Proactive Network Slice Reconfiguration by Exploiting Prediction Interval and Robust Optimization

Fengsheng Wei, Gang Feng, *Senior Member, IEEE*, Yao Sun, Yatong Wang, Shuang Qin, *Member, IEEE*
University of Electronic Science and Technology of China
E-mail:fenggang@uestc.edu.cn

***Abstract*—It is widely acknowledged that the agile reconfiguration of network slice according to traffic demand is of vital importance in 5G-and-beyond systems. Existing relevant works make reconfiguration decisions based either on point prediction of the uncertain demand, which lacks indications on how accurate it is, or on handcrafted uncertainty set with robust optimization, which may lead to resource over-provisioning due to the lack of prediction mechanism. To overcome these drawbacks, in this paper, we propose a predictor-optimizer framework that intelligently performs inter-slice reconfiguration with the aim of minimizing the energy consumption of serving these slices. Specifically, the predictor produces a prediction interval comprised of lower and upper bounds that bracket the future traffic demands with a prespecified probability. Then by regarding the prediction interval as the uncertainty set, we formulate the network slice reconfiguration problem as a Robust Mixed Integer Programming (RMIP). We solve this RMIP by using linearization technique and robust optimization. Numerical results demonstrate that the proposed framework outperforms traditional methods in terms of robustness and energy consumption. Meanwhile, the trade-off between robustness and the energy consumption can be automatically adjusted according to the type of slice and traffic demands.**

I. Introduction

NEtwork slicing has been regarded as a key technology to address the diversified service requirements of various scenarios in 5G-and-beyond systems [1], [2]. Based on the flexibility provided by Software Defined Network (SDN) and Network Function Virtualization (NFV), network slicing enables the on-demand, fully-atomized provisioning of services according to the user's traffic demands. However, the traffic demands in a network are highly dynamic. Thus, during their lifetime, network slices need to be adaptively reconfigured according to the customers' traffic demands.

A network slice is defined by its Service Function Chain (SFC), which is composed of sequentially interconnected Virtual Network Functions (VNFs) [3]. Therefore, inter-slice reconfiguration mainly involves VNF scaling and VNF migration [4], which are time-consuming and costly. Moreover, from the Quality of Service (QoS) perspective, migrating VNFs may cause service interruption due to the migration of the virtual machines and their associated traffic data. Therefore, considering reconfiguration cost and QoS degradation, proactive slice reconfiguration with respect to traffic demand becomes imperative.

Recently, two techniques are widely used to accomplish proactive reconfiguration: Robust Optimization (RO)-based methods and prediction-based methods. In RO-based investigations such as [5], [6], the traffic demands are assumed to be within an *uncertainty set* which is used to formulate an RO model. The solution of the RO model is "robust" enough such that any constraints in the uncertainty set should be satisfied. Due to the lack of prediction mechanism, the width of the uncertainty set in these works cannot be automatically adjusted, leading to an issue of resource over-provisioning even under the scenarios with low traffic variation.

On the other hand, in prediction-based researches such as [7], [8], the authors usually choose to predict future traffic demands and then formulate the slice reconfiguration as a deterministic optimization problem based on the predicted results. However, the predictor used in these methods is single point predictor that does not provide any indication of its accuracy. Moreover, point predictions could be unreliable and inaccurate if the training data is sparse, or if the predicted targets are affected by probabilistic events [9]. This causes inaccuracy of the formulated optimization problem, and thus the obtained optimal solution cannot accurately match future traffic demands, resulting in resource over-provisioning or Service Level Agreement (SLA) violation.

Unlike point prediction that forecasts the traffic demands without confidence level, *prediction interval* [9] gives a range of future traffic demand with a prespecified probability called *confidence interval* $(1-\alpha)$, which can be flexibly set according to the slice types. Furthermore, since network slices are highly dynamic networks that may have short or long-lived effects (e.g., changes in the network topology and in the number of connected customers), the overall traffic demands exhibit periodicity as well as variability [10], which can be reflected by the prediction intervals. Accordingly, this information can be used to perform better slice reconfiguration decisions.

