Provably Efficient Online End-to-end SLA Decomposition

***Abstract*—Network slicing plays a crucial role in realizing 5G advancements, bringing diverse SLA requirements related to latency, throughput, and reliability. Since network slicing is typically implemented in an end-to-end (e2e) manner across multiple domains—such as access, transport, and core networks in mobile systems—it is essential to allocate the total SLA budget across participating domains based on the heterogeneity among these domains accordingly. Proper decomposition could lead to resource-efficient e2e SLA satisfaction. However, domain conditions (on which SLA decomposition relies) are in practice not known in advance, and must be learned through feedback samples in an online manner. To address this gap, we first give a theoretical formulation for this problem in both online and offline settings. In the online setting, we give a novel Bayesian Optimization-based algorithm and show that it is provably efficient in terms of performance metrics we define in the theoretical formulation. We further conduct experiments through simulations and real-world testbeds to demonstrate the effectiveness of out algorithm over other baselines.**

***Index Terms*—Network slicing; service level agreement; Bayesian Optimization; online learning; E2E orchestration**

I. Introduction

The rise of 5G technology has brought new possibilities for networks, enabling a variety of services with different and strict Service Level Agreement (SLA) requirements. These SLAs, which often focus on latency, throughput, and reliability, are crucial for ensuring the quality of service (QoS) that users expect. Network slicing, a key feature of 5G, allows multiple virtual networks to be created on a shared physical infrastructure, each designed to meet specific SLA needs. However, implementing network slicing usually involves multiple domains—such as the Radio Access Network (RAN), transport, and core networks—each with its own unique characteristics and limitations. As a result, efficiently dividing the total SLA budget across these different domains becomes a major challenge.

End-to-end (e2e) SLA decomposition [6], [13], [14] is a key process in this context. It involves breaking down the overall SLA into smaller, domain-specific SLAs that together meet the e2e requirements. Proper decomposition not only ensures that the e2e SLA is met but also makes the best use of resources across the network. However, this task is not easy. Domain conditions, which are essential for effective SLA decomposition, are often unknown in advance and must be learned through real-time feedback. This dynamic and uncertain environment requires an online learning approach, where the algorithm continuously adapts to unknown network conditions without prior information to make the best decomposition decisions.

Existing work on SLA decomposition has mainly focused on offline settings, where data is collected in advance to train models that predict domain-level SLA acceptance probabilities. While these approaches have their strengths, they are limited by the reliance on pre-collected data. In contrast, our work addresses the problem in an online setting, where the algorithm learns to make better decomposition decisions through continuous interaction with the network, without needing prior knowledge of domain-specific control policies.

In this paper, we present a new Bayesian Optimization (BO)-based algorithm for online e2e SLA decomposition. Our approach is provably efficient, meaning it guarantees both low cost and SLA constraint violations as it learns from feedback over time. We also demonstrate the practical effectiveness of our algorithm through extensive simulations and real-world testbed evaluations.

A. Challenges

1) Network dynamics: The main challenge in SLA decomposition rises from the dynamic and unknown network conditions. Without knowing the network conditions, blindly decomposing e2e SLA into static domain-level values could incur highly sub-optimal performance. However, since network conditions are highly changing over time, there does not exist a universal default decomposed SLAs that work well under diverse network conditions. Similarly, supervised/offline Machine Learning also suffers from such distribution shift: models/experience learned from one network condition may not work well under a highly different one.

2) Bayesian Optimization for Network Optimization: As a powerful and suitable framework, BO has been widely applied to a number of problems on autonomous network control, including power control [17], [21], task offloading, resource allocation [21], and mobility (handover) management [5].

As it is well-known that BO in generally suffers an undesirable computation complexity in the high-dimensional regime [17] (i.e., the number of parameters to control is large), BO is found practical only when there are not too many control variables (e.g., less then 20). Instead in our formulation, since the decision space of the orchestrator is always the decomposed SLAs, the BO algorithm does not suffer high computation complexity even if the number of control variables in the domains is large.

B. Contributions

Our key contributions are as follows:

1) Algorithmic Contributions:

- To our best knowledge, we are the first to give a *theoretical formulation* for online SLA decomposition, and design *provably efficient* BO-based algorithm. It has further advantages such as 1) being agnostic to the domains' own control contexts and 2) enjoying better theoretical guarantee by leveraging domain-level feedback.

2) Realworld Testbed implementation and evaluation:

3) Large scale simulations:

II. Background and Motivation

A. Background - What is SLA Decomposition

While in 5G network slicing, an SLA requirement is defined in an end-to-end manner, it is really important that each domain is contributing to the end-to-end SLA with the right proportion, so that the end-to-end SLA can be satisfied (with desirable total resource cost). Therefore, SLA decomposition becomes an emerging technology to study. To be specific, an end-to-end domain orchestrator decomposes end-to-end SLA into domain-level ones based on information/observations about the capability/resource availability of each domain. Note that this can be done even in a repeated and interactive form: The orchestrator can keep monitoring the domain information, collecting feedback samples, and improving decomposed SLAs.

While being promising, this line of research is still relatively open and exhibits challenges. System-wise, it is still open regarding the *transparency* between the domain controllers and end-to-end orchestrator (i.e., what information is shared between them). We refer the readers to [9] for a good survey.

Algorithm-wise, it is non-trivial to find good decomposed domain-level SLAs, given the heterogeneity (e.g., resource and non-stationary) among the domains (which may not even be known to the orchestrator as prior information).

B. What is Black-Box Optimization

Black-box optimization is a type of optimization problem/fashion where the objective function is unknown or lacks an explicit mathematical form, meaning it can only be evaluated by running experiments or simulations. The function is treated as a "black box" because its internal mechanism is not accessible, and only inputs and the corresponding (noisy) outputs can be observed. Common methods for such problems include random search [2], [18], Bayesian Optimization, evolutionary algorithms [1], [10], and more.

