to samples.

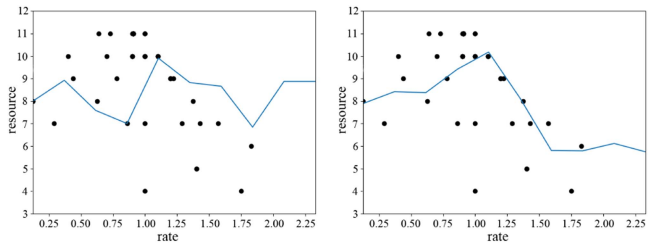
Next, assuming that the hyperparameter set is $\theta = \{\theta_i | i \in [1, n]\}$, we use the conjugate gradient method to calculate the partial derivative for any θ_i, so as to obtain the optimal hyperparameters, as follows:

$$\begin{aligned} \frac{\partial \log(p)}{\partial \theta_i} &= -\frac{1}{2} \operatorname{tr}\left(P^{-1} \frac{\partial P}{\partial \theta_i} \right) + \frac{1}{2} \operatorname{tr}\left(\alpha^T \frac{\partial P}{\partial \theta_i} \alpha \right) \\ &= \frac{1}{2} \operatorname{tr}\left(\left(\alpha^T \alpha - P^{-1} \right) \frac{\partial P}{\partial \theta_i} \right) \end{aligned} \tag{32}$$

where $P = \widetilde{K} + \sigma^2 I$ and $\alpha = P^{-1} (\gamma - \mu)$.

4) Prediction: The data are generated using simulation tools that emulate real-world VNF performance under varying network conditions. To validate prediction accuracy, test data are derived from flows without severe congestion. According to (32), optimal hyperparameters are computed and applied to a prediction model that takes VNF processing rates as input and allocated resources as output. As illustrated in Fig. 2, the postoptimization fitting curve [see Fig. 2(b)] more closely aligns with actual observations, leading to improved prediction accuracy.

As discussed, the key feature for predicting VNF resource usage is the VNF processing rate, which is highly influenced by real-time traffic characteristics. For instance, severe network congestion can reduce the VNF processing rate. To improve prediction accuracy, the trained model must be continuously updated online with new real-time traffic data, ensuring adaptability to dynamic network conditions.



(a) (b)

Fig. 2. Example of fitting curves. (a) Before hyperparameter optimization. (b) After hyperparameter optimization.

Based on the posterior distribution in (29), we regard it as the initial distribution and re-express it as

$$f_0 = \mathcal{N}(\gamma \mid \mu_0, K_0 + \sigma_0^2 I) \tag{33}$$

where the corresponding initial input and output are $f_0 = \{f_0(\xi^1), f_0(\xi^2), \ldots, f_0(\xi^n)\}$ and $y_0 = \{y_0^1, y_0^2, \ldots, y_0^m\}$, respectively. It is noted that the model should be updated only when the amount of data sample exceeds a certain level, because updating the model with less data is unnecessary and may even reduce the efficiency.

Assuming that we obtain a new sample set $\boldsymbol{\xi_t}$ at the k-th round, the posterior distribution is updated with the integral form, as follows:

$$\begin{aligned} p_k &= p_k\left(f \mid y_k\right) \\ &= \int p\left(y_k \mid f, f_k\right) p\left(f \mid f_k\right) p\left(f \mid y_{k-1}\right) df_k \\ \text{where } f_k &= \{f_k(\xi^1), f_k(\xi^2), \ldots, f_k(\xi^n)\} \\ p_{k-1} &= \mathcal{N}\left(f \mid \mu_{k-1}, K_{k-1} + \sigma_{k-1}^2 I\right) \\ y_{k-1} &= \{y_{k-1}^1, y_{k-1}^2, \ldots, y_{k-1}^m\}. \end{aligned} \tag{34}$$

In addition, GPR supports incremental learning by continuously integrating newly collected data, enabling the model to adapt to evolving network conditions. However, blindly accumulating training samples over time can lead to increased memory and computation overhead, and more critically, may cause the model to drift due to outdated or irrelevant patterns. To address this, we employ a dual strategy: 1) incorporating real-time data to maintain adaptability; and 2) progressively reducing the influence of historical information to mitigate long-term accumulation bias. This is achieved through a forgetting function that dynamically attenuates the influence of older samples, defined as

$$\text{forget}(\xi^s) = \phi^{(a-s)} \tag{35}$$

where $\text{forget}(\xi^s)$ represents the forgetting curve related to the data of ξ^s; a, s are the sequences of the new collected and history data samples; $\phi \in (0, 1)$ is the forgetting factor. A higher value of ϕ leads to longer memory and smoother predictions, while a lower value results in faster adaptation to recent changes.

