Joint Carbon-Latency Optimization for Online Service Function Chain Deployment

Yung-Lun Yang, Yan-Wei Chen, and Li-Hsing Yen
Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan.

***Abstract*—Service function chain (SFC) deployment is to embed virtualized network function (VNF) instances into a cloud-based infrastructure and chain them in sequence to provide a specific network service. Many approaches have been proposed for SFC deployments with diverse objectives. However, no SFC deployment approach factors in the embodied and operational carbon emissions, which are complicated by location- and time-varying carbon intensity (CI) and the amortization of carbon footprint among SFCs when they share VNF instances and communication links. Moreover, prioritizing lower carbon emissions for an SFC might degrade its end-to-end latency and vice versa. This work aims to jointly minimize carbon emissions and latency in SFC deployment, factoring in 1) SFC lifetime, traffic rates, and resource usage, 2) amortized embodied and operational carbon emissions, and 3) processing and propagation delay. We propose a rule to amortize the carbon footprint to SFCs and an SFC deployment algorithm based on Monte Carlo Tree Search (MCTS) with a time-series forecasting method to predict spatial and temporal fluctuations of CIs. Simulations based on real-world historical CI data and dynamic SFC arrivals and departures confirm the effectiveness of the proposed approach.**

***Keywords*—carbon footprint analysis, service function chain deployment, Monte Carlo Tree Search, time series forecasting, virtual network function placement**

I. Introduction

Traditional networks usually provide services via proprietary hardware, which ensures performance but also incurs high capital and operating costs [1]. As a way to enhance the flexibility and scalability of service provisioning, Network Function Virtualization (NFV) decouples network functions from proprietary hardware and turns them into software-based Virtual Network Functions (VNFs). VNFs are able to run on commercial off-the-shelf (COTS) servers, which enables more agile and efficient service provisioning solutions for network operators and enterprises.

A prominent use case of NFV is the service function chain (SFC), which provides specific network services by chaining VNFs in particular sequences to meet users' service demands. SFC deployment is to embed VNF instances into an NFV infrastructure (NFVI) and chain them in sequence. We may reduce the cost of SFC deployment by sharing a VNF instance or a server connection among multiple SFCs, but doing so may potentially increase the SFC end-to-end latency or other costs. Many studies on SFC deployment emphasized optimizing VNF placement for diverse goals, including latency [2, 3], quality of service [4], and so on. Study [5] aimed to optimize cost and enhance security, while [6] minimized response time and resource usage through heuristics. Meanwhile, energy-aware VNF placement is also a key concern. Study [7] employed lifecycle management to optimize energy efficiency, while others [8, 9] adopted dynamic scaling of VNF instances to reduce congestion and energy costs.

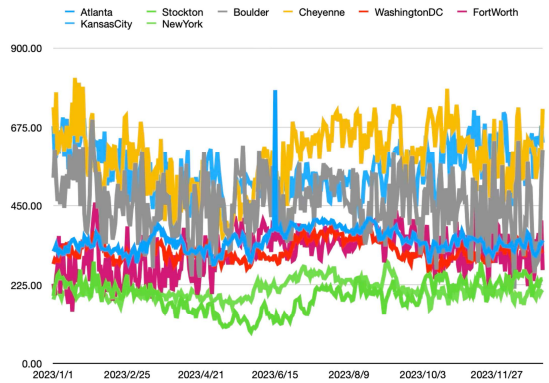

Figure 1: Daily Carbon Intensity (CI) in 2023 (Source: [16])

Despite the existence of these studies, carbon emissions in SFC deployment remain largely overlooked. As global emissions reached 37.15 billion tonnes in 2022 [10], and global temperatures have risen over $0.80°$C during 1961–1990 [11], there is growing urgency toward net-zero targets and sustainable consideration. Following this, the carbon footprint, a key metric in sustainability analysis, typically measures total greenhouse gas emissions, encompassing daily operational and lifetime embodied carbon footprint [12]. Carbon intensity (CI), representing the carbon dioxide (CO_2) output per unit of energy or economic value, is commonly used in literature to quantify carbon footprint emissions. Yet, few studies considered carbon emissions in SFC deployments. The research [13] minimized energy and traffic-aware costs, and [14] briefly mentioned that the electricity cost increases as the carbon footprint grows. The study [15] highlighted the trade-off between profit and carbon emissions in SFC deployment but used static CI and overlooked embodied and shared emissions.