As such, in this paper, we propose the *predictor-optimizer*, a proactive network slice reconfiguration framework that jointly exploits prediction interval and RO. First, the predictor fore-

This work has been supported by the R&D Program in Key Areas of Guangdong Province (Grant No. 2018B010114001), China Postdoctoral Science Foundation under Grant 2019M663476, and the General Program of National Natural Science Foundation of China (Grant No. 61871099).

casts the aggregate traffic demands of each slice and produces the $(1-\alpha)$ prediction interval, where α is specified according to the type of slice. In the design of the optimizer, we then formulate the network slice reconfiguration problem as an RO model by using the prediction intervals as its uncertainty set. Finally, through linearizing the RO model and deriving its *robust counterpart*, the RO model is transformed into a Mixed Integer Linear Programming (MILP) and can be solved efficiently.

The main contributions of this paper are summarized as follows:

1) We propose to exploit prediction interval-oriented predictors to forecast the traffic demands with confidence level, which guides the subsequent slice reconfiguration without making risky decisions under uncertainty.

2) By jointly utilizing prediction interval and RO, we design a predictor-optimizer framework that proactively reconfigures the network slice. This novel approach opens the door for further attempts to take advantage of prediction interval to address the uncertainty issues in network slicing problems.

3) We employ Long Short-Term Memory (LSTM) and the bootstrap method to produce prediction intervals for the traffic demands. Through simulations based on real traffic data, we demonstrate that our proposed predictor-optimizer framework can achieve high robustness at a low resource price. Moreover, the size of the prediction interval can be automatically and adaptively adjusted according to the traffic demands, thereby enabling flexible trade-off between robustness and resource cost.

The remainder of the paper is organized as follows: Section II elaborates on the prediction interval oriented predictor. In Section III, we present the system model and formulate the network reconfiguration problem as a robust MIP. In Section IV, the robust MIP is linearized to a robust MILP and its robust counterpart is derived. In Section V, we present the numerical results, and finally we conclude the paper in Section VI.

II. Overview of Predictor Design

We consider the scenario in which $|\mathcal{K}|$ network slices of different types are running on top of a common infrastructure. During the lifetime of these slices, their aggregated traffic demands are varying due to the changing requirements of their users. Therefore, the slices need to be reconfigured accordingly to match the ever-changing demands. To overcome the demand uncertainty, for each slice, a corresponding predictor that generates prediction intervals of the future demands is trained by the historical data. In the following, we elaborate the design of the predictor.

A. The Overall Design of the Predictor

The predictor is used to predict the aggregated traffic demands of the network slice in future time windows. The predictions are made on the basis of historical traffic demands of each slice, i.e., $r_0^k, \cdots, r_{t-1}^k, \forall k \in \mathcal{K}$, where $\mathcal{K}$ is the set of network slices in the system. The aggregated traffic demands of a network slice is a time series which exhibits strong periodicities and distinct overall long-term trends [10]. Therefore, the trend of traffic demands of a network slice can be predicted by the popular LSTM network, which is an improved Recurrent Neural Network (RNN) for long time-dependent series predictor.

However, LSTM suffers from two limitations despite its popularity in predicting traffic demands. LSTM produces unsatisfactory prediction accuracy under the case of high traffic uncertainty. This is because of the unpredictable nature of the short-lived events which cause traffic bursts beyond the overall trend. Moreover, LSTM only provides point estimates, without any indication of the confidence of the forecasts. What is worse, the accuracy and reliability of point predictions drop sharply when the traffic demands fluctuate heavily. These drawbacks make point predictions unable to provide sufficient information and the reliability for subsequent resource allocation.

Instead, prediction intervals can cover future traffic values with a given probability, thereby making the predictions more meaningful for the subsequent decisions. Therefore, instead of pursuing an unreliable point prediction, we propose to predict an interval that the real traffic demands can be covered with the prespecified probability $(1-\alpha)$. Such prediction intervals not only give the prediction results, but also the accuracy of the prediction. There are a number of methods to produce prediction intervals, such as bootstrap method, Bayesian method, Delta method, etc [9]. In the design of our predictor, due to the stability of the bootstrap method under data uncertainty [9], we exploit this method to generate prediction intervals for the traffic demands of the network slice. A detailed principle behind the bootstrap-based prediction interval is elaborated in the following subsection.