B. Trigger of Rate Coordination

Building upon the GPR-based traffic prediction model, we now introduce a trigger mechanism for rate coordination. Considering the fact that the traffic of any service chain has to traverse the required VNF instances in strict order, the downstream VNF instance would inevitably cause pressure on the upstream VNF instance, due to the fact that the former may not be able to process the packets from the latter in time. In this condition, the backpressure between any two adjacent VNF instances is calculated by the difference between their real-time packet processing rates. Given any service chain s with the VNF requirement F_s, where $\exists v_i, v_{i-1} \in F_s, \forall i \in (1, |F_s|)$, r_{i-1}^v and r_i^v are the real-time packet processing rates of the upstream VNF v_{i-1} and the downstream VNF v_i, respectively. Then, the backpressure is calculated as follows:

$$\delta_{v_i}^{v_{i-1}} = r_{i-1}^v - r_i^v \quad \forall v_i, v_{i-1} \in F_s, i > 1 \tag{36}$$

where $\delta_{v_i}^{v_{i-1}}$ represents the backpressure between v_i and v_{i-1}, a larger value of $\delta_{v_i}^{v_{i-1}}$ indicates a significant mismatch between the processing capabilities of the two VNFs, meaning that v_{i-1} is sending packets much faster than v_i can process them. This disparity is critical because it leads to an accumulation of packets (i.e., a backlog) at the downstream VNF, potentially degrading overall service performance. To further assess the risk of congestion, we define the following normalized congestion condition:

$$\frac{\delta_{v_i}^{v_{i-1}}}{q_i^v} \begin{cases} > 1, & v_i \text{ is backlogged} \\ \leq 1, & \text{otherwise} \end{cases} \tag{37}$$

where q_i^v denotes the queue threshold, which is initialized according to each VNF's processing characteristics, allowing the mechanism to adapt across heterogeneous VNF types and runtime conditions. The ratio $\frac{\delta_{v_i}^{v_{i-1}}}{q_i^v}$ serves as a congestion indicator. If this ratio exceeds 1, it implies that the upstream VNF's processing rate is excessively higher than what the downstream VNF's queue can accommodate, triggering congestion. Conversely, if the ratio is less than or equal to 1, the processing rates are well-aligned, and congestion is unlikely to occur. Then, according to the reverse of the short plate theory [28], the largest backpressure along the service chain determines the overall service quality, so that we give the service backpressure definition as follows.

Definition 2: Given a service chain s with the VNF requirement F_s, its service backpressure is evaluated by the backpressure between any two adjacent VNF instances. According to the reverse of the short plate theory, the service backpressure of s is calculated as follows:

$$\max_{\substack{i=1 \\ j=i+1}}^{|F_s|-1} \{\delta_{v_i}^{v_j}\} \tag{38}$$

where the larger the value, the worse the service quality and the more the waste of work.

Now, substituting the VNF backpressure in Definition 1 by (37), we achieve the trigger condition for rate coordination of a service chain instead of one single VNF instance, that is

$$\max_{\substack{i=1 \\ j=i+1}}^{|F_s|-1} \left\{ \frac{\delta_{v_i}^{v_j}}{q_j^v} \right\} \begin{cases} > 1, & s \text{ is backlogged} \\ \leq 1, & \text{otherwise.} \end{cases} \tag{39}$$

In particular, from (39), we also find the VNF instance that suffers backpressure the most, which has the highest priority to be optimized when executing rate coordination for the service chain.

C. Prediction-Based Rate Coordination

The prediction process yields a well-trained model that forecasts future resource demands from historical traffic patterns. Typically, increased resource allocation improves VNF processing rates, allowing proactive adjustments when a coordination trigger is predicted. Rate coordination is triggered in two scenarios: 1) the congested VNF has available resources for local mitigation; or 2) it lacks resources, requiring inter-VNF coordination within the same service chain. To address these demands, we adopt heuristic algorithms, which offer near-optimal solutions with low latency. Compared to exact methods, they are more suitable for large-scale scenarios, ensuring efficient and scalable resource management.

Given the VNF v_i of the service chain s with the processing rate $r_i^v(t)$ at the current time t, let the predicted resource demand be $c_i^v(t+1)$, then, we have

$$c_i^v(t+1) \propto r_i^v(t+1). \tag{40}$$

$\forall v_i \in F_s (i \in [1, |F_s|])$, if the condition that

$$\frac{r_{i-1}^v(t+1) - r_i^v(t+1)}{q_i^v} > 1 \tag{41}$$

is true, v_i is predicted to be backlogged at the time point of $t+1$.