C. Motivation

We give an real-world example to motivate SLA decomposition. Say our task is live streaming (i.e., real-time video transmission), in which satisfying the e2e latency has the highest priority, so that the customers enjoy a smooth watching experience. Say the e2e latency target is 90 ms, and the network has three segments/domains, namely, transport, Core, and RAN. Then the e2e latency would be the sum of the three decomposed domain-level SLAs. Below we justify the importance of setting proper domain-level SLAs from the perspectives of both SLA satisfaction and cost efficiency.

First of all, note that domains could have heterogeneous capability in terms of providing low latency. For example, at some moment, the transport network could have heavy load from other jobs, and the lowest latency it could provide for the live streaming job is 60 ms. In this case, if we simply decompose e2e SLA uniformly, i.e., $90/3 = 30$ ms for each domain, then the actual e2e latency would be at least $60+2\times 30 = 120$ ms, which is an about $(120-90)/90 \approx 33\%$ violation of the e2e SLA and could negatively impact users' experience, due to the transport domain being a bottleneck. This simple example demonstrates that in order to satisfy the e2e SLA, it is crucial to determine decomposed SLAs according to the capacity of each domain.

In the same example, now it is clear that the RAN and Core networks should take over the remaining $90 - 60 = 30$ ms latency in order to satisfy the e2e SLA, and we assume that these two domains have sufficient capability to provide very low latency. Then, e2e SLA satisfaction is not a problem, however, the further decomposition between these two domains may significantly affect the total cost consumption, which should be minimized: For example, comparing two different decomposed SLAs $(20\text{ms}, 10\text{ms})$ and $(10\text{ms}, 20\text{ms})$, it is possible that the cost consumption in RAN network is not sensitive to these two different values (i.e., RAN network needs very little additional resource to reduce its domain-level latency from 20 ms to 10 ms), while this 10 ms gap could incur huge resource increment in Core network, which may affect the quality of other services in the network as the total resource is limited. Therefore, beyond ensuring e2e SLA, the decomposition should further seek lowest total cost.

III. Design

A. Optimization Problem Formulation

B. System Model and Offline Problem

We design an algorithm deployed at the end-to-end orchestrator for coordinating domain controllers through domain-level SLAs to satisfy the end-to-end SLA whiling minimizing the total cost. Throughout this work, we work with latency as the SLA of interest, and our framework is readily applied to other SLAs with the same (additive) composition rule.

We let D denote the number of domains in an end-to-end network, L denote the end-to-end SLA, and $x = (x^1, \ldots, x^D) \in (\Re_+)^D$ be a SLA decomposition. Decomposition x incurs domain-level latencies $f^1(x^1), \ldots, f^D(x^D)$ and costs $g^1(x^1), \ldots, g^D(x^D)$.

Taking latency decomposition in a 5G network (with $D = 3$ domains, namely, Radio Access Network, Transport, and Core) as an example, after determining the decomposed SLA, say $(2, 3, 4)$ (in milliseconds), the end-to-end orchestrator broadcasts the decomposed latencies to the corresponding domains. Each domain takes the requested latency (e.g., 2 ms for the first domain), performs its own management policy (which is unknown to the orchestrator) based on the requested latency

(and the stochastic domain network condition), resulting in the true latency and resource cost (which is some manually-defined function on allocated resources of interest).

Given the system model and the objective, we first formulate them as an offline optimization problem.

In the offline setting (where domain latency functions $f^1, \dots, f^D$ and cost functions $g^1, \dots, g^D$ are all known), the objective is to find the SLA decomposition such that the end-to-end SLA is satisfied while the total resource cost is minimized. By definition, the end-to-end SLA (e.g., latency) is the sum of all domain-level SLAs, and the total cost is assumed to be also the sum of all domain-level costs. Therefore, the offline optimization can be written as

$$\min_{x=(x^1,\dots,x^D)\in(\Re_+)^D} \sum_{i=1}^{D} g^i(x^i), \text{s.t. } \sum_{i=1}^{D} f^i(x^i) \le L. \quad (1)$$

Assumption 1. *Problem* (1) *is feasible, i.e., there exists some* $x \in (\Re_+)^D$ *such that* $\sum_{i=1}^{D} f^i(x^i) \le L$.

While this offline formulation captures our objective and constraints, we may not readily find the optimal solution of interest from it, as the maps from decomposed SLAs to latencies and costs (which are determined by the unknown domain management policies) are unknown a priori. Instead, these functions must be learned through trial and error in an online manner. Due to this need, we further propose an online formulation in the following subsection.

C. Online Problem

In the online setting, the functions are unknown, and any algorithm has to "learn" those functions through interactions. Specifically, in each round t, the algorithm decides the decomposed SLAs $x_t = (x_t^1, \dots, x_t^D)$. Each domain i takes requested latency x_t^i, performs domain management based on both x_t^i and stochastic domain network condition. As a result, the learning algorithm observes (noisy) feedback $y_t^i = f^i(x_t^i) + \epsilon_t^{i,f}$ and $c_t^i = g^i(x_t^i) + \epsilon_t^{i,g}$, where $\epsilon_t^{i,f}$ and $\epsilon_t^{i,g}$ are conditionally zero-mean sub-Gaussian noise with sub-Gaussian norms $R^{i,f}$ and $R^{i,g}$, respectively for each domain $i = 1, \dots, D$.

The performance metrics we typically use for the online algorithms are regret and constraint violation (with respect to the number of interactions N), denoted by R_N and V_N respectively, and defined as

$$R_N := \sum_{t=1}^{N} \sum_{i=1}^{D} \left(g^i(x_t^i) - g^i(x_*^i) \right), \quad (2)$$

$$V_N := \sum_{t=1}^{N} \left(\sum_{i=1}^{D} f^i(x_t^i) - L \right)^+, \quad (3)$$

where $x_* = (x_*^1, \dots, x_*^D)$ is the optimizer of optimization problem (1), and $x^+ := \max\{x, 0\}, \forall x \in \Re$. R_N quantifies the cumulative excessive cost consumption and V_N reflects the cumulative (hard)[1] SLA constraint violation. A good learning algorithm in the online setting is expected to ensure both small regret and constraint violation, implying that it converges to the optimal offline solution very soon.