In this paper, we address carbon emissions and end-to-end latency for SFC deployments in geo-distributed clouds. On one hand, we may exploit the spatial and temporal variations of CIs (Fig. 1) to minimize the carbon footprint for SFC deployments. We may also reuse existing VNF instances as much as possible to amortize the associated carbon footprint. On the other hand, doing so may increase the end-to-end latency due to extra processing and propagation delays. How to strike a balance between these two goals becomes an issue. We formulate an online SFC deployment (OSD) problem for

such a balance. The problem is challenged by the location- and time-varying CIs, and also calls for an accurate and fair way to allocate the carbon footprint (including both operational and embodied carbon emissions) due to the deployments to SFCs when SFCs share VNF instances and communication links.

The key contributions of this paper are summarized below.

- We show how to amortize carbon emissions to SFCs, factoring in SFC lifetime, traffic rates, and resource usage.
- We formulate an OSD problem for optimal SFC deployment, using FB Prophet for forecasting spatial and temporal CI values and MCTS for joint optimization of carbon emissions and end-to-end latency.
- Experimental results show that the proposed approach achieved the lowest cost compared to other methods.

The rest of the paper is organized as follows. Sec. II delivers the network architecture, analysis model, and problem definition, Sec. III details the algorithm, and Sec. IV presents performance evaluations. Finally, we conclude the paper in Sec. V.

II. System Model and Problem Formulation

A. Substrate Network Model

The substrate network from ETSI [17] is modeled as an undirected weighted graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_{|\mathcal{V}|}\}$ represents servers and $\mathcal{E} = \{e_{i,j} \mid v_i, v_j \in \mathcal{V}\}$ denotes physical links connecting two servers. Every server $v \in \mathcal{V}$ has limited computational R_v^{cpu} and memory resource R_v^{mem}, while each link $e_{i,j} \in \mathcal{E}$ has a maximum bandwidth B_e.

B. SFC Deployment and VNF Placement Model

We consider a fixed set of VNF types denoted as $\mathcal{F} = \{f_1, f_2, \cdots, f_{|\mathcal{F}|}\}$. Each VNF type $f \in \mathcal{F}$ is described by $(r_f^{\text{cpu}}, r_f^{\text{mem}}, \rho_f, \sigma_f)$, which specifies in sequence the CPU demand, the memory demand, the processing latency per unit traffic rate, and traffic scaling factor. The traffic scaling factor specifies the impact of a VNF type on traffic rates. The egress traffic rate of a VNF instance of type f will be $\sigma_f x$ if the instance's ingress traffic rate is x.

We use $\mathcal{S} = \{1, 2, \cdots, |\mathcal{S}|\}$ to denote the set of SFC deployment requests. Each request $s \in S$ designates 1) a sequence of VNFs $F_s = (f_{s1}, f_{s2}, \ldots, f_{sK_s})$, where K_s is the length of s, 2) a starting date t_s^{start} and an ending date t_s^{end} of s, 3) an ingress node v_s^{in} and an egress node v_s^{ex}, and 4) the traffic rate λ_s generated by v_s^{in} toward the first VNF. Let b_{sk} be the egress traffic rate of VNF f_{sk} (also the ingress traffic rate of f_{sk+1} if $k < K_s$). Accordingly, we have $b_{sk} = b_{sk-1}\sigma_{f_{sk}} = \lambda_s \prod_{j=1}^{k} \sigma_{f_{sj}}$, $\forall k, 1 \le k \le K_s$, where $b_{s0} = \lambda_s$.

We use $\mathbf{w}_s = (w_{s1}, \ldots, w_{sK_s})$ to represent a deployment of SFC s, where $w_{sk} \in \mathcal{V}$ is the server where the type-f_{sk} instance runs. We use $N_v^f(\tau)$ to indicate whether a type-f VNF instance is active on server v at time slot τ. Our model allows different VNF instances to run on the same server and different SFCs to share the same VNF instance. To track which SFC s uses VNF f on server v in time slot τ, we define $P_v^f(\tau) = \{(i,j) \mid f_{ij} = f, w_{ij} = v, t_i^{\text{start}} \le \tau \le t_i^{\text{end}}, \forall i \in \mathcal{S}, 1 \le j \le K_i\}$.

C. Carbon Footprint Model

The carbon footprint of substrate networks originates from servers and links, and is categorized into operational and embodied emissions [12]. We discuss these two parts separately in the following.