Then, we need to execute the rate coordination between v_i and its upstream VNF instance v_{i-1}, because the faster processing rate of v_{i-1} is the main reason leading to such an issue. In this way, we reduce the resource allocated to v_{i-1} until $r_{i-1}^v = r_i^v$ is reached. Nevertheless, it is aware that the service chain includes many VNF instances and executing the rate coordination only between two adjacent VNF instances may fall into the local optimum. Thus, the judgment condition of (41) is expanded to the whole service chain, as follows:

$$\arg \max_{\substack{i=1 \\ j=i+1}}^{|F_s|-1} \frac{r_{i-1}^v(t+1) - r_i^v(t+1)}{q_i^v} > 1. \tag{42}$$

When condition (42) is met, it becomes necessary to reallocate resources on the upstream VNF instance to synchronize their processing rates. This iterative resource adjustment process continues until the service chain attains a balanced state. To ensure the algorithm's real-time feasibility and stability during this resource reallocation, we have implemented several optimization strategies. Specifically, backpressure triggers help minimize unnecessary adjustments, while sparse GPR reduces computational load. In addition, the hybrid approach of offline training combined with online updates enhances adaptability. Furthermore, GPR-predicted preadjustments and gradual reallocation work together to further prevent congestion, allowing the service chain to effectively reach a balanced state.

D. Time Complexity Analysis

The proposed algorithm consists of three main components: the prediction model establishment and hyperparameter optimization process, the online model update process, and the prediction-based rate coordination process. The first component builds upon traditional GPR, which has a worst-case time complexity of $O(n_h^3)$, where n_h represents the number of historical data samples. To address this computational bottleneck, we adopt a sparse approximation strategy that constructs a low-rank surrogate kernel matrix based on n_s sparse samples, reducing the complexity to $O(n_s^2 n_h)$, where $n_s \ll n_h$. This effectively enables GPR inference to scale in large and dynamic environments without sacrificing real-time responsiveness. The second component updates the prediction model online using new data samples. Its time complexity is proportional to the number of new samples, given by $O(\frac{n_e}{x})$, where n_e is the number of new data samples and x is a constant representing the number of samples stored before use. The third component adjusts VNF processing rates through resource reallocation. Since coordination is required only between adjacent VNF instances, the complexity is $O(|F_s|-1) \approx O(|F_s|)$. Therefore, the overall time complexity is $O(n_s^2 n_h) + O(\frac{n_e}{x}) + O(|F_s|) = O(n_s^2 n_h + \frac{n_e}{x} + |F_s|)$.

TABLE I
EXPERIMENT SETTING

Topology setting	
Number of physical nodes of Vlt-Wavennet and CERNET2	83, 41
Number of physical links of Vlt-Wavennet and CERNET2	84, 46
Computing resource of physical node	100 (unit)
Storage resource of physical node	100 (unit)
Link bandwidth	[2.5, 10] (unit)
Flow setting	
Number of services	[0, 120]
Number of VNF in each service chain	[2, 10]
Computing resource required by each flow	[5, 10]
Computing resource required by each VNF	[5, 20]
Number of packets of each flow	[2, 10]
Environment setting	
storage	8GB
OS	Windows 10 (64 b)
Language	Java, Python

V. PERFORMANCE EVALUATION

A. Experimental Environment

To evaluate the performance of the proposed algorithm, we implemented it using Java and Python, and validated it through a series of experiments. Three real-world network topologies—VltWavennet, CERNET2, and America Bone Network [29]—were selected to simulate environments of different scales. VltWavennet represents an Internet backbone network with 83 nodes and 84 links, CERNET2 represents China's Education and Research Network with 41 nodes and 46 links, and America Bone Network consists of 100 nodes and 171 links. For resource configuration, each physical node was provisioned with 100 units of computing and storage resources to ensure balanced resource allocation. Backbone link bandwidth was configured to 10 units, and edge link bandwidth was set to 2.5 units. These settings allow us to simulate the typical bandwidth differences between backbone and edge networks. The resource unit represents the minimum resource required by each VNF, ensuring the algorithm is tested under resource-constrained conditions while reflecting edge computing scenarios.