In the online setting, the main challenge lies in the tradeoff between exploration and exploitation. To be more specific, since the learning algorithm starts without knowing the underlying functions, it has to try out some (noisy) samples that help reduce the uncertainty on the unknown functions (exploration), at the sacrifice of choosing the "best" decision based on the current knowledge of the environment (exploitation). A good learning algorithm should strike a balance between them.

Theoretically, a non-trivial algorithm should ensure that both regret R_N and constraint violation V_N is sub-linear in the number of interactions N (i.e., $R_N = o(N)$ and $V_N = o(N)$), meaning that the the algorithm is converging to the offline optimal solution, and the step-average regret and constraint violation both converge to zero, i.e., $\lim_{N\to\infty} \frac{R_N}{N} \to 0$ and $\lim_{N\to\infty} \frac{V_N}{N} \to 0$.

Moreover, motivated by a piratical consideration from the real networks, in this work we strictly generalize the standard one-decision-one-sample setup, and allow heterogeneous sample querying cycle from the domains. In particular, for the domain i, the sample querying cycle τ^i means that one noisy feedback (described above) is generated and observed by the learning algorithm every τ^i unit of time (e.g., milliseconds). Meanwhile, the orchestrator is allowed to make a new decision (i.e., proposing new decomposed SLAs) every τ^{DM} unit of time. In this work, we assume $\tau^{\text{DM}} > \max_i \tau^i$, i.e., at least one new sample is collected from each domain when a new decision is made. Between two interactions, since the decisions are not changed, all the newly collected samples are evaluated at the same point (which is the latest decision).

Under this new formulation (in particular when the learning algorithm has the flexibility to choose how frequently a new decision is made), a better fit for the performance metrics is regret (and constraint violation) within *time* T:

$$R(T) := \tau^{\text{DM}} \cdot R_{T/\tau^{\text{DM}}} = \tau^{\text{DM}} \cdot \sum_{t=1}^{T/\tau^{\text{DM}}} \left(\sum_{i=1}^{D} g^i(x_t^i) - g^i(x_*^i) \right), \quad (4)$$

$$V(T) := \tau^{\text{DM}} \cdot V_{T/\tau^{\text{DM}}} = \tau^{\text{DM}} \cdot \sum_{t=1}^{T/\tau^{\text{DM}}} \left(\sum_{i=1}^{D} f^i(x_t^i) - f^i(x_*^i) \right)^+. \quad (5)$$

To see why this new definition makes sense, for any fixed decision-making cycle τ^{DM}, the number of iterations is T/τ^{DM}, however the time interval between two decisions are now varying (which is namely τ^{DM}). Therefore, finally the T/τ^{DM}-round regret and constraint violation are scaled by the

[1] "Hard" means that we do not allow a negative violation to cancel out a positive one in another round, which is a stronger guarantee that the "soft" constraint. Any guarantee on hard constraint apply to soft constraint, but not vice versa.

length of the time interval to reflect that now the "weight" of a new decision is (proportional to) τ^{DM}.

Learning is not possible with arbitrary functions. To get meaningful results, we put some "regularity assumption" on the unknown functions to ensure they are learnable.

Assumption 2. *It is assumed that $f^i \in \mathcal{H}_{k^{i,f}}, g^i \in \mathcal{H}_{k^{i,g}}$, where $k^{i,f}$ (respectively $k^{i,g}$) is some symmetric, positive-semidefinite kernel function and $\mathcal{H}_{k^{i,f}}$ (respectively $\mathcal{H}_{k^{i,G}}$) is corresponding reproducing kernel Hilbert space (RKHS). Moreover, we assume that $\|f^i\|_{k^{i,f}} \leq B^{i,f}$ and $\|g^i\|_{k^{i,g}} \leq B^{i,g}$, where $\|\cdot\|_{k^{i,f}}$ and $\|\cdot\|_{k^{i,g}}$ are the respective norms induced by the inner product in the corresponding RKHS.*

Remark 1. *Another common assumption in BO is the compact decision space $\mathcal{X}$, which is the case in this paper ($(\Re_+)^D$).*

Remark 2. *In both the offline and online formulation, the absolute performance (i.e., how much cost is incurred and the corresponding SLA satisfaction) depends on the underlying domain management policies, which are implicitly encoded into functions $\{f^i, g^i\}_{i=1}^D$ and assumed to be given and arbitrary. The objectives in formulations aim at approaching the best solution,* given the fixed domain management policies*, no matter how (un)reasonable they are.*

IV. Gaussian Process

In this section, we provide the basics of Gaussian Process (GP).

We use GP as the surrogate model for the algorithm design (to learn each black-box function f^i). Conditioned on the set of all observations in the past $H_t^i := \{x_s^i, y_s^i\}_{s\in[t]}$, the posterior distribution for f^i is $\mathcal{GP}(\mu_t^{i,f}(\cdot), k_t^{i,f}(\cdot,\cdot))$, where

$$\mu_t^{i,f}(x) := k_t^{i,f}(x)^\top (K_t^{i,f} + \lambda I_t)^{-1} Y_t^{i,f}, \tag{6}$$

$$k_t^{i,f}(x,x') := k^{i,f}(x,x') - k_t^{i,f}(x)^\top (K_t^{i,f} + \lambda I_t)^{-1} k_t^{i,f}(x'), \tag{7}$$