1) Operational Carbon Emissions

Operation carbon emissions stem from energy consumed in computation and communications. For the computation part, let pb_v and pm_v be server v's energy consumptions in idle and full-load modes, respectively. We assume that v's actual energy consumption during a time slot τ is proportional to the actual CPU resource usage in that slot [18]. Formally, v's energy consumption in time slot τ is

$$\mathbb{E}(v,\tau) = pb_v + (pm_v - pb_v) \cdot \frac{\sum_{f \in \mathcal{F}} (N_v^f(\tau) \cdot r_f^{\text{cpu}})}{R_v^{\text{cpu}}}. \quad (1)$$

Let $CI(v,\tau)$ be the CI value for powering server v in time slot τ, measured in gCO_2eq/kWh. The operational carbon emission of v in τ is $OCES(v,\tau) = \mathbb{E}(v,\tau) \cdot CI(v,\tau)$.

When multiple VNF instances share server v's CPU and memory resources, these instances share $OCES(v,\tau)$ in proportion to their resource usages. The sharing ratio of a VNF instance f is

$$\mathbb{F}(v,f,\tau) = \frac{\beta \cdot N_v^f(\tau) \cdot r_f^{\text{cpu}} + \gamma \cdot N_v^f(\tau) \cdot r_f^{\text{mem}}}{\beta \sum_{f' \in \mathcal{F}} (N_v^{f'}(\tau) \cdot r_{f'}^{\text{cpu}}) + \gamma \sum_{f' \in \mathcal{F}} (N_v^{f'}(\tau) \cdot r_{f'}^{\text{mem}})}, \quad (2)$$

where β and γ are weighting factors of CPU and memory usages, respectively. Furthermore, when a single VNF instance serves multiple SFCs, we distribute the instance's carbon footprint among these SFCs in proportion to their traffic rates processed by the instance. The ratio of SFC s's traffic rate processed by f_{sk} is given by

$$\mathbb{B}(v, f_{sk}, \tau) = \frac{b_{sk-1}}{\sum_{(i,j) \in P_v^{f_{sk}}(\tau)} b_{ij-1}}. \quad (3)$$

Therefore, the total operational carbon footprint in computation associated with deployment $\mathbf{w}_s$ is

$$C_{\text{sr}}^{\text{op}}(\mathbf{w}_s) = \sum_{\tau = t_s^{\text{start}}}^{t_s^{\text{end}}} \sum_{k=1}^{K_s} [OCES(w_{sk},\tau) \cdot \mathbb{F}(w_{sk}, f_{sk}, \tau) \cdot \mathbb{B}(w_{sk}, f_{sk}, \tau)]. \quad (4)$$

The operational carbon emission in communications needs special treatment because the two ends of a link may have different CI values due to spatial variation. A simple treatment is to take the average value as the link CI. We define $CID(u,v,\tau)$ to be the CI of the link from servers u to v in time slot τ. The total traffic rate passing through link (u,v) in time slot τ is $\mathbb{T}(u,v,\tau) = \sum_{i \in S} \sum_{j=1}^{K_i} \mathbb{I}_{u=w_{ij-1}} \mathbb{I}_{v=w_{ij}} \cdot b_{ij-1}$. Letting $\dot{\mathbb{E}}(x)$ be the energy consumption associated with traffic rate x, the amount of operational carbon emissions for link (u,v) in time slot τ is $OCEL(u,v,\tau) = \dot{\mathbb{E}}(\mathbb{T}(u,v,\tau)) \cdot$

$CID(u, v, \tau)$. Since a link bears traffic for multiple SFCs, the emissions are proportionally allocated to these SFCs based on their traffic rate contributions. Therefore, the total operational carbon footprint in communications associated with $\mathbf{w}_s$ is

$$C_{\text{lk}}^{\text{op}}(\mathbf{w}_s) = \sum_{\tau=t_s^{\text{start}}}^{t_s^{\text{end}}} \sum_{k=1}^{K_s} \mathbb{I}_{w_{sk-1} \neq w_{sk}} \, \mathit{OCEL}(w_{sk-1}, w_{sk}, \tau) \cdot \frac{b_{sk-1}}{\mathbb{T}(w_{sk-1}, w_{sk}, \tau)}, \tag{5}$$

where the condition $w_{sk-1} \neq w_{sk}$ is to rule out scenarios where f_{sk-1} and f_{sk} are on the same server (which makes (w_{sk-1}, w_{sk}) a virtaul link).