This study utilized simulated network data to train and test a GPR model for VNF performance prediction. In these simulated data, we covered a variety of network scenarios, including different topologies and traffic patterns, which provided the model with realistic training data. Through simulation, we generated traffic patterns under different network conditions to simulate

TABLE II
METRICS OF DIFFERENT ALGORITHMS UNDER 100 SERVICES

Algorithm	Completion time	Delay time	Delay variance	Handling time	Packet loss rate	Throughput	Capacity
Priority	5.230	3.538	35.086	2.171	0.151	3.939	4.289
Heuristic	4.527	2.710	20.187	1.913	0.006	5.947	2.806
Shunt	4.710	3.124	24.957	2.189	0.0638	5.213	3.211
Rate coordination	3.727	2.224	19.348	1.637	0	6.770	1.792

the actual changes in network traffic. To verify the accuracy of the proposed model, test data were selected from simulated traffic that did not experience congestion. In addition, the traffic flows arrive randomly following a Poisson distribution, and the number of flows is randomly set to [0, 120]. For each flow, we assume that it is separated into [2, 10] packets. The bandwidth demanded by this flow is between [5, 10] units randomly and the number of VNF instances required by this flow is within the range of [2, 10] randomly. For each VNF instance, it requires the computing resource by [5,10] units randomly. The hardware environment used to run the simulation is Intel(R) Core(TM) i7-6700 CPU 3.40 GHz with 8 G memory in a 64 b Windows10 OS.

The overall experiment setting is summarized in Table I.

B. Benchmarks and Metrics

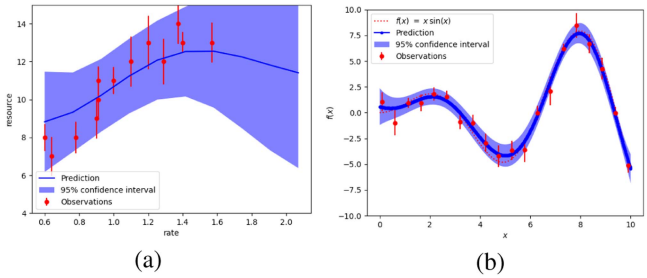
1) Benchmarks: In order to comprehensively evaluate and compare the performance of the proposed algorithm, three state-of-the-art methods are used as benchmarks, as follows.

1) *Priority algorithm*: Focuses on balancing service quality and fairness, making it widely used in real-world networks for ensuring high-priority tasks receive appropriate resources. It has a time complexity of $O(n \log n)$, where n is the number of VNF instances [12].
2) *Improved slope algorithm*: A computationally efficient method that minimizes service completion time, commonly used in complex optimization problems. It has a time complexity of $O(n^2)$ [23].
3) *Shunt algorithm*: Enhances load balancing and throughput by distributing traffic across multiple paths, improving resource utilization and network performance. It has a time complexity of $O(\log n)$ [18].

In addition, for convenience, the proposed algorithm is referred to as the rate coordination algorithm.

2) Metrics: The corresponding metrics used to verify the effectiveness of the proposed algorithm are as follows.

1) *Prediction result*: It evaluates the prediction accuracy and the balance condition among VNF instances after executing the rate coordination.
2) *Average completion time*: It evaluates the time to finish a service task and the less, the better.
3) *Average delay and variance*: It evaluates the time for a packet to be transmitted from source to destination and the corresponding variance evaluates the stable condition of delay. The smaller the delay and variance, the better.
4) *Average processing time*: It evaluates the time spent on packet processing.
5) *Packet loss rate*: It evaluates the rate of packets being dropped in the experiments.
6) *Throughput*: It evaluates the number of packets been processed and forwarded within a time unit.
7) *Backlog*: It evaluates the number of packets accumulated in the queue.



(a) (b)

Fig. 3. Prediction results. (a) Prediction accuracy. (b) Validation result.

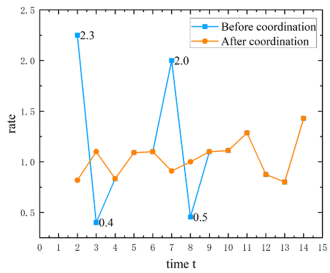


Fig. 4. VNF processing rate before and after coordination.

C. Experimental Results

1) Prediction Results: The prediction model estimates future resource demands based primarily on VNF processing rates, directly affecting the proposed algorithm's performance. As shown in Fig. 3, the prediction results [see Fig. 3(a)] and validation results [see Fig. 3(b)] indicate an average prediction accuracy of approximately 88%, reaching nearly 95% when processing rates range from [0.9, 1.3]. For further validation, we compare the predictions against another function, $f(x) = x \cdot \sin(x)$, with sample prediction errors remaining within 5%, demonstrating high accuracy. Based on these predictions, rate coordination is executed to preemptively mitigate potential VNF congestion, as depicted in Fig. 4, which shows the average processing rate of a VNF instance over time, with two lines representing rates before and after coordination. Before coordination, VNF processing rates fluctuate significantly, dropping from 2.3 to 0.4 at $t = 2$ and from 2 to 0.5 at $t = 7$. Such fluctuations can lead to severe congestion and inefficient resource utilization. In contrast, after coordination, processing rates stabilize over time, with variances calculated at 0.286 before and 0.044 after coordination. These results confirm the effectiveness of the proposed rate coordination algorithm in balancing processing rates and preventing congestion.