in which $k_t^{i,f}(x) := [k^{i,f}(x_1^i, x), \ldots, k^{i,f}(x_t^i, x)]^\top$, $K_t^{i,f} = [k^{i,f}(x_u^i, x_v^i)]_{u,v\in[t]}$, I_t is the t-dimensional identity matrix, and $Y_t^{i,f}$ is the noisy observation vector $[y_1^{i,f}, \ldots, y_t^{i,f}]^\top$. We also define $\sigma_t^{i,f}(x)^2 = k_t^{i,f}(x,x)$. Under the concrete context of latency, vector $Y_t^{i,f}$ contains all (noisy) latency observations up to the t-th interaction from domain i, $\mu_t^{i,f}(x)$ is the estimated latency if we specify x as a domain-level SLA to domain i, and $\sigma_t^{i,f}(x)^2$ represents the uncertainty at x. We let $K_A^{i,f} := [k^{i,f}(x,x')]_{x,x'\in A}$ for any finite $A \in \mathcal{X}$, then we use $\gamma_t(K^{i,f}, \mathcal{X})$ to denote the information gain, which is defined as $\gamma_t(K^{i,f}, \mathcal{X}) := \max_{A\subseteq\mathcal{X}:|A|=t} \frac{1}{2}\ln|I_t + \lambda^{-1}K_A|$ and plays an important role in the theoretical analysis. The maximum information gain depends on both the kernel $k^{i,f}$ and domain $\mathcal{X}$ (which is $(\Re_+)^D$ in this work), and will be simplified as γ_t whenever the context is clear. For simplicity, we assume all unknown functions share the same kernel, but our framework can readily handle the case in which the kernel functions are not identical.

In the constrained BO problem, the learning algorithm uses another surrogate model for the function g^i calculated in the same way, based on another set of observations $\widetilde{H}_t^i = \{x_s^i, c_s^i\}_{s\in[t]}$, i.e.,

$$\mu_t^{i,g}(x) := k_t^{i,g}(x)^\top (K_t^{i,g} + \lambda I_t)^{-1} Y_t^{i,g}, \tag{8}$$

$$k_t^{i,g}(x,x') := k^{i,g}(x,x') - k_t^{i,g}(x)^\top (K_t^{i,g} + \lambda I_t)^{-1} k_t^{i,g}(x'). \tag{9}$$

Algorithm 1 Black-box Cost-aware End-to-end SLA Decomposition

Input: end-to-end SLA $L > 0$, kernel k, the number of domains D, decision-making cycle τ^{DM}, feedback querying cycles $\tau^1, \ldots, \tau^D < \tau^{\mathrm{DM}}$, noise sub-Gaussian norms $\{R^{i,f}, R^{i,c}\}_{i=1}^D$, horizon T, failure probability $\delta \in (0,1)$
Define: the number of interactions $N = T/\tau^{\mathrm{DM}}$, $\beta_t^{i,f} := B^{i,f} + \frac{R^{i,f}}{\sqrt{\mathrm{DM}/\tau^i}}\sqrt{2\left(\gamma_t + 1 + \ln(2D/\delta)\right)}$, and $\beta_t^{i,g} := B^{i,g} + \frac{R^{i,g}}{\sqrt{\mathrm{DM}/\tau^i}}\sqrt{2\left(\gamma_t + 1 + \ln(2D/\delta)\right)}$
for $t = 1, \ldots, N$ **do**
  From each domain i, collect $\tau^{\mathrm{DM}}/\tau^i$ independent noisy samples evaluated at x_t^i, and the averages $(\overline{y}_t^i, \overline{c}_t^i)$
  Update surrogate models as in Eqs. (6), (7), (8), and (9)
  Find new decomposed SLAs $x_t = (x_t^1, \ldots, x_t^D) \in \operatorname{argmin}_x \sum_{i=1}^D (\mu_t^{i,g}(x^i) - \beta_t^{i,g}\cdot\sigma_t^{i,g}(x^i))$ subject to $\sum_{i=1}^D (\mu_t^{i,f}(x^i) - \beta_t^{i,g}\cdot\sigma_t^{i,f}(x^i)) - L \leq 0$, and apply them to all domains
end for

Remark 3. *When all the functions share the same kernel, all the posterior variances $\{k_t^{i,f}(\cdot,\cdot), k_t^{i,g}(\cdot,\cdot)\}_{i=1}^D$ are identical.*

V. Algorithm Design and Theoretical Analysis

A. Algorithm Overview

Our algorithm design leverages the framework for constrained BO in [19]. The principle of the design is keeping solving the offline problem, except that the unknown true functions to learn are replaced with some optimistic/pessimistic estimates built upon noisy observations. Intuitively, as more and more feedback is collected overtime, the estimates are getting improved and "closer" to the ground truth, and hence the decisions are converging to the optimal ones. To tackle our problem, we generalize the original design in [19] to the case in which one single objective function (and one cost function) is replaced with an additive composition of multiple base functions (which corresponds to multiple domains) and (noisy) observations can be queried from each of them with potentially different frequencies. The full pseudo-code is given in Algorithm 1.

Given the horizon T and decision-making interval τ^{DM}, the algorithm performs T/τ^{DM} rounds of interactions. Before the t-th decision is made, there are $\tau^{\mathrm{DM}}/\tau^i$ newly collected samples from domain i, all of which are (independent) noisy feedback evaluated at x_{t-1}^i. The samples are averaged and then used for updating the surrogate model. The averaging not only makes the analysis simple, but also saves computational cost (compared to adding them one by one) [16].

B. Theoretical Analysis

We first the state the concentration results, on which the final regret/constraint violation analysis replies heavily. It roughly says how close the surrogate model is to the ground truth.

Lemma 1 (Concentrations)**.** *With probability at least* $1-\delta$*, it holds for any* $t \in [T/\tau^{DM}]$ *and domain* i *that* $|f^i(x) - \mu_t^{i,f}(x)| \le \beta_t^{i,f} \cdot \sigma_t^{i,f}(x)$ *and* $|g^i(x) - \mu_t^{i,g}(x)| \le \beta_t^{i,g} \cdot \sigma_t^{i,g}(x)$*, where* $\beta_t^{i,f} := B^{i,f} + \frac{R^{i,f}}{\sqrt{DM/\tau^i}}\sqrt{2\left(\gamma_t + 1 + \ln(2D/\delta)\right)}$ *and* $\beta_t^{i,g} := B^{i,g} + \frac{R^{i,g}}{\sqrt{DM/\tau^i}}\sqrt{2\left(\gamma_t + 1 + \ln(2D/\delta)\right)}$.

Proof of Lemma 1. For any sub-Gaussian norm $R > 0$ and number of samples $n \in \mathbb{N}^+$, the sum of n independent R-sub-Gaussian random variables is $\frac{R}{\sqrt{n}}$-sub-Gaussian. The claimed results are then directly implied by Lemma 2.4 in [19] by replacing the sub-Gaussian norm. □

Another key lemma we need is also a standard argument in the BO literature, which bounds the sum of total deviation in the posterior distribution and formally stated below.

Lemma 2 (Lemma 4 in [4])**.** *For any domain* i*, the sequence* $x_1^i, \dots, x_N^i$ *chosen by Algorithm 1 satisfies that*

$$\sum_{t=1}^{N} \sigma_t^{i,f}(x_t^i) \le 2\sqrt{(N+2)\gamma_N}, \tag{10}$$

$$\sum_{t=1}^{N} \sigma_t^{i,g}(x_t^i) \le 2\sqrt{(N+2)\gamma_N}. \tag{11}$$

Theorem 1 (Theoretical Guarantee)**.** *With probability at least* $1-\delta$*, Algorithm 1 ensures both* $R(T) = O\left(\tau^{DM}\sqrt{T \cdot \gamma_{T/\tau^{DM}}} \cdot \sum_{i=1}^{D}\left(B^{i,f} + \frac{R^{i,f}}{\sqrt{\tau^{DM}/\tau^i}}\sqrt{\gamma_{T/\tau^{DM}}}\right)\right)$ *and* $V(T) =$

$$O\left(\tau^{DM}\sqrt{T \cdot \gamma_{T/\tau^{DM}}} \cdot \sum_{i=1}^{D}\left(B^{i,g} + \frac{R^{i,g}}{\sqrt{\tau^{DM}/\tau^i}}\sqrt{\gamma_{T/\tau^{DM}}}\right)\right).$$

Proof of Theorem 1. This proof is an adaptation from that of Theorem 4.3 in [19].

We will first bound regret in terms of the number of interactions (and constraint violation in the same way).

Let $r_t := \sum_{i=1}^{D}\left(g^i(x_t^i) - g^i(x_*^i)\right)$ be the instantaneous regret incurred in the t-th decision. It can be upper bounded as

$$\begin{aligned}
r_t &= \sum_{i=1}^{D}\left(g^i(x_t^i) - g^i(x_*^i)\right) \\
&\overset{(a)}{\le} \sum_{i=1}^{D}\left(\mu^{i,g}(x_t^i) + \beta_t^{i,g}\cdot\sigma_t^{i,g}(x_t^i)\right) \\
&\quad - \sum_{i=1}^{D}\left(\mu^{i,g}(x_*^i) - \beta_t^{i,g}\cdot\sigma_t^{i,g}(x_*^i)\right) \\
&\overset{(b)}{\le} \sum_{i=1}^{D}\left(\mu^{i,g}(x_t^i) + \beta_t^{i,g}\cdot\sigma_t^{i,g}(x_t^i)\right) \\
&\quad - \sum_{i=1}^{D}\left(\mu^{i,g}(x_t^i) - \beta_t^{i,g}\cdot\sigma_t^{i,g}(x_t^i)\right) \\
&= 2\cdot\sum_{i=1}^{D}\beta_t^{i,g}\cdot\sigma_t^{i,g}(x_t^i),
\end{aligned} \tag{12}$$

where step (a) is due to Lemma 1 and step (b) holds since

$$\sum_{i=1}^{D}(\mu_t^{i,f}(x_*^i) - \beta_t^{i,g}\cdot\sigma_t^{i,f}(x_*^i)) \le \sum_{i=1}^{D} f^i(x_*^i) \le L.$$

Now we can readily bound the cumulative regret (in terms of the number of interactions N) as

$$\begin{aligned}
R_N = \sum_{t=1}^{N} r_t &\le 2\cdot\sum_{t=1}^{N}\sum_{i=1}^{D}\beta_t^{i,g}\cdot\sigma_t^{i,g}(x_t^i) \\
&\le 2\cdot\sum_{i=1}^{D}\beta_N^{i,g}\sum_{t=1}^{N}\sigma_t^{i,g}(x_t^i) \\
&\le 4\cdot\sum_{i=1}^{D}\beta_N^{i,g}\sqrt{(N+2)\gamma_N}.
\end{aligned} \tag{13}$$

Finally, we arrive at the bound on $R(T)$ by replacing N and $\beta_N^{i,g}$ with their own definitions respectively, and multiplying R_N by τ^{DM}.

Similarly, we define $v_t := \left(\sum_{i=1}^{D} f^i(x_t^i) - L\right)^+$ as the instantaneous (hard) constraint violation incurred in the t-th decision. It can be bounded as

$$\begin{aligned}
v_t &\overset{(a)}{\le} \left(\sum_{i=1}^{D}\left(\mu_t^{i,f}(x_t^i) + \beta_t^{i,f}\cdot\sigma_t^{i,f}(x_t^i)\right) - L\right)^+ \\
&\le \left(\sum_{i=1}^{D}\left(\mu_t^{i,f}(x_t^i) - \beta_t^{i,f}\cdot\sigma_t^{i,f}(x_t^i)\right) - L\right)^+ + 2\beta_t^{i,f}\cdot\sigma_t^{i,f}(x_t^i) \\
&\overset{(b)}{\le} 2\beta_t^{i,f}\cdot\sigma_t^{i,f}(x_t^i),
\end{aligned} \tag{14}$$

where step (a) is from Lemma 1 and step (b) holds simply since

$$\sum_{i=1}^{D}\left(\mu_t^{i,f}(x_t^i) - \beta_t^{i,f}\cdot\sigma_t^{i,f}(x_t^i)\right) - L \le 0 \tag{15}$$

due to the algorithm design.