2) Embodied Carbon Emissions

Embodied carbon emissions stem from production, transport, and manufacturing to disposal throughout products' lifecycle. For computation part, we denote server v's embodied carbon footprint (ECF) by $\mathit{ECF}(v)$, and its daily contribution is $\mathit{ECF}(v)/L_v$, where L_v is the server's lifetime in days. We use (2) to distribute server v's daily ECF to all VNF instances running on v. We further distribute a particular instance's ECF equally to all SFCs that use it, which means, unlike (3), each SFC's share ratio is $\mathbb{N}(v, f, \tau) = 1/|P_v^f(\tau)|$. Each SFC that uses VNF f on server v in time slot τ thus receives an ECF given by $\mathit{ECES}(v, f, \tau) = (\mathit{ECF}(v)/L_v) \cdot \mathbb{F}(v, f, \tau) \cdot \mathbb{N}(v, f, \tau)$.

For a particular deployment $\mathbf{w}_s$ of SFC s, the lifetime ECF of s in computation sums up the ECF shares s receives from all its VNF instances, as

$$C_{\text{sr}}^{\text{em}}(\mathbf{w}_s) = \sum_{\tau=t_s^{\text{start}}}^{t_s^{\text{end}}} \sum_{k=1}^{K_s} \big[\mathit{ECES}(w_{sk}, f_{sk}, \tau)\big]. \tag{6}$$

For ECF in the communications part, let $\mathit{ECF}(e)$ be the ECF of link e and its daily ECF be $\mathit{ECF}(e)/L_e$, where L_e is the product lifetime of link e in days. The daily link ECF is equally shared by all the SFCs that use e at the same time, so each SFC receives a share $ECEL(e) = (\mathit{ECF}(e)/L_e)/(\sum_{j \in S} \sum_{l=1}^{K_j} \mathbb{I}_{\mathcal{E}_j^l = e})$, where $\mathcal{E}_j^l$ is the link from the l-th VNF to the $(l+1)$-th VNF of SFC j.

For a particular deployment $\mathbf{w}_s$ of SFC s, the lifetime ECF of s in communications sums up the ECF shares it receives from all links it uses:

$$C_{\text{lk}}^{\text{em}}(\mathbf{w}_s) = \sum_{\tau=t_s^{\text{start}}}^{t_s^{\text{end}}} \sum_{k=0}^{K_s} \mathit{ECEL}(\mathcal{E}_s^k). \tag{7}$$

The total carbon emissions of an SFC deployment $\mathbf{w}_s$ is therefore

$$C(\mathbf{w}_s) = C_{\text{sr}}^{\text{op}}(\mathbf{w}_s) + C_{\text{lk}}^{\text{op}}(\mathbf{w}_s) + C_{\text{sr}}^{\text{em}}(\mathbf{w}_s) + C_{\text{lk}}^{\text{em}}(\mathbf{w}_s). \tag{8}$$

D. End-to-End Latency

The end-to-end latency of an SFC consists of processing time by VNFs and propagation delay between them. Assuming that the processing time of a VNF instance is proportional to its ingress traffic rate, the total processing time of SFC s sums up all it's VNF instances' processing time:

$$L^{\text{proc}}(\mathbf{w}_s) = \sum_{\tau=t_s^{\text{start}}}^{t_s^{\text{end}}} \sum_{k=1}^{K_s} \underbrace{\left(\rho_{f_{sk}} \sum_{(j,l) \in P_{w_{sk}}^{f_{sk}}(\tau)} b_{jl-1}\right)}_{\text{processing latency of VNF } f_{sk} \text{ on server } w_{sk}}, \tag{9}$$

where $\rho_{f_{sk}}$ is the processing time per unit traffic rate of VNF f_{sk}.

The propagation delay through a link e depends on its length $\text{di}(e)$ and the constant signal speed c. The propagation latency of SFC s is the sum of all link propagation delays:

$$L^{\text{prop}}(\mathbf{w}_s) = \sum_{\tau=t_s^{\text{start}}}^{t_s^{\text{end}}} \sum_{k=0}^{K_s} \frac{\text{di}(\mathcal{E}_s^k)}{c}. \tag{10}$$

Therefore, the end-to-end latency associated with a deployment $\mathbf{w}_s$ is the sum of (9) and (10):

$$L(\mathbf{w}_s) = L^{\text{proc}}(\mathbf{w}_s) + L^{\text{prop}}(\mathbf{w}_s). \tag{11}$$

E. Problem Formulation

The online SFC deployment (OSD) problem is to create or reuse VNF instances for online SFC requests to minimize SFC latency and carbon footprint with resource constraints and the consideration of time- and location-varying CIs for servers and communication links. We regard the OSD problem as an integer nonlinear programming problem [18] to minimize carbon emissions and latency while meeting resource constraints.