2) Average Completion Time: The results of the average completion time are shown in Fig. 5. Fig. 5(a) presents the average completion time for different algorithms in the large-scale VltWavennet topology, while Fig. 5(b) shows the results for the

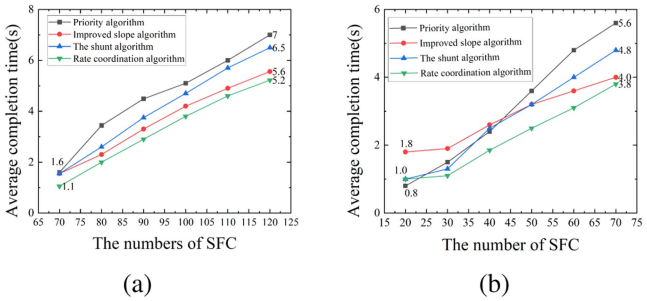


Fig. 5. Average completion time. (a) VltWavennet. (b) CERNET2.

small-scale CERNET2 topology. As expected, the average completion time increases with the number of services for all four algorithms. This is reasonable, as a higher network load leads to longer service completion times and increased congestion. Among them, the priority-based algorithm exhibits the fastest increase in completion time. This is primarily due to the long waiting times of low-priority flows, which make up the majority of network traffic. For instance, when the number of services reaches 120, the service completion time of the priority-based algorithm is 1.6 times longer than that of the proposed rate coordination algorithm.

In contrast, the shunt-based, improved slope-based, and rate coordination-based algorithms show a more gradual increase in completion time as the number of services grows. Despite this, the proposed rate coordination algorithm achieves the lowest completion time, demonstrating superior performance. The main reasons include the following:

1) the shunt-based algorithm struggles to find suitable VNF instances for parallel transmission with the increasing number of services;
2) the improved slope-based algorithm lacks the strategy to handle the dynamically changing environment;
3) In contrast, the proposed algorithm employs online updating to continuously improve its model and adapt to new situations, thus achieving the best performance.

3) Average Delay and Variance: The results of average delay achieved by different algorithms are shown in Fig. 6, where Fig. 6(a) is the average delay achieved in VltWavennet and Fig. 6(b) is the average delay achieved in CERNET2. This is attributed to the increased network load and processing time increases naturally. Besides, we also see that the average delay is relatively small when the number of services is small, but the average delay begins to increase noticeably when the number of services exceeds 50 in CERNET2, indicating the signal of network congestion. Under this situation, the priority-based algorithm performs worse, because low-priority packets may experience long waiting times before being scheduled, leading to unfairness and longer delays (about 1.64 times that of the proposed algorithm).

As for the other three methods, they have a certain ability to handle congestion control, so that their delay increases slowly. Despite this, when the network load exceeds the threshold, the disadvantages of the shunt and improved slope-based algorithms begin to appear. For example, the probability of successfully finding proper VNF instances for traffic split becomes lower, so the delay of the shunt algorithm begins to increase faster and it is about 1.35 times that of the proposed algorithm. Lastly, the proposed rate coordination algorithm outperforms the Improved

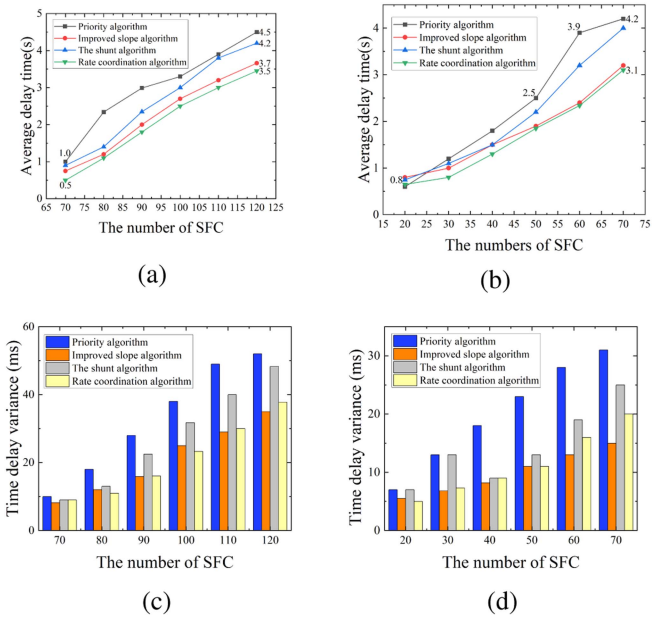


Fig. 6. Average delay and variance. (a) Delay in VltWavennet. (b) Delay in CERNET2. (c) Variance in VltWavennet. (d) Variance in CERNET2.