Lastly, we bound cumulative constraint violation $V(T)$ via same path from r_t to $R(T)$. □

VI. Testbed Implementation

A. Transport Network

We simulate the transport domain using Mininet software.[2] In the topology, we there are six hosts (with indexes from 1 to 6) and eight switches. Between each pair of hosts, there are multiple different paths via (some of) these switches. Between every host pair formed among hosts 3 to 6, we generate data flows with varying traffic rates as "background traffic". Now, among all the paths between hosts 1 and 2, the available remaining bandwidth is different, and consequently, so will be the latency between hosts 1 and 2 via those paths.

To generate trace data, we further generate data flow between hosts 1 and 2 via different paths. For each path, we measure the latency using *Two-Way Active Measurement Protocol* (TWAMP)[3] [11] and the available bandwidth using *Bandwidth Monitor NG*.[4]

Given any decomposed latency for the transport domain, the selected path will be the one with lowest available bandwidth among all paths whose measured latency is lower than the decomposed latency. The actual latency and cost would be those of the selected path, respectively. To understand this, the path selection principle would be selecting the "cheapest" one that can satisfy the required SLA.

B. RAN Network

C. Core Network

VII. Experimental Results

A. Testbed evaluation

B. Large scale evaluation - Simulation

We first show evaluation results on synthetic (i.e., some artificial functions). We choose three monotonically increasing functions to serve as f_1, f_2, f_3 (a higher decomposed latency incurs a higher actual latency from the domain) and three monotonically decreasing functions to serve as g_1, g_2, g_3 (a higher decomposed latency incurs a lower cost consumption from the domain).

To reflect the heterogeneity across the domains, we try to make these functions have diverse shapes. To be specific, for the latency functions f_1, f_2, f_3, we choose the logarithmic function (of form $w\log(x)$), exponential function (of form $w\exp(x)$), and quadratic function (of form wx^2), and for the cost functions, we choose the rational function (of form $\frac{w}{x+b}$), inversed quadratic function (of form w/x^2), and Gaussian function (of form $w\exp(-x^2)$), where w and b are some scaling/shifting factors and vary by function.

We include the follow two baselines to compare with our main algorithm (Algorithm 1):

- The end-to-end variant of Algorithm 1. The only difference from Algorithm 1 is that it treats the entire network as a (bigger) black-box function. I.e., in online interactions, it observes the end-to-end latency and total costs, without leveraging the fine-grained domain-level feedback. Mathematically, it directly builds surrogate models on $f' := f_1 + f_2 + f_3$ and $g' := g_1 + g_2 + g_3$.
- Explore-then-commit (ETC) algorithms. The high-level idea of ETC is very similar to that in [6]. That is, for any fraction $\alpha \in (0, 1)$ (which is typically small), in the first αT rounds, ETC plays uniformly random actions to collect samples, and in the remaining $(1 - \alpha)T$ rounds, ETC commits the action which is obtained by solving the offline problem in which the unknown true functions are replaced with surrogate models built upon samples from the uniform exploration stage (i.e., the first αT rounds). In the experiments we try $\alpha = 0.1$ and $\alpha = 0.2$.

By swapping the functions, we can get in total 6 combinations in terms of the latency function and the cost function. Due to the page limits and the similar results from these combinations, we only show the plots from one combination.

For each end-to-end latency SLA $L \in \{2.0, 2.5, \ldots, 15.0\}$ and each algorithm, we conduct 5 independent runs, and each run consists of $T = 5 \times 10^4$ rounds. The domain-level SLA is selected from $\{1.0, 1.5, \ldots, 6.0\}$ so the entire decision space is $\{1.0, 1.5, \ldots, 6.0\}^3$. From each run, we get the average end-to-end latency and cost over all T rounds. Finally in the plot, we show the average latency and cost from these 5 runs together with the error bar which reflects the standard deviation (which is resealed by some fixed constant in all results for readability).

Figure 1 shows both the average latency and cost under different SLAs. From the cost curves, we can see that our main algorithm enjoys lowest cost in most values of L, while in the latency curves, the one from the main algorithm almost coincides with the reference line, if it does not fall under, which means that it does a good job in satisfying the end-to-end SLA.

The comparison with the end-to-end variant demonstrates the importance of leveraging domain-level feedback, and that with ETC algorithms shows the superior performance of adaptive decision-making.

Since real-worlds network typically do not stay stationary for a long time, we also conduct the experiments with very few interaction rounds ($T = 100$) and show the results below in Figure 2.

In the few-observation scenario, we can see that our main algorithm does not strictly perform better than the ETC approach. On the other hand, our main algorithm enjoys significantly lower variance than ETC and offline approaches, which is a highly desirable advantage, meaning that with only one single run (which is indeed the case in practice), our main algorithm very likely would provide an acceptable performance, while the ETC and offline approaches could be potentially very bad.