B. The Principle of Prediction Interval

The prediction target (i.e., the real traffic demands to be predicted at time t, r_t^k) can usually be modeled by

$$r_t^k = y_t^k + \epsilon_t^k, \tag{1}$$

where y_t^k is the true regression mean, and ϵ_t^k is the noise which has zero mean and is dependently and identically distributed. In practice, the true regression mean y_t^k is approximated by the output of the LSTM $\bar{y}_t^k$. Thus, we have

$$r_t^k - \bar{y}_t^k = (y_t^k - \bar{y}_t^k) + \epsilon_t^k. \tag{2}$$

To quantify the difference between the real traffic r_t^k and the predicted traffic $\bar{y}_t^k$, we need to evaluate the mean and variance of their difference. The basic idea behind the bootstrap method is that an ensemble of LSTMs will produce a less biased estimate of the true regression y_t^k. Therefore, N LSTM models are built and the true regression is approximated by the mean of the point predictions of these LSTM models, i.e.,

$$\bar{y}_t^k = \frac{1}{N}\sum_{i=1}^{N} \bar{y}_t^{k,i}, \tag{3}$$

where $\bar{y}_t^{k,i}$ is the prediction result of traffic demand of the ith LSTM network. Suppose that $y_t^k - \bar{y}_t^k$ and ϵ_t^k are statically independent, from (2) we have

$$\sigma_{r_t^k}^2 = \sigma_{\bar{y}_t^k}^2 + \sigma_{\epsilon_t^k}^2, \tag{4}$$

where $\sigma_{\bar{y}_t^k}^2$ is the variance caused by model misspecification errors, and $\sigma_{\epsilon_t^k}^2$ is the variance of random noise. The variance caused by model misspecification is approximated by the variance of the N LSTM models' output:

$$\sigma_{\bar{y}_t^k}^2 = \frac{1}{N-1}\sum_{i=1}^{N}(\bar{y}_t^{k,i} - \bar{y}_t^k) \tag{5}$$

From (2) and (4), the variance of the noise can be approximated by:

$$\sigma_{\epsilon_t^k}^2 \simeq E[(r_t^k - \bar{y}_t^k)^2] - \sigma_{\bar{y}_t^k}^2. \tag{6}$$

We train another LSTM with the same structure as the ones used for the point predictions, and use it to predict the remaining residuals from the input values:

$$\delta_{t,k}^2 = max((y_t^k - \bar{y}_t^k)^2 - \sigma_{\epsilon_t^k}^2, 0). \tag{7}$$

The activation function of the residual LSTM output node is set to be exponential, making a positive value for $\sigma_{\epsilon_t^k}^2$. According to (7), a new dataset, $\{(r_\tau^k, \delta_{t,k}^2)\}_{\tau=1}^{t-1}$, is created to train the residual LSTM. The maximum likelihood is used as the loss function, which has the following compact form [9]:

$$L = \frac{1}{2}\sum_{\tau=1}^{t-1}[ln(\sigma_{\epsilon_t^k}^2) + \frac{\delta_{t,k}^2}{\sigma_{\epsilon_t^k}^2}]. \tag{8}$$

Now we know both the mean and variance of the point prediction, the prediction interval can be calculated from the t-distribution. The $(1-\alpha)\%$ prediction interval for r_t^k is computed as:

$$PI_t^k = [\bar{y}_t^k - \tau_{1-\alpha/2}\cdot\sigma_{\bar{y}_t^k}, \bar{y}_t^k + \tau_{1-\alpha/2}\cdot\sigma_{\bar{y}_t^k}], \tag{9}$$

where $\tau_{1-\alpha/2}$ is the $(1-\alpha/2)$ percentile of t-distribution. The prediction interval generated above will be feed to the input of the robust optimizer, which will be elaborated in the following subsequent sections.

III. The Robust Network Slice Reconfiguration Problem

Rather than some existing approaches in the literature which make resource decision based on the upper bound of the prediction [8], or apply robust optimization that uses handcraft uncertainty set [5], we instead propose to use the prediction interval as the uncertainty set of RO to reconfigure the network slices. In this section, we present the system model as well as the formulation of the robust problem. The solution of the problem will be given in the next section.

1) Infrastructure Model: We model the substrate network as a directed graph denoted by $G = (V, E)$, where V and E denote the set of physical nodes and physical links respectively. The computing capacity of node $i \in V$ is denoted by c_i, and the bandwidth of link $(i,j) \in E$ is denoted by b_{ij}. Unlike the virtual network embedding problem in which the physical nodes can offer all the required VNFs, the nodes in network slicing can only offer a subset of the VNF set $\mathcal{F}$ in the system. Thus, we use a binary indicator variable $\delta_i(f)$ to indicate whether node $i \in V$ can provide function $f \in \mathcal{F}$ (i.e., $\delta_i(f) = 1$ if node $i \in V$ is capable of function $f \in \mathcal{F}$, otherwise $\delta_i(f) = 0$).