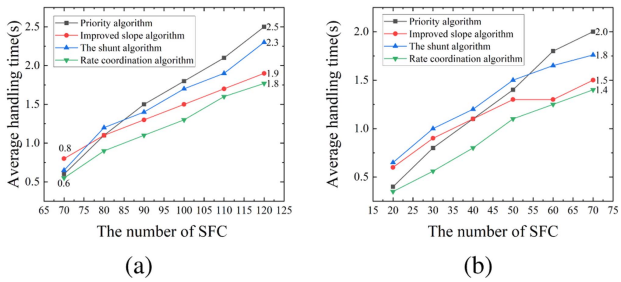


Fig. 7. Average processing time. (a) VltWavennet. (b) CERNET2.

slope algorithm greatly when the number of services is smaller than 50. However, when the number of services exceeds 50, they achieve similar delay results. Overall, the rate coordination algorithm still outperforms the improved slope method.

Fig. 6(c) and (d) illustrates the variance results of delay distribution, with larger variance indicating greater instability. The priority-based algorithm has the highest variance due to long wait times, meanwhile the improved slope-based algorithm has the lowest. The proposed rate coordination algorithm achieves comparable variance to the improved slope algorithm, but with significantly lower average delay. Hence, jointly taking these situations into consideration, the proposed algorithm has superior performance.

4) Average Processing Time: The results of the average processing time achieved by different algorithms are shown in Fig. 7, where Fig. 7(a) presents the processing time results in VltWavennet, while Fig. 7(b) presents the processing time results in CERNET2.

The experimental results indicate that all four algorithms perform well when the number of services is below 40 in CERNET2 and 80 in VltWavennet. However, as the number of services increases, network congestion arises due to higher traffic volume, leading to longer processing times. The shunt

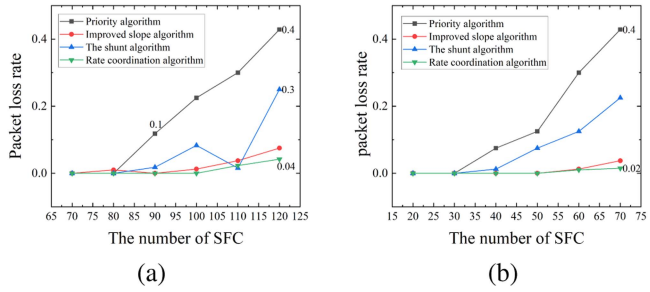


Fig. 8. Packet loss rate. (a) VltWavennet. (b) CERNET2.

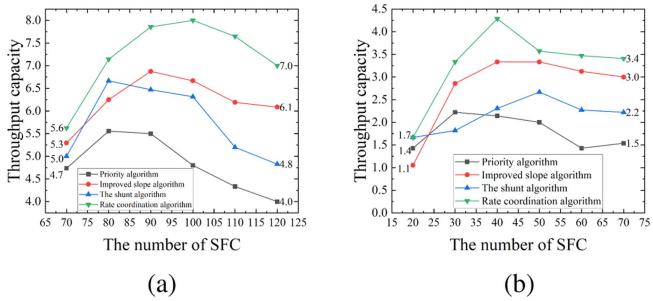


Fig. 9. Throughput. (a) VltWavennet. (b) CERNET2.

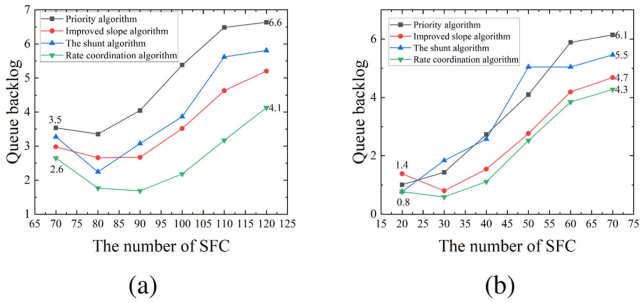


Fig. 10. Queue backlog. (a) VltWavennet. (b) CERNET2.

and priority-based algorithms are more vulnerable to congestion and exhibit higher average processing times compared to the improved slope and rate coordination algorithms. Although the improved slope algorithm achieves lower processing times than the priority and shunt-based algorithms, it still lags behind the rate coordination algorithm. In contrast, the proposed rate coordination algorithm proactively predicts and mitigates potential congestion by taking preemptive measures based on prediction results, ensuring superior performance.

5) Packet Loss Rate: The results of the packet loss rate achieved by different algorithms in VltWavennet and CERNET2 are shown in Fig. 8, where Fig. 8(a) presents the packet loss rate in VltWavennet, while Fig. 8(b) presents the packet loss rate in CERNET2.