In terms of the end-to-end variant, it has very flat curves when L is changing, and the reason is that the corresponding surrogate model is converging very slow with as few as 100 samples (so that no changes are reflected through the

[2]https://mininet.org/

[3]https://github.com/nokia/twampy

[4]https://github.com/vgropp/bwm-ng

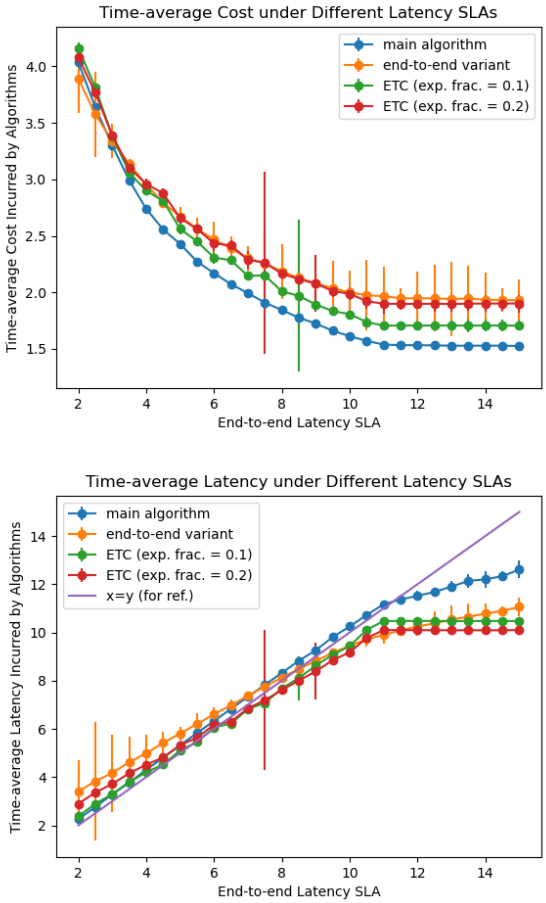

Fig. 1: Results on Synthetic Data when $T = 5 \times 10^5$

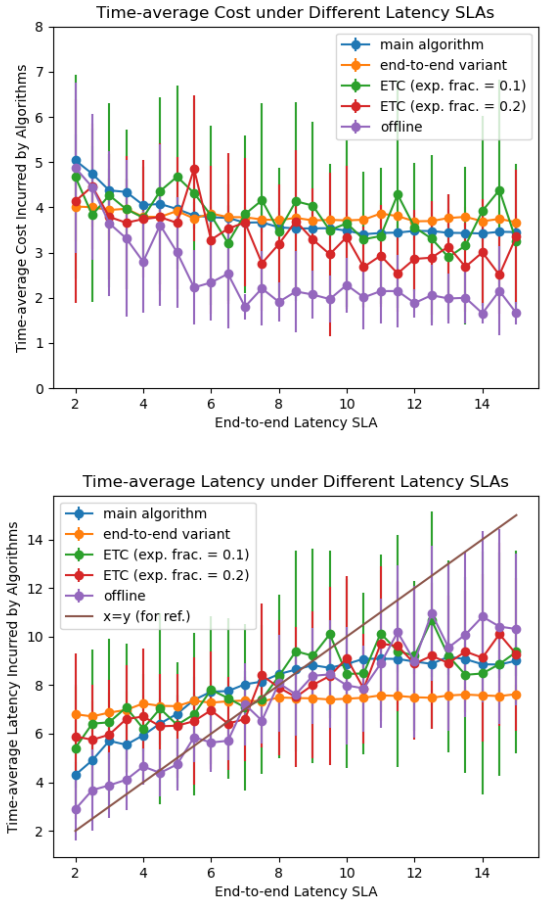

Fig. 2: Results on Synthetic Data when $T = 100$

outcome). This again suggests the advantage of leveraging domain-level feedback in designing the main algorithm.

C. Results on Trace Data

VIII. Related Work and Discussion

A. Previous work details

There are several previous works that study SLA decomposition, but they differ from this work in terms of problem formulation/setup, and/or algorithmic framework.

In [6], [12], the decision variables that the orchestrator needs to determine, are the same as in this work, namely, the decomposed SLAs. However, the objective and setup therein, are both totally different from ours. In these two works, it is assumed that each domain responds with either acceptance or rejection to the proposed decomposed domain-level SLA, where the probability of acceptance is some unknown function of the domain-level SLAs. The goal of the orchestrator is to find decomposed SLAs such that the given end-to-end SLA is satisfied and the probability of all domains simultaneously accepting the corresponding domain-level SLAs is maximized. In terms of the solution, they approach this problem in an offline fashion: firstly, some data samples (namely, domain-level SLAs and acceptance or not) are collected to train (parameterized) surrogate models (one for each domain and each type of SLA if multiple, which maps the domain-level SLA to the probability of acceptance). Finally, the decomposed SLA is determined by solving the constrained optimization with respect to the learned surrogate model.

Compared to them, our objective is to optimize the total cost through orchestrator. Moreover, we study the problem in the online setting where the algorithm learns to make better decomposition through interactions and more samples, instead of pre-collecting samples in an offline stage and fitting them with a surrogate in just one shot. The offline framework could suffer relatively limited applicability, as in practice there is typically no such "warm-up" stage for offline training.

In the follow-up work [13], the authors extend the aforementioned formulation in [6], [12] to the online scenario, and the key of the new algorithm design is to utilize a memory buffer to store (most recent) samples for online model update. Besides the difference in terms of the objective, our work utilizes theories from Bayesian Optimization to provide an *provably efficient* online algorithm, rather than heuristic.

While SLA decomposition is also included in the formulation of [9], the decision space for the orchestrator includes slice configuration, SLA decomposition, resource allocation,

etc, while in our formulation, the orchestrator is unaware of (and need not know) any details regarding the domain-level control. Our agnostic formulation enjoys less communication burden between the domain controllers and the end-to-end orchestrator, and naturally it does not need to know every different configuration space.