2) Network Slice Model: There are $|\mathcal{K}|$ slices deployed over the substrate network, where $\mathcal{K}$ denotes the set of network slices. The SFC of slice k consists of l_k ordered VNFs denoted by $\mathcal{F}_k = \{f_1^k, \cdots, f_{l_k}^k\}$. For ease of presentation, we use two dummy VNFs f_0^k and $f_{l_k+1}^k$ to denote the ingress node s_k and egress node d_k of slice k, respectively. The virtual link between the virtual nodes f_m^k and f_{m+1}^k is denoted by e_m^k. For each slice, we build a predictor that outputs the prediction interval of the traffic demand at time t, which is denoted by $\left[\bar{r}_t^k - \hat{r}_t^k, \bar{r}_t^k + \hat{r}_t^k\right]$, where $\hat{r}_t^k = \tau_{1-\alpha/2}\delta_{\bar{y}_t^k}$ as that in (9). This interval is regarded as the uncertainty set of the robust optimizer. For ease of representation, we denote the traffic demand of slice k by $\tilde{r}_t^k$, which falls within the prediction interval $\left[\bar{r}_t^k - \hat{r}_t^k, \bar{r}_t^k + \hat{r}_t^k\right]$ with probability $(1-\alpha)$.

3) The Constraints: The VNF instances (VNFI) of the network slices are created and managed by the VNF Manager (VNFM) under the VNF-MANO framework proposed by ESTI. Practically, a VNFI is implemented as a Virtual Machine (VM) or container which is deployed on a specific physical node. We use the binary variable $x_{i,k}(f_m^k)$ to indicate whether the virtual node f_m^k of slice k is deployed on physical node i or not. For virtual node f_m^k, only the physical nodes with function f_m^k can be its mapping target. Therefore, we have

$$x_{i,k}(f_m^k) \le \delta_i(f_m^k), \forall f_m^k \in \mathcal{F}_k, \forall k \in \mathcal{K}, i \in V. \tag{10}$$

Similarly, we use binary variable $z_{p,q}(e_m^k) \in \{0,1\}$ to indicate whether virtual link e_m^k is mapped to the physical link (p,q). To avoid coordination overhead caused by path splitting, we require that each VNF is mapped to exactly one physical node and each virtual link is mapped to exactly one physical path. To this end, we have the following constraints to avoid node splitting:

$$\sum_{i\in V} x_{i,k}(f_m^k) = 1, \forall f_m^k \in \mathcal{F}_k, \forall k \in \mathcal{K}. \tag{11}$$

To avoid link splitting, we have the following two constraints for node $i \in V$:

$$\sum_{p\in V} z_{i,p}(e_m^k) \le 1, \forall m \in \{1,\cdots,l_k-1\}, \forall k \in \mathcal{K}, \tag{12}$$

$$\sum_{q\in V} z_{q,i}(e_m^k) \le 1, \forall m \in \{1,\cdots,l_k-1\}, \forall k \in \mathcal{K}. \tag{13}$$

Furthermore, the variables $\{z_{p,q}(e_m^k)\}$ and $\{x_{i,k}(f_m^k)\}$ should be constrained to form a connected path for each network slice [11]. Therefore, for all $(i,k,f_m^k) \in V \times \mathcal{K} \times$

$\mathcal{F}_k \cup \{f_0^k\}$, we have

$$\begin{cases} \sum_i z_{p,i}(e_m^k) - \sum_i z_{i,p}(e_m^k) = 1, \text{ if } p = s_k, m = 0 \\ \sum_i z_{p,i}(e_m^k) - \sum_i z_{i,p}(e_m^k) = -1, \text{ if } p = d_k, m = l_k \\ \sum_i z_{p,i}(e_m^k) - \sum_i z_{i,p}(e_m^k) = x_{i,k}(f_m^k) - x_{i,k}(f_{m+1}^k), \text{ else} \end{cases} \tag{14}$$

Since some VNFs will possibly compress or encrypt the incoming packets, the traffic rates entering each VNFs are different. Indeed, the incoming traffic rates of VNF f_m^k is computed as

$$\widetilde{r}_t^k(f_m^k) = \widetilde{r}_t^k \cdot \prod_{l=1}^{m-1} \theta(f_l^k) \tag{15}$$

where $\theta(f_l^k)$ is the compression ratio of VNF f_l^k. We assume one unit data flow consumes β_c units of computing resources and β_b units of bandwidth. Therefore, we have the following capacity constraints on each physical node:

$$\widetilde{c}_{allc}^i = \sum_{k\in\mathcal{K}} \sum_{f_m^k \in \mathcal{F}_k} \beta_c x_{i,k}(f_m^k) \widetilde{r}_t^k(f_m^k) \le c_i, \forall i \in V. \tag{16}$$

Similarly, we have the following link capacity constraint:

$$\sum_{k\in\mathcal{K}} \sum_{m=1}^{l_k-1} \beta_b z_{p,q}(e_m^k) \widetilde{r}_t^k(f_m^k) \le b_{pq}, \forall (p,q) \in E. \tag{17}$$

4) Objective Function: In this study, our objective is to minimize the energy consumption of the system, since it accounts for a large proportion of the operators' expenditure and becomes the focus of attention [12]. In our model, we focus only on the power consumption of servers and communication networks. For node $i \in V$, its power consumption consists of three parts: the power consumed by the VNFs, and the power consumed by the Network Interface Cards (NICs) which are used for packet forwarding, and the power required to turn on server i [12], i.e.,

$$\widetilde{P}^i = P_{cpu}^i + P_{NIC}^i + P_{static}^i \tag{18}$$

First, the power consumed by the VNFs on server i is decided by the amount of allocated computing resources [12], i.e.,

$$P_{cpu}^i = (P_{max}^i - P_{idle}^i) \times \widetilde{c}_{allc}^i / c_i. \tag{19}$$

where P_{max}^i is the maximum power consumption at full utilization, P_{idle}^i is the power consumption of server i on idle state, and $\widetilde{c}_{allc}^i$ is defined as in (16).

Second, the power consumption of NICs is a significant contributor to the system power consumption which can account for up to 10% of the total power consumption [12]. As in [12], the power consumption of NICs is different under idle state and busy state. In our model, since the traffic flows are not split, the activated NICs are always working in busy state. Consequently, the power consumption model of NICs on server i is calculated as:

$$P_{NIC}^i = P_{port} \times N_{port}^i, \tag{20}$$

where P_{port} is the power consumption of the NICs in active mode, which corresponds to $P_{dynamic}$ in (129) of [12], and N_{port}^i is the number of activated NICs on server i, which can be calculated as:

$$N_{port}^i = \sum_{k\in\mathcal{K}} \sum_{m}^{l_k-1} \left(\sum_{q\in V} z_{i,q}(e_m^k) + \sum_{q\in V} z_{q,i}(e_m^k) \right). \tag{21}$$

Finally, the power required to keep the ith server on is calculated according to:

$$P_{static}^i = y^i P_{idle}^i, \tag{22}$$

where y^i is a binary variable indicates whether server i is on or not. The on-off state of server i is only dependent on the number of active NICs, i.e.,

$$y^i = \begin{cases} 1, \text{ if } \; N_{port}^i > 0 \\ 0, \text{ otherwise} \end{cases} \tag{23}$$

5) Robust Problem Formulation: Given the traffic prediction intervals and the assumption discussed above, our robust problem of network slice reconfiguration that minimizes the power consumption is formulated as follows:

$$\min_{\boldsymbol{x},\boldsymbol{z}} \sum_i \widetilde{P}^i \tag{24}$$

$$s.t. \; (10) - (23) \tag{25}$$

$$x_i(f_m^k) \in \{0,1\}, \forall i \in V, k \in \mathcal{K}, m \in \{1, \cdots, l_k\} \tag{26}$$

$$z_{p,q}(e_m^k) \in \{0,1\}, \forall (p,q) \in E, k \in \mathcal{K}, m \in \{1, \cdots, l_k - 1\} \tag{27}$$

Because the parameter $\widetilde{r}_t^k$ is a random variable that falls in the prediction interval, this problem is therefore a robust optimization problem. In particular, the term P^i in the objective function is uncertain since the allocated resource $\widetilde{c}_{allc}^i$ depends on the uncertain parameter $\widetilde{r}_t^k$. In addition, for the same reason, constraints (16) and (17) are uncertain constraints as well.

IV. The Robust Optimizer

In this section, we present the robust optimizer in the following steps: 1) First, we transform the robust problem into an MILP by linearization technique; 2) Second, the robust counterpart is derived; 3) Finally, the robust counterpart is solved by the MIP solver.