When the number of services is small, the packet loss rates of all four algorithms approach zero due to light network loads and the capacity limits not being exceeded. However, as the number of services increases, effective strategies are required to mitigate packet loss. The priority-based algorithm exhibits the highest and most rapidly increasing packet loss rate, as it fails to manage congestion, resulting in dropped packets. Although the shunt algorithm distributes traffic across different paths and VNF instances, its effectiveness diminishes under high network loads, leading to a rapid decline in performance. In contrast, the proposed rate coordination algorithm maintains the lowest and most stable packet loss rate as the number of services increases, demonstrating its effectiveness in preventing congestion. While both the improved slope and rate coordination algorithms exhibit slow increases in packet loss rates, the advantages of the proposed rate coordination approach become increasingly evident with higher service volumes.

6) Throughput: The improved slope and rate coordination algorithms display slow and delayed increases in packet loss rates. However, the advantages of the proposed rate coordination algorithm become significant when the number of services is high, as observed by the results for 120 services in VltWavennet and 70 services in CERNET2, while the throughput results for each algorithm in VltWavennet and CERNET2 are shown in Fig. 9(a) and (b), respectively.

The figure shows that the throughput initially increases and then stabilizes with an increase in the number of services, which is proportional to the number of packets processed. However, when the number of services exceeds the network threshold of 90 in VltWavennet and 40 in CERNET2, the throughput immediately suffers from a short decrease and then becomes stable gradually. In addition, comparing the four algorithms from the overall perspective, we can notice that the throughput of the rate coordination algorithm is the highest, while that of the priority algorithm is the lowest in most cases. Nevertheless, there are still some unexpected conditions, for example, the improved slope algorithm has the lowest throughput when the number of services is 20 and the shunt algorithm has the lowest throughput when the number of services is 30 in CERNET2.

7) Queue Backlog: The results of queue backlog achieved by different algorithms in VltWavennet and CERNET2 are shown in Fig. 10, where Fig. 10(a) presents the queue backlog in VltWavennet, while Fig. 10(b) presents the queue backlog in CERNET2.

The backlog indicator measures the number of packets in the queue. As shown in Fig. 10, the backlog exhibits a generally stable growth trend as the number of services increases across all four algorithms, in contrast to the increasing throughput trend observed in Fig. 9. In terms of detailed results, the proposed rate coordination algorithm consistently achieves the lowest backlog across different service volumes in both VltWavennet and CERNET2. This is primarily because it effectively balances network conditions, reducing the load on individual VNF instances. In addition, the algorithm accounts for strict VNF sequence constraints when provisioning services, further improving performance.

To further demonstrate the superiority of our algorithm, we conducted experiments in a larger-scale network environment (i.e., America Bone Network) and evaluated its performance in a scenario with 100 services. Table II shows the experimental results, indicating that our algorithm has significant performance advantages in this scenario.

VI. Conclusion

In this work, we propose a VNF rate coordination algorithm based on GPR and optimization. Specifically, the GPR model predicts and captures the relationship between VNF processing

rates and allocated resources, enabling informed rate coordination decisions to alleviate network congestion. The prediction model is continuously refined with newly collected data to adapt to dynamic network environments. Experimental results demonstrate that the proposed algorithm outperforms state-of-the-art methods in average completion time, delay, packet loss rate, throughput, and backlog. While this study focuses on short-to-medium-term dynamics, future work will extend the "predict-then-optimize" framework with lightweight learning to handle higher dimensional dynamics, including energy efficiency, cost, and quality of service (QoS), particularly in long-term or abrupt reconfiguration scenarios, while balancing computational overhead and adaptability.

REFERENCES

[1] X. Lv, S. Rani, S. Manimurugan, A. Slowik, and Y. Feng, "Quantum-inspired sensitive data measurement and secure transmission in 5G-enabled healthcare systems," *Tsinghua Sci. Technol.*, vol. 30, no. 1, pp. 456–478, 2025.

[2] Z. Wang, B. Yi, S. Kumari, C. M. Chen, and M. J. F. Alenazi, "Graph convolutional networks and deep reinforcement learning for intelligent edge routing in IoT environment," *Comput. Commun.*, vol. 232, Feb. 2025, Art. no. 108050.

[3] J. Lv, B. Kim, B. D. Parameshachari, A. Slowik, and K. Li, "Large model-driven hyperscale healthcare data fusion analysis in complex multi-sensors," *Inf. Fusion*, vol. 115, pp. 1–15, Mar. 2025.

[4] W. Wang et al., "Anomaly detection of industrial control systems based on transfer learning," *Tsinghua Sci. Technol.*, vol. 26, no. 6, pp. 821–832, Dec. 2021.