We use a vector $\mathbf{w} = (\mathbf{w}_1, \cdots, \mathbf{w}_{|\mathcal{S}|})$ of decision variables to denote an offline deployment for all SFCs in $\mathcal{S}$. The offline version is formally formulated as

$$\min_{\mathbf{w}} \mathbb{E}\left[\sum_{i \in \mathcal{S}} (\phi_{\text{C}} C(\mathbf{w}_i) + \phi_{\text{L}} L(\mathbf{w}_i))\right] \tag{12}$$

subject to

$$\sum_{f \in F} \left(N_v^f(\tau) \cdot r_f^{\text{cpu}}\right) \leq R_v^{\text{cpu}}, \forall v \in \mathcal{V}, \forall \tau \geq 0, \tag{13}$$

$$\sum_{f \in F} \left(N_v^f(\tau) \cdot r_f^{\text{mem}}\right) \leq R_v^{\text{mem}}, \forall v \in \mathcal{V}, \forall \tau \geq 0, \tag{14}$$

$$\sum_{s \in \mathcal{S}} \sum_{k=0}^{K_s} \left(\mathbb{I}_{e \in \mathcal{E}_s^k} \cdot b_{sk}\right) \leq B_e, \forall e \in \mathcal{E}, \tag{15}$$

$$w_{sk} \in \mathcal{V}, \forall s \in \mathcal{S}, 1 \leq k \leq K_s, \tag{16}$$

$$0 \leq \phi_{\text{C}}, 0 \leq \phi_{\text{L}}. \tag{17}$$

Equations (13), (14) and (15) are resources constrains, while Eq. (16) ensures valid deployments.

III. Proposed Approaches

A. Time Series Forecasting

The study [19] finds FB Prophet effective for forecasting energy consumption over five years, making it promising for CI prediction within SFC lifetime due to its correlation with energy generation and consumption. Since electricity generation follows periodic patterns, we apply FB Prophet to forecast CI values, leveraging its periodic patterns. Our forecasting model follows an additive regression [20] as $CI(v,t) = f_v(t) + g_v(t) + \epsilon_t$, where the unit t is a day.

The logistic growth $f_v(t)$ captures the non-periodic trend [20] of the CI value of server v. The Fourier series $g_v(t)$ represents periodic fluctuations [19], like yearly seasonality. The normally distributed error ϵ_t reflects any idiosyncratic variations. The following equation computes the forecasting CI of the rented server v within a lifetime of SFC L_s in the network and updates before the deployment for precise results.:

$$CI_v = \sum_{\tau \in L_s} (CI(v,\tau)). \tag{18}$$

B. Greedy and A* Algorithm

We consider a greedy algorithm that iteratively selects the best server for each VNF in an SFC. For each $f_{sk} \in F_s$, it checks each server v for resource constraints and estimates deployment cost. The best candidate is chosen, and the process repeats.

We also consider an A* algorithm that regards the deployment as the shortest path problem in a weighted graph through a single objective merging carbon emissions and latency, and designing an approximate heuristic for an immediately identifiable goal. A greedy algorithm iteratively selects the best VNF placement until reaching a terminal state, which serves as the heuristic evaluation. The algorithm selects the lowest-cost state, places the next VNF, updates the heuristic, and repeats until completion.

C. Single-Player Monte-Carlo Tree Search (SP-MCTS)

We propose a Single-Player MCTS (SP-MCTS) [21] to solve the OSD problem. SP-MCTS, a tree search algorithm [22], incrementally builds a decision tree where each node n_i represents the optimal VNF-to-server mapping. A node is defined as $n_i = (r_i, N_i, n_p, \mathbf{w}_i)$, where r_i is the node's reward, N_i the visit count, n_p the parent node, and $\mathbf{w}_i$ the current deployment state. SP-MCTS balances precise tree search with random space sampling to evaluate potential moves. SP-MCTS follows five key stages [21]: (1) Selection, (2) Expansion, (3) Simulation, (4) Backpropagation, and (5) Best Selection.

In the selection step, a target node is selected down from the root using a modified Upper Confidence Bounds for Trees (m-UCT) [21], balancing exploration and exploitation by selecting the highest UCT value of each successive node to guide the search. For expansion, if the selected leaf node is not terminal, its children are generated, each representing a possible deployment. The simulation step then uses a greedy approach to explore deployment sequences until all VNFs are placed and evaluates the deployment results. Finally, the simulation reward is propagated back through the visited nodes.