A. The Linearization of the Robust Problem

The robust problem formulated in the previous section involves one nonlinear term, i.e., (23). In this subsection, we will transform this nonlinear term into linear thus to facilitate the optimization procedure of our problem. To represent the nonlinear relation of $y^i = 1$ if $N_{port}^i > 0$ in (23), we equivalently transform it to:

$$N_{port}^i \le M^i y^i, y^i \in \{0,1\}, \tag{28}$$

where M^i is a large positive constant which guarantees the relationship between N_{port}^i and y^i. To reduce the feasible region of our problem as much as possible, M^i is set to the total number of NICs of server i. Please note that after this transformation, y^i becomes an optimization variable. By using this method, the nonlinear objective in (24) can be replaced by the following objective with the extra constraints in (28):

$$\widetilde{P}^i = P^i_{cpu} + P^i_{NIC} + y^i N^i_{port}, \tag{29}$$

B. Deriving the Robust Counterpart

The *robust counterpart* is a tractable deterministic equivalence of the robust problem. To derive the *robust counterpart* of problem (24), we need to transform our problem into the standard robust problem [13]. Thus, we first eliminate the uncertainty in the objective function by introducing an artificial scalar variable ρ. In this way, the original robust problem is therefore transformed into the following:

$$\min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\rho} \rho \tag{30}$$

$$s.t. \sum_i \widetilde{P}^i \le \rho, \tag{31}$$

$$(10) - (23), (26) - (29) \tag{32}$$

By using the transformation process in [13], the robust counterpart is derived as follows:

$$\min_{\boldsymbol{x},\boldsymbol{z},\boldsymbol{y},\boldsymbol{u},\boldsymbol{w},\rho} \rho \tag{33}$$

$$\begin{aligned} s.t. \sum_i \frac{P^i_{max} - P^i_{idle}}{c_i} \sum_k \sum_m \beta_c \theta(f^k_m) \\ \cdot [\bar{r}^k_t x_{i,k}(f^k_m) + \hat{r}^k_t u_{i,k}(f^k_m)] \le \rho \end{aligned} \tag{34}$$

$$\begin{aligned} \sum_{k \in \mathcal{K}} \sum_{m=1}^{l_k - 1} \beta_b [z_{p,q}(e^k_m) \bar{r}^k_t (f^k_m) + w_{p,q}(e^k_m) \hat{r}^k_t (f^k_m)] \\ \le b_{pq}, \forall (p,q) \in E \end{aligned} \tag{35}$$

$$\sum_k \sum_m \beta_c \theta(f^k_m) [\bar{r}^k_t x_{i,k}(f^k_m) + \hat{r}^k_t u_{i,k}(f^k_m)] \le c_i, \forall i \in V \tag{36}$$

$$-u_{i,k}(f^k_m) \le x_{i,k}(f^k_m) \le u_{i,k}(f^k_m) \tag{37}$$

$$-w_{p,q}(e^k_m) \le z_{p,q}(e^k_m) \le w_{p,q}(e^k_m) \tag{38}$$

$$(10) - (13), (26) - (29) \tag{39}$$

This problem is a Mixed Integer Linear Programming (MILP) and can be efficiently solved by the commercial MIP solvers, such as gurobi, CPLEX, .etc.

V. NUMERICAL RESULTS

In this section, we conduct numerical simulations to verify the effectiveness of our proposed predictor-optimizer framework. We first give the network parameters as well as the hyper-parameters of the predictors. After that, we present our simulation results and discussions.

A. Simulation Settings

The topology of the substrate network is the same as that in [11], which is widely used in network slicing simulations. The VNF set of the system contains 7 VNFs, and the compression ratio of each VNF is uniformly distributed in $[0.8, 1.2]$. The VNF capability of the nodes is randomly set to contain 2 to 4 VNFs. The capacity of each node is randomly generated in $[40, 200]$ units. Similarly, the bandwidth of each physical link is randomly generated in $[10, 200]$ units. The SFC of each network slice is a random VNF chain which contains 3 to 5 VNFs. The aggregated traffic demand in each slice is sampled from the real traffic which is gathered from the backbone network [14].

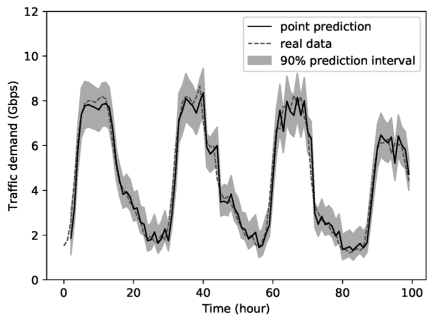

Fig. 1. 90% prediction intervals on 100 steps of test data

We use TensorFlow to build our prediction interval-oriented predictor for each network slice. Each predictor is comprised of 201 point predictors, each of which is a stacked LSTM network. 200 of them are used to evaluate the true regression mean and the remaining one is used to estimate the variance of the residual. Each predictor is trained by 50% of the historical traffic data of the corresponding network slice. The rest of the data are equally divided into two parts, which are used for evaluation and testing respectively. In addition, the optimization models are solved on a PC with an Intel i7-7500U CPU and 16GB RAM, applying the python MIP modeling package and gurobi Version 9.0.1.

B. Numerical Results

We use the predictor to predict 100 steps of the traffic demand in one slice with 90% prediction interval. Both the point predictions and the prediction intervals are shown in Fig. 1. As can be seen from the results, the prediction intervals can cover nearly 100% of the real traffic data, while the point predictions become less accurate when the traffic fluctuates heavily. Furthermore, we can observe that as the fluctuation of traffic becomes flatter, the width of the prediction interval decrease accordingly, and vice versa. Therefore, this result demonstrates that the width of the prediction interval can be automatically adjusted according to the traffic demand variations. Consequently, this property gives us the flexibility to make trade-offs between resource cost and the achieved robustness.

To verify the performance of the predictor-optimizer framework, we plot the reconfiguration results under different confidence levels ranging from 0.1 to 0.9 in Fig. 2. For comparison purpose, the reconfiguration results of point prediction and the handcrafted 1-std uncertainty set [5] are also shown in this figure, which correspond to the results of the confidence levels of 0.0 and 1.0 respectively. The results are averaged in 100 time intervals to avoid randomness.

Please note that Fig. 2 has two y-axes. The left y-axis is used to plot the *demand violation ratio* (R_v, defined in (40)),

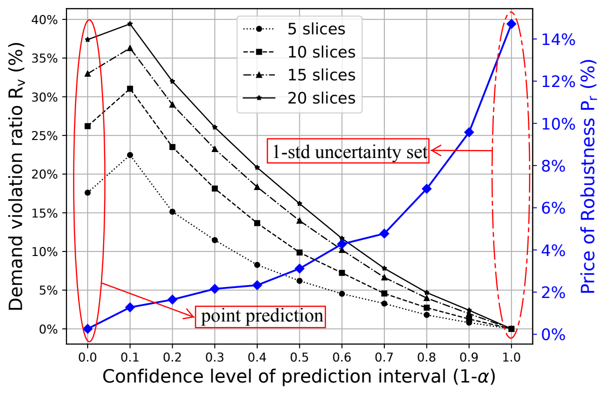

Fig. 2. Demand violation ratio and price of robustness *vs.* confidence level. The results of point prediction and handcrafted 1-std uncertainty set correspond to the confidence levels of 0.0 and 1.0, respectively.

which represents the number of time intervals of resource under-provisioning. The right y-axis is used to plot *price of robustness* (P_r, defined in (41)), which indicates the added energy consumption due to robustification compared with the energy consumed by real traffic demands. Formally, these two performance metrics are respectively defined as:

$$R_v = \frac{\text{number of violated time intervals}}{\text{total time intervals}} \times 100\%, \qquad (40)$$

$$P_r = \frac{\rho - \rho^*}{\rho^*} \times 100\%, \qquad (41)$$

where ρ^* is the value of the objective value for real traffic demands, and ρ is the objective value of the predictor-optimizer framework. Please note that the results of P_r are obtained by evaluating the scenario with 20 network slices.

First, from Fig. 2, we find that R_v decreases with the confidence level as we expected. Furthermore, this result also reveals that the violation ratio of the prediction interval is worse than that of point prediction when the confidence level is too small. Second, we can observe that R_v increases with the number of network slices. This is because the increase of network slices intensifies resource competition, leading to an increased risk of demand violation. Third, this result also shows that by using the prediction interval as the uncertainty set for the subsequent RO, the price of robustness P_r of our proposed framework is much lower than that of the handcrafted uncertainty set. This is due to the handcrafted uncertainty set is too conservative, leading to unnecessary resource provisioning. By shifting $(1-\alpha)$ from 1.0 to 0.9, we can decrease P_r about 7% at a cost of increasing R_v about 2.5%, which is possible in the slices with best-effort guaranteed SLA. In this way, we can flexibly achieve a trade-off between the robustness and the energy consumption.