B. Future work and expansion of the current algorithms

In this work, we make the first attempt towards approaching SLA decomposition problem using Online Learning framework. There could be several interesting future directions on top of our work:

- We derive provably sufficient algorithms in the *stationary* setting only, which means that the underlying functions stay fixed and do not change over time. Given that the real-word networks can be time-evolving, it could be important to study the *non-stationary* setting [3], [7], [8], [15], [20].
- In this work, we consider only one SLA agreement (i.e., one constraint in the optimization problem given in Eq. (1) which corresponds to the end-to-end latency satisfaction). What if there are multiple SLAs to satisfy simultaneously, such as latency, throughput, and reliability? In that case, one decomposition needs to be done for each type of SLA. Note that SLAs like throughput and reliability do not have an additive end-to-end (de-)composition rule as latency does. As a result, both the problem formulation and designing BO-type algorithms could be highly non-trivial.
- We initiate the study by considering only one service. A more practical setting is that there are multiple services in the network (every of which has its own cost and constraint and forms an optimization like Eq. (1)) and we need to find a good set of decomposed SLAs for each of then. A reasonable formulation for this is not obvious, total resources will be shared and competed among them, which means there would be correlation across domains on the effects of decomposed SLAs to the observations. In other words, decomposed SLAs for domain 1 will affect not only the performance of domain 1 itself, but also the observations from other domains, since the resources remaining for other mains are affected by decomposed SLAs in domain 1, which makes the problem more complicated.

IX. Conclusion

In this paper, we presented a novel Bayesian Optimization-based algorithm for online end-to-end SLA decomposition in 5G network slicing. Our approach addresses the challenge of allocating SLA budgets across multiple domains by learning from real-time feedback, ensuring both SLA satisfaction and efficient resource utilization. Through theoretical analysis, simulations, and real-world testbed evaluations, we demonstrated the effectiveness and adaptability of our algorithm. This work advances the state-of-the-art in SLA decomposition, providing a practical solution for optimizing network slicing in 5G and beyond. Future research could explore extending our approach to non-stationary settings and multi-SLA scenarios, further enhancing its applicability in real-world networks.

References

[1] Thomas Bäck and Hans-Paul Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation*, 1(1):1–23, 1993.

[2] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[3] Xu Cai and Jonathan Scarlett. Lower bounds for time-varying kernelized bandits. *arXiv preprint arXiv:2410.16692*, 2024.

[4] Sayak Ray Chowdhury and Aditya Gopalan. On kernelized multi-armed bandits. In *International Conference on Machine Learning*, pages 844–853. PMLR, 2017.

[5] Eloise de Carvalho Rodrigues, Alvaro Valcarce Rial, and Giovanni Geraci. Towards mobility management with multi-objective bayesian optimization. In *2023 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE, 2023.

[6] Danny De Vleeschauwer, Chrysa Papagianni, and Anwar Walid. Decomposing slas for network slicing. *IEEE Communications Letters*, 25(3):950–954, 2020.

[7] Yuntian Deng, Xingyu Zhou, Arnob Ghosh, Abhishek Gupta, and Ness B Shroff. Interference constrained beam alignment for time-varying channels via kernelized bandits. In *2022 20th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)*, pages 25–32. IEEE, 2022.

[8] Yuntian Deng, Xingyu Zhou, Baekjin Kim, Ambuj Tewari, Abhishek Gupta, and Ness Shroff. Weighted gaussian process bandits for non-stationary environments. In *International Conference on Artificial Intelligence and Statistics*, pages 6909–6932. PMLR, 2022.

[9] H Esmat and B Lorenzo. Sla decomposition for sustainable end-to-end multi-domain multi-technology network slicing. *IEEE Wireless Communications*, 2024.

[10] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner Gardner, Marc Parizeau, and Christian Gagné. Deap: Evolutionary algorithms made easy. *The Journal of Machine Learning Research*, 13(1):2171–2175, 2012.

[11] Kaynam Hedayat, R Krzanowski, Al Morton, Kiho Yum, and Jozef Babiarz. A two-way active measurement protocol (twamp). Technical report, 2008.

[12] Cyril Shih-Huan Hsu, Danny De Vleeschauwer, and Chrysa Papagianni. Sla decomposition for network slicing: A deep neural network approach. *IEEE Networking Letters*, 2023.

[13] Cyril Shih-Huan Hsu, Danny De Vleeschauwer, and Chrysa Papagianni. Online sla decomposition: Enabling real-time adaptation to evolving systems. *arXiv preprint arXiv:2408.08968*, 2024.

[14] Michael Iannelli, Muntasir Raihan Rahman, Nakjung Choi, and Le Wang. Applying machine learning to end-to-end slice sla decomposition. In *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, pages 92–99. IEEE, 2020.

[15] Shogo Iwazaki and Shion Takeno. Near-optimal algorithm for non-stationary kernelized bandits. *arXiv preprint arXiv:2410.16052*, 2024.

[16] Fengjiao Li, Xingyu Zhou, and Bo Ji. (private) kernelized bandits with distributed biased feedback. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 7(1):1–47, 2023.

[17] Lorenzo Maggi, Alvaro Valcarce, and Jakob Hoydis. Bayesian optimization for radio resource management: Open loop power control. *IEEE Journal on Selected Areas in Communications*, 39(7):1858–1871, 2021.

[18] MA Schumer and Kenneth Steiglitz. Adaptive step size random search. *IEEE Transactions on Automatic Control*, 13(3):270–276, 1968.

[19] Wenjie Xu, Yuning Jiang, Bratislav Svetozarevic, and Colin Jones. Constrained efficient global optimization of expensive black-box functions. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 38485–38498. PMLR, 23–29 Jul 2023.

[20] Wenjie Xu, Yuning Jiang, Bratislav Svetozarevic, and Colin N Jones. Primal-dual contextual bayesian optimization for control system online optimization with time-average constraints. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 4112–4117. IEEE, 2023.

[21] Jia Yan, Qin Lu, and Georgios B Giannakis. Bayesian optimization for task offloading and resource allocation in mobile edge computing. In *2022 56th Asilomar Conference on Signals, Systems, and Computers*, pages 1086–1090. IEEE, 2022.