Table I: Specification of VNF Types [23, 24, 25]

VNF	CPU	Memory	ρ_f	σ_f
NAT	1 core	1 GB	0.05 s/Mbps	1
FW	2 cores	3 GB	0.05 s/Mbps	1
TM	1 core	3 GB	0.05 s/Mbps	1
WOC	1 core	2 GB	0.05 s/Mbps	0.65
IDPS	2 cores	2 GB	0.05 s/Mbps	1.5
VOC	2 cores	2 GB	0.05 s/Mbps	0.65

Table II: SFC Types [25, 26]

Service	Chained VNFs	Data Rate	Prob.
Web Service	NAT-FW-TM-WOC-IDPS	0.1 Mbps	50%
Video Streaming	NAT-FW-TM-VOC-IDPS	4 Mbps	20%
Cloud Gaming	NAT-FW-VOC-WOC-IDPS	4 Mbps	20%
VoIP	NAT-FW-TM-FW-NAT	0.064 Mbps	10%

Algorithm 1 SP-MCTS Pseudo-Code

Require: n_0: initial root
Ensure: optimal node in the tree
EstimateCI(n_0) ▷ Updated carbon intensity
repeat
  $n_i \leftarrow n_0$
  Top-down select n_i with maximum m-UCT value
  if Constraints of n_i are satisfied **then**
    if n_i has been visited **then**
      Generate child nodes
    Perform greedy method and evaluate deployment
  Compute cost
  Backpropagate the cost through the visited nodes
until Iteration budget is reached
return BESTCHILD(n_0)

IV. Performance Evaluation

This section details the experimental setup, simulation, and performance evaluation of carbon emissions and latency.

A. Environment Setting

Experiments were conducted on a 2.1 GHz Intel(R) Core(TM) i7-14700 CPU with 128-GB RAM.

We evaluated up to 180 SFCs ($|S| \in [1, 180]$) and six VNF types ($|F| = 6$): Network Address Translator (NAT), Firewall (FW), Traffic Monitor (TM), WAN Optimization Controller (WOC), Intrusion Detection Prevention System (IDPS), and Video Optimization Controller (VOC), each with specific parameters detailed in Table I. The electricity intensity was 0.06 kWh/GB [27]. There were four types of SFCs: web service, video streaming, cloud gaming, and VoIP. Each type

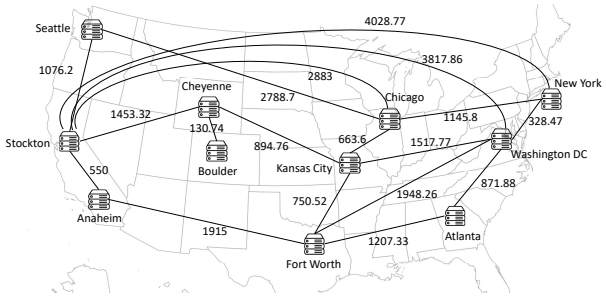

Figure 2: Sprint Network Topology

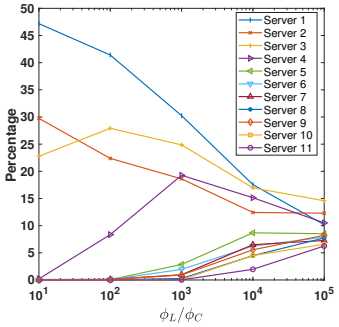

Figure 3: Percentage of VNF instances at each server

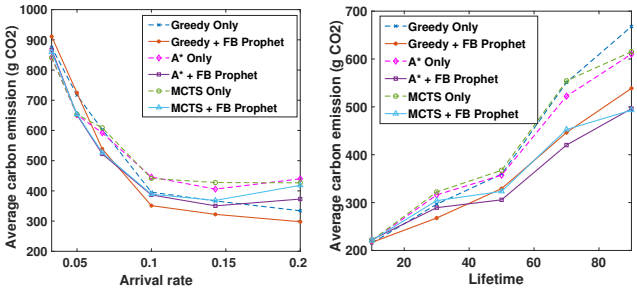

Figure 4: Tendency of Carbon Emissions w.r.t. (a) Arrival Rate (b) Lifetime

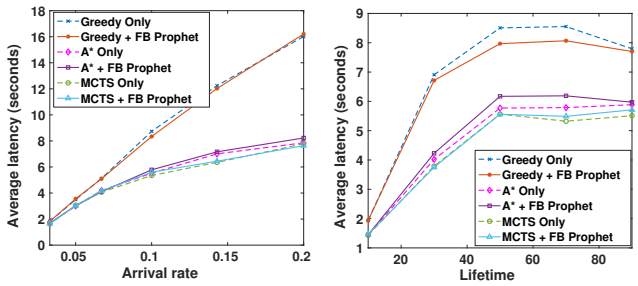

Figure 5: Results of Latency w.r.t. (a) Arrival Rate (b) Lifetime

required a specific SFC and input data rate λ (see Table II). SFC requests were generated by a Poisson process (mean rate: 0.2 to 0.033 requests per day) with the type of SFC determined by the probability indicated in Table II. The lifetime of an SFC was exponentially distributed with a mean ranging from 10 to 90 days. Results were averaged over 30 simulations, with a simulation interval from March 1 to September 30.

B. Network Topology

We used the Sprint network topology from the Internet Topology Zoo [28], which consists of 11 servers and 18 links as shown in Fig. 2. The base and max energy consumptions of servers were 0.17 kW and 0.5 kW, respectively. Each server had 50 CPU cores and 50 GB of memory. Links were of 15 Mbps bandwidth and different lengths. Daily CI was gathered from historical data in [16]. The weights for carbon emission and latency were $\phi_{\mathrm{C}} = 1$ and $\phi_{\mathrm{L}} = 100$. For 80% of the SFCs, the ingress and the egress nodes were the same due to locality.

C. Tradeoff between Carbon Emission and Latency

Fig. 3 shows the percentage of VNF instance placements in every server under different latency-to-carbon emission weight ratios. Most VNFs were placed on low-carbon servers and shared emissions. As the ratio increased, VNFs were distributed more evenly since latency became the dominant factor, reducing processing delays.

D. Carbon Emissions

Building on the previous setup, we wanted to observe the impacts of arrival rate and SFC lifetime on carbon emissions. Fig. 4(a) illustrates that higher arrival rates reduced average carbon emissions as more SFCs shared emissions over the same time interval. Conversely, Fig. 4(b) highlights emissions rising with longer lifetimes due to extended service durations. Meanwhile, both results confirm the effectiveness of the FB Prophet model. Applying the FB Prophet model created gaps between all three approaches, especially for longer SFC lifetimes, since it enhanced SFC deployment by predicting future CI within SFC's lifetime.

E. End-to-End Latency

We also analyzed the impacts of arrival rate and SFC lifetime on end-to-end latency. Fig. 5(a) shows that as the arrival rate increased, so did the average processing latency as more SFCs were involved. Greedy approaches suffered more from congestion, while MCTS maintained lower latency. On the other hand, Fig. 5(b) shows that longer SFC lifetimes also increased latency, particularly for 10- to 50-day lifetimes. However, deploying VNF instances on high-CI servers mitigated congestion when lifetimes reached 50 to 90 days, albeit at the cost of higher carbon emissions. This is because our setting of ϕ_{C} and ϕ_{L} prioritized latency. Overall, MCTS achieved lower latency by placing VNF instances on less congested servers.

V. Conclusions

This paper explores SFC deployment to minimize carbon emissions and end-to-end latency while meeting resource constraints. We devise an accounting rule that amortizes operational and embodied carbon emissions to SFCs based on SFC lifetime, traffic rate, and resource usage. We use FB Prophet to predict spatiotemporally varying CI and propose MCTS for SFC deployments. Simulation results confirmed the trade-off under consideration. Moreover, intensive SFC arrivals reduced

average carbon emissions due to amortizations, while longer lifetimes increased both emissions and latency. Results also show that MCTS with FB Prophet outperformed the others.

For future work, we will explore the scalability and adaptability of the proposed model and approach in more complex scenarios, including larger network topologies and diverse server types. We also plan to apply some advanced models, such as Transformer-based architectures, for more accurate and robust CI pattern capturing and forecasting. Additionally, we may conduct further investigation into how prediction uncertainty affects deployment decisions and overall system performance. Finally, extensions to our analysis model should consider more realistic assumptions, including non-linear resource usage-amortization models and highly dynamic variations in CI, to better reflect real-world conditions.

ACKNOWLEDGMENT

This work was supported in part by the National Science and Technology Council of Taiwan under grant numbers 113-2221-E-A49-180 and 114-2218-E-A49-018.

REFERENCES

[1] X. Han *et al.*, "A service function chain deployment method based on network flow theory for load balance in operator networks," *IEEE Access*, vol. 8, pp. 93 187–93 199, 2020.

[2] L. Popokh *et al.*, "Physical and virtual resources inventory modeling for efficient VNF placement," in *IEEE Int'l Conf. on Consumer Electronics*, 2020, pp. 1–6.

[3] F. Tian *et al.*, "Bidirectional service function chain embedding for interactive applications in mobile edge networks," *IEEE Trans. Mob. Comput.*, vol. 23, no. 5, pp. 3964–3980, 2024.