VI. Conclusions

In this paper, we propose a novel robust network slice reconfiguration framework, which jointly leverages prediction interval and robust optimization. First, by exploiting LSTM and bootstrap method, we design a predictor that generates prediction intervals for the traffic demands per slice. Then we use the prediction interval as the uncertainty set to formulate the network slice reconfiguration problem to an RO problem, which is solved by commercial solver after some transformations. Finally, the performance of our framework is evaluated on real traffic data, and the numerical results demonstrate the proposed predictor-optimizer can provide adjustable trade-off between robustness and energy consumption.

This work provides a framework which combines prediction interval and robust optimization to perform inter-slice reconfiguration. Actually, this framework can be served as a uniform approach for resource allocation in wireless networks with uncertainty and service quality requirements. Therefore, this work gives guidance and insights for constructing robust mobile networks.

References

[1] Y. Sun, S. Qin, G. Feng, L. Zhang, and M. A. Imran, “Service provisioning framework for ran slicing: user admissibility, slice association and bandwidth allocation,” *IEEE Transactions on Mobile Computing*, 2020.

[2] Y. Sun, G. Feng, L. Zhang, P. V. Klaine, M. A. Iinran, and Y.-C. Liang, “Distributed learning based handoff mechanism for radio access network slicing with data sharing,” in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–6.

[3] F. Wei, G. Feng, Y. Sun, Y. Wang, and Y. Liang, “Dynamic network slice reconfiguration by exploiting deep reinforcement learning,” in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[4] T. Subramanya and R. Riggio, “Machine learning-driven scaling and placement of virtual network functions at the network edges,” in *2019 IEEE Conference on Network Softwarization (NetSoft)*, 2019, pp. 414–422.

[5] A. Baumgartner, T. Bauschert, A. A. Blzarour, and V. S. Reddy, “Network slice embedding under traffic uncertainties — A light robust approach,” in *2017 13th International Conference on Network and Service Management (CNSM)*, nov 2017, pp. 1–5.

[6] R. Wen, G. Feng, J. Tang, T. Q. S. Quek, G. Wang, W. Tan, and S. Qin, “On Robustness of Network Slicing for Next-Generation Mobile Networks,” *IEEE Transactions on Communications*, vol. 67, no. 1, pp. 430–444, jan 2019.

[7] V. Sciancalepore, X. Costa-Perez, and A. Banchs, “RL-NSB: Reinforcement Learning-Based 5G Network Slice Broker,” *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1543–1557, aug 2019.

[8] M. Yan, G. Feng, J. Zhou, Y. Sun, and Y.-C. Liang, “Intelligent Resource Scheduling for 5G Radio Access Network Slicing,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7691–7703, jun 2019.

[9] A. Khosravi, S. Nahavandi, D. Creighton, and A. F. Atiya, “Comprehensive review of neural network-based prediction intervals and new advances,” *IEEE Transactions on Neural Networks*, vol. 22, no. 9, pp. 1341–1356, 2011.

[10] K. Papagiannaki, N. Taft, Z. L. Zhang, and C. Diot, “Long-term forecasting of Internet backbone traffic,” *IEEE Transactions on Neural Networks*, vol. 16, no. 5, pp. 1110–1124, 2005.

[11] F. Wei, G. Feng, Y. Sun, Y. Wang, S. Qin, and Y. Liang, “Network slice reconfiguration by exploiting deep reinforcement learning with large action space,” *IEEE Transactions on Network and Service Management*, 2020. [Online]. Available: https://doi.org/10.1109/TNSM.2020.3019248

[12] M. Dayarathna, Y. Wen, and R. Fan, “Data center energy consumption modeling: A survey,” *IEEE Communications Surveys and Tutorials*, 2016.

[13] A. Ben-Tal, L. E. Ghaoui, and A. Nemirovski, *Robust optimization*, 2009.

[14] P. Cortez, M. Rio, M. Rocha, and P. Sousa, “Multi-scale Internet traffic forecasting using neural networks and time series methods,” vol. 29, no. 2, pp. 1–13, 2012.