[5] J. Wu, "Challenges and opportunities in algorithmic solutions for re-balancing in bike sharing systems," *Tsinghua Sci. Technol.*, vol. 25, no. 6, pp. 721–733, Dec. 2020.

[6] C. Paniagua and J. Delsing, "Industrial frameworks for Internet of Things: A survey," *IEEE Syst. J.*, vol. 15, no. 1, pp. 1149–1159, Mar. 2021.

[7] W. Bekri, R. Jmal, and L. C. Fourati, "Softwarized Internet of Things network monitoring," *IEEE Syst. J.*, vol. 15, no. 1, pp. 826–834, Mar. 2021.

[8] D. Li, P. Hong, K. Xue, and J. Pei, "Virtual network function placement considering resource optimization and SFC requests in cloud datacenter," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 7, pp. 1664–1677, Jul. 2018.

[9] G. Sun, G. Zhu, D. Liao, H. Yu, X. Du, and M. Guizani, "Cost-efficient service function chain orchestration for low-latency applications in NFV networks," *IEEE Syst. J.*, vol. 13, no. 4, pp. 3877–3888, Dec. 2019.

[10] J. F. Riera, E. Escalona, J. Batalle, E. Grasa, and J. A. Garcia-Espin, "Virtual network function scheduling: Concept and challenges," in *Proc. Int. Conf. Smart Commun. Netw. Technol.*, 2014, pp. 1–5.

[11] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and S. Davy, "Design and evaluation of algorithms for mapping and scheduling of virtual network functions," in *Proc. 1st IEEE Conf. Netw. Softwarization*, 2015, pp. 1–9.

[12] D. Liao et al., "AI-based software-defined virtual network function scheduling with delay optimization," *Cluster Comput.*, vol. 22, no. 6, pp. 13897–13909, 2019.

[13] H. Cao, H. Zhu, and L. Yang, "Dynamic embedding and scheduling of service function chains for future SDN/NFV-enabled networks," *IEEE Access*, vol. 7, pp. 39721–39730, 2019.

[14] H. Yao et al., "Joint optimization of function mapping and preemptive scheduling for service chains in network function virtualization," *Future Gener. Comput. Syst.*, vol. 108, pp. 1112–1118, 2020.

[15] B. Yi, X. Wang, and M. Huang, "A generalized VNF sharing approach for service scheduling," *IEEE Commun. Lett.*, vol. 22, no. 1, pp. 73–76, Jan. 2018.

[16] S. G. Kulkarni et al., "NFVnice: Dynamic backpressure and scheduling for NFV service chains," in *Proc. Conf. ACM Special Int. Group Data Commun.*, 2017, pp. 71–84.

[17] J. G. Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 3, pp. 518–532, Sep. 2016.

[18] M. T. Beck, J. F. Botero, and K. Samelin, "Resilient allocation of service function chains," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw.*, 2016, pp. 128–133.

[19] J. Elias, F. Martignon, S. Paris, and J. Wang, "Efficient orchestration mechanisms for congestion mitigation in NFV: Models and algorithms," *Serv. Comput.*, vol. 10, no. 4, pp. 534–546, 2017.

[20] L. Guet al., "Fairness-aware dynamic rate control and flow scheduling for network utility maximization in network service chain," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1059–1071, May 2019.

[21] Y. Xieet al., "Virtualized network function forwarding graph placing in SDN and NFV-enabled IoT networks: A graph neural network assisted deep reinforcement learning method," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 1, pp. 524–537, Mar. 2022.

[22] B. Yang, W. K. Chai, Z. Xu, K. V. Katsaros, and G. Pavlou, "Cost-efficient NFV-enabled mobile edge-cloud for low latency mobile applications," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 1, pp. 475–488, Mar. 2018.

[23] Y. Chen and J. Wu, "Flow scheduling of service chain processing in a NFV-based network," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 1, pp. 389–399, Jan.–Mar. 2021.

[24] P. Germainet al., "PAC-Bayesian theory meets Bayesian inference," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, vol. 29, pp. 1–10.

[25] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral grouping using the Nyström method," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 2, pp. 214–225, Feb. 2004.

[26] S. Gengbiao, L. Qing, J. Yong, W. Yu, and L. Jianhui, "A four-stage adaptive scheduling scheme for service function chain in NFV," *Comput. Netw.*, vol. 175, 2020, Art. no. 107259.

[27] M. Franchi and P. Paolo, "A general inversion theorem for cointegration," *Econometric Rev.*, vol. 38, no. 10, pp. 1176–1201, 2019.

[28] G. Li, Q. Zhang, Q. Lin, and W. Gao, "A three-level radial basis function method for expensive optimization," *IEEE Trans. Cybern.*, vol. 52, no. 7, pp. 5720–5731, Jul. 2022.

[29] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.