[4] M. M. Tajiki *et al.*, "Joint energy efficient and QoS-aware path allocation and VNF placement for service function chaining," *IEEE Trans. Netw. Serv. Manag.*, vol. 16, no. 1, pp. 374–388, 2019.

[5] D. Dwiardhika and T. Tachibana, "Cost efficient VNF placement with optimization problem for security-aware virtual networks," in *IEEE Int'l Conf. on Cloud Networking*, 2018, pp. 1–3.

[6] L. Dinh-Xuan *et al.*, "Performance evaluation of service functions chain placement algorithms in edge cloud," in *Int'l Teletraffic Congress*, vol. 01, 2018, pp. 227–235.

[7] M. Chen *et al.*, "Energy-saving and resource-efficient algorithm for virtual network function placement with network scaling," *IEEE Trans. Green Commun. Netw.*, vol. 5, no. 1, pp. 29–40, 2021.

[8] A. Gember *et al.*, "Stratos: A network-aware orchestration layer for virtual middleboxes in clouds," 2014, *arXiv:1305.0209*.

[9] X. Wang *et al.*, "Online learning-assisted VNF service chain scaling with network uncertainties," in *IEEE Int'l Conf. on Cloud Computing*, 2017, pp. 205–213.

[10] H. Ritchie and M. Roser, "CO_2 emissions," *Our World in Data*, 2020, https://ourworldindata.org/co2-emissions.

[11] H. Ritchie *et al.*, "CO_2 and greenhouse gas emissions," *Our World in Data*, 2023, https://ourworldindata.org/co2-and-greenhouse-gas-emissions.

[12] T. Kennes, "Measuring IT carbon footprint: What is the current status actually?" 2023, *arXiv:2306.10049*.

[13] C. Pham *et al.*, "Traffic-aware and energy-efficient VNF placement for service chaining: Joint sampling and matching approach," *IEEE Trans. Serv. Comput.*, vol. 13, no. 1, pp. 172–185, 2020.

[14] Y. Yue *et al.*, "Energy-efficient and traffic-aware VNF placement for vertical services in 5G networks," in *Proc. IEEE TrustCom*, 2022, pp. 1316–1322.

[15] T. Di Riccio *et al.*, "Sustainable placement of VNF chains in intent-based networking," in *Proc. IEEE/ACM Int'l Conf. Utility and Cloud Computing*, 2024.

[16] "Carbon intensity map," accessed: Apr. 20, 2025. [Online]. Available: https://app.electricitymaps.com

[17] "Network functions virtualisation (NFV); architectual framework," ETSI, GS NFV 002 v1.2.1, Dec 2014.

[18] J. Liang *et al.*, "Sustainable virtual network function placement and traffic routing for green mobile edge networks," *IEEE Trans. Green Commun. Netw.*, vol. 8, no. 4, pp. 1450–1465, Dec. 2024.

[19] S. Chaturvedi *et al.*, "A comparative assessment of SARIMA, LSTM RNN and FB prophet models to forecast total and peak monthly energy demand for india," *Energy Policy*, vol. 168, p. 113097, 2022.

[20] S. J. Taylor and B. Letham, "Forecasting at scale," *Am. Stat.*, vol. 72, no. 1, pp. 37–45, 2018.

[21] C. B. Browne *et al.*, "A survey of Monte Carlo tree search methods," *IEEE Trans. Comput. Intell. AI Games.*, vol. 4, no. 1, pp. 1–43, 2012.

[22] D. Perez *et al.*, "Knowledge-based fast evolutionary MCTS for general video game playing," in *IEEE Conf. on Comput. Intell. and Games*, 2014, pp. 1–8.

[23] D. Dietrich *et al.*, "Multi-provider service chain embedding with nestor," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, no. 1, pp. 91–105, 2017.

[24] W. Rankothge *et al.*, "On the scaling of virtualized network functions," in *IFIP/IEEE Symp. on Integrated Network and Service Management*, 2019, pp. 125–133.

[25] L. Ruiz *et al.*, "A genetic algorithm for VNF provisioning in NFV-enabled cloud/MEC RAN architectures," *Applied Sciences*, vol. 8, no. 12, 2018.

[26] M. Savi *et al.*, "Impact of processing-resource sharing on the placement of chained virtual network functions," *IEEE Trans. Cloud Comput.*, vol. 9, no. 4, pp. 1479–1492, 2021.

[27] J. Aslan *et al.*, "Electricity intensity of internet data transmission: Untangling the estimates," *J. Ind. Ecol.*, vol. 22, no. 4, pp. 785–798, 2018.

[28] S. Knight *et al.*, "The Internet topology Zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.