AI Model Placement for 6G Networks under Epistemic Uncertainty Estimation

Liming Huang, Yulei Wu, Juan Marcelo Parra-Ullauri, Reza Nejabati, and Dimitra Simeonidou
School of Electrical, Electronic and Mechanical Engineering, University of Bristol, BS8 1UB, Bristol, UK
{liming.huang, y.l.wu, jm.parraullauri, reza.nejabati, dimitra.simeonidou}@bristol.ac.uk

***Abstract*—The adoption of Artificial Intelligence (AI) based Virtual Network Functions (VNFs) has witnessed significant growth. This surge has introduced the critical challenge of orchestrating AI models within next-generation 6G networks. Finding optimal AI model placement is significantly more challenging than placing conventional software based VNFs, due to the introduction of considerably more uncertain factors by AI models, such as varying computing resource consumption, dynamic storage requirements, and changing model performance. To explore the important AI model placement problem in the context of such epistemic uncertainties, this paper presents a novel approach employing a sequence-to-sequence (S2S) neural network integrated with uncertainty considerations. The S2S model, characterized by its encoding-decoding architecture, is designed to take the service chain with a number of AI models as input and produce the corresponding placement of each AI model. To address the introduced uncertainties, our methodology incorporates orthonormal certificates and fuzzy logic as uncertainty estimators, augmenting the capabilities of the S2S model. Experiments demonstrate that the proposed strategy achieves competitive results across diverse AI model profiles, network environments, and service chain requests.**

***Index Terms*—Network Function Virtualization, AI Model Placement, Uncertainty, Fuzzy Logic, 6G**

I. Introduction

With the deployment of 5G technology and its global standardization, many efforts are intensified to explore the future landscape of 6G networks. Virtual Network Function (VNF) placement, a pivotal technique within Network Function Virtualization (NFV) for 5G, is poised to retain its crucial role in shaping the architecture of 6G networks. Different from conventional VNFs, typically reliant on specific functional softwares, the evolving trend in 6G networks is the increasing integration of VNFs driven by artificial intelligence (AI) models [1] AI model-based VNFs possess a significant edge over traditional software-based VNFs due to their adaptability and efficiency[1]. Unlike traditional VNFs, AI models can dynamically adjust to varying network conditions in real-time, optimizing performance and resource allocation. Additionally, AI models have the capability to self-learn and improve over time, enhancing overall network functionality and reducing the need for manual intervention. This superior adaptability and self-optimization make AI models a more reliable and scalable solution for modern network infrastructures, ultimately leading to improved reliability, performance, and cost-effectiveness.

[1]In this paper, we refer to AI model-based VNFs as AI models, and to traditional software-based VNFs as traditional VNFs.

There are noticeable differences between the placement of traditional VNFs and the placement of AI models. Traditional VNF placement primarily deals with software functions that often have fixed power consumption, running time, and model size. These predetermined factors contribute to more predictable placement strategies [2]. In contrast, AI model placement introduces several challenges due to the inherent uncertainties surrounding AI models. One of the primary challenges is the variability in computing resource requirements of AI models. That is because AI models often necessitate extensive training, the duration of which is unpredictable and depends on factors such as dataset size, complexity, and available computing resources. Varying training time brings uncertain computing power and resource consumption in AI model placement. Furthermore, the storage space required for training AI models can vary significantly based on the model's architecture and the nature of the input data. This uncertainty impacts the allocation and management of storage resources within the network. Additionally, AI models' operational performance fluctuates based on changing network conditions and evolving data patterns, making it challenging to predict their efficiency and adaptability over time. Consequently, these uncertainties surrounding computing resource, model storage, and model performance pose significant hurdles in determining optimal placement strategies for AI models within dynamic network environments.

To this end, this paper focuses on the placement of AI models and explicitly considers the impact of various epistemic uncertainties introduced by AI models. This problem is formalized as a constrained combinational optimization problem but it is NP-hard to solve [3]. Since it is impossible to directly obtain the optimal solution for a given specific instance, this paper adopts a neural combinatorial optimization (NCO) strategy to solve the placement of AI models. The NCO strategy is to train the neural network by constructing a reinforcement learning (RL) environment where the RL reward can help the neural network perform loss design and backpropagation without the need for supervised labels. To achieve the AI model placement, we devise a sequence-to-sequence (S2S) neural network with the encoder-decoder structure of multiple Long Short-Term Memory (LSTM) modules. The proposed S2S neural network, for a given Network Service (NS) request, considers the state of the NFV infrastructure and then learns a placement policy to minimize the overall energy consumption while satisfying specific quality of service

(QoS) constraints. To solve the uncertainty issues, we propose to use the Orthonormal Certificates (OCs) [4] for uncertainty estimations, and then use the fuzzy logic to represent the estimations and fuse them into the S2S model.

The main contributions of this paper are given as follows:

- This paper is the first of its kind to address AI model placement problems for 6G networks, explicitly considering the uncertainties introduced by AI models due to varying computing resource consumption, dynamic storage requirements, and changing model performance.
- A sequence-to-sequence neural network is devised for this problem. We adopt the neural combinatorial optimization strategy to train the proposed network within an RL environment.
- We propose to use the Orthonormal Certificates for uncertainty estimations in AI model placement. Then, the fuzzy logic is devised to represent uncertainty estimations and fuse them into the sequence-to-sequence model.
- Experiments substantiate the competitive performance of our model across a spectrum of AI model profiles, varying network environments, and diverse service chain requests. Notably, in scenarios where a service chain comprising 30 AI models is allocated to 20 servers, our model excels with a remarkable request acceptance ratio that is $3 \sim 4$ times more than that of other related models.

II. Related Work

A. VNF Placement

The VNF placement problem has garnered considerable attention. Woldeyohannes et al. [5] addressed it through a multi-objective integer linear programming approach, optimizing admitted flows, node and link utilization, and meeting delay requirements. Chen et al. [6] focused on delay-aware VNF placement, introducing heuristic algorithms for optimal backup VNF placement and routing to ensure NS availability while minimizing transmission delays. Studies on VNF reusability and sharing for cost reduction have been conducted as well. Malandrino et al. [7] emphasized VNF placement within a single physical node with shared instances among multiple NSs, using an efficient heuristic to prioritize NSs while adhering to delay requirements. Ren et al. [8] examined VNF reusability in delay-aware multicast NS placement, developing algorithms to maximize system throughput and minimize costs for admitted multicast NSs while meeting delay requirements.

While the aforementioned research has made significant strides in addressing VNF placement problems, it predominantly focuses on conventional software functions, overlooking the increasingly prevalent usage of AI models. Moreover, the uncertainties associated with AI models placement remain unexplored within this body of work. This paper aims to delve into this novel concern and investigates the implications of deploying AI models, a dimension largely overlooked in prior studies on VNF placement.

B. Uncertainty Learning

Uncertainty learning holds a key role in determining when to abstain from predictions across various tasks. In deep learning, Natasa et al. [4] introduced neural network estimations for aleatoric and epistemic uncertainties. In RL, Vincent et al. [9] proposed the utilization of uncertainty estimation to alleviate the adverse effects of noisy supervision, and introduced an inverse variance RL approach to enhance sample efficiency and overall performance.

Within VNF placement, prevailing research highlights uncertainty as primarily originating from the dynamic nature of the network environment, consequently necessitating adaptive resource allocation strategies [10], [11]. However, in the domain of VNF placement, limited attention has been directed toward understanding uncertainty issues arising from the AI training process. This paper uniquely centres on AI-based VNF placement, delving into the exploration of an uncertainty estimator that navigates epistemic uncertainty and fuzzy logic, thus contributing to a more comprehensive understanding of optimal AI model placement strategies in NFV.

III. Problem Formulation

A. Problem Statement

This paper addresses the strategic placement of a variety of AI models into a set of host servers, denoted as $h \in H$, in support of the delivery of network services. Each host server possesses limited computing, storage, and connectivity resources ($r \in R$). The host servers are interconnected using specific link connections ($i \in L$), each with its unique attributes such as bandwidth and propagation delay. The primary aim is to optimize the placement of AI models involved in a given network service, minimizing the overall infrastructure power consumption. This objective holds significant importance, as it directly impacts operational costs, promotes sustainability by reducing the carbon footprint, extends hardware lifespan, and ensures long-term scalability. Beyond minimizing power consumption, this strategic placement must align with constraints concerning virtual resources availability, link capacities, and the latency thresholds mandated by individual services. Meanwhile, the uncertainty factors brought by AI models should also be considered in solving this problem.

Following the nomenclature in [3], the set of host servers is denoted as $\{h_1, h_2, \cdots, h_n\}$ within H, and A represents the repository of available AI models. A network service comprises an array of $m \in \{1, \cdots, M\}$ AI models forming a service chain $s = (a_1, a_2, \cdots, a_m)$ where $a \in A$. The entire combinatorial space of these service chains is represented as S. The problem entails identifying the optimal set of placements, denoted as $x \in \{0,1\}^{m \times n}$, where x_{ah} represents a binary status variable indicating whether an AI model $a \in A$ is placed in host $h \in H$ (1 for positive placement and 0 for negative). The search space for the solution is $\Omega = x \in \{0,1\}^{m \times n} \; s.t. \; \sum_h x_{ah} = 1, \; \forall a \in s$, where the constraint ensures each AI model is placed on only one host server at a time. In order to more accurately

TABLE I: Problem Formalization Variables

1	H	set of hosts
2	L	set of links
3	A	set of AI models
4	S	set of NS chains
5	R	set of resources
6	P	set of placements
7	r_h	amount of resources r available in host h
8	r_a	amount of resources r requested by AI model a
9	W_h^e	idle power consumption of host h
10	W_h^c/W_h^g	power consumption of each CPU/GPU in host h
11	W_{net}	power consumption per bandwidth unit on links
12	b_i	bandwidth of the link i
13	l_a	latency due to computation time of AI model a
14	l_i^s	latency on the link i produced by service chain s
15	b_a^s	bandwidth demanded by a in service chain s
16	l^s	maximum latency allowed on the service chain s
17	x_{ah}	binary placement variable for AI a in host h
18	y_h	binary activation variable for host h
19	g_i	binary activation variable for link i

restore the real network environment, we also introduce server activation variables $y_h \in \{0,1\}$ indicating whether the server is executing any AI models (1 for active, and 0 for powered off) and link activation variables $g_i \in \{0,1\}$ signifying whether a link carries traffic (1) or not (0).

In terms of power consumption, activated servers ($y_i = 1$) consume a minimum power W_h^e and their power consumption escalates with the cumulative CPU and GPU demand of the placed AI models. Each unit of CPU and GPU in use consumes W_h^c and W_h^g watts, respectively. Concerning links, they too have an associated energy cost calculated based on the power consumption of each bandwidth unit (W_{net} multiplied by bandwidth utilized in each link). Moreover, the resource availability ($r \in R$) of each server h is denoted as r_h. The resource requirements of an AI model a are denoted as r_a^{type} for types c (CPU) or g (GPU). The bandwidth required for data transfer of AI model a within service $s \in S$ is denoted as b_a^s. For each link, the maximum bandwidth allowed in link i is represented as b_i. For the latency requirement, l_a signifies the latency due to the computation time of AI model a, and l_i^s represents the latency in link i due to service s. The maximum latency permitted for each service chain s is denoted as l_s. A summary of defined variables and problem parameters is presented in Table I.

B. Mathematical Formulation

In previous VNF placement research [12], [13], host servers are usually characterized by a certain power profile that grows in proportion to the computing utilization because traditional VNFs usually have fixed energy consumption. Therefore, utilising our nomenclature, the optimized cost function with certain relationships for VNF placement can be described as follows:

$$\arg\min_{x\in\Omega}\Big\{\sum_{h\in H}\Big[\sum_{a\in s}(W_h^c\cdot r_a^c + W_h^g\cdot r_a^g)\cdot x_{ah} + W_h^e\cdot y_h\Big] + \sum_{i\in L}\sum_{a\in s} W_{net}\cdot b_a^s\cdot x_{ah}\Big\} \tag{1}$$

$s.t.$

$$\sum_{a\in s} r_a\cdot x_{ah} \le y_h\cdot r_h, \forall h\in H, r\in R \tag{2}$$

$$\sum_{a\in s} b_a^s\cdot x_{ah} \le g_i\cdot b_i, \forall i\in L \tag{3}$$

$$\sum_{h\in H}\sum_{a\in s} l_a\cdot x_{ah} + \sum_{i\in L}\sum_{a\in s} l_i^s\cdot x_{ah} \le l^s, \forall h\in H, i\in L \tag{4}$$

where Eq. (1) represents the calculation of power consumption, incorporating the summation of power usage attributed to activated servers and the collective expense associated with active links. In Eq. (2), this constraint ensures that the total resource usage within a server does not surpass the available resources r_h in active servers. Eq. (3) defines capacity constraints for bandwidth. Lastly, Eq. (4) outlines constraints regarding the latency requirements l^s for a network service s which includes the computing time of VNFs and the latency over the links.

In AI model placement, the AI training process generates a range of uncertainties. Firstly, model training time t is an uncertain factor for almost all AI models causing the dynamic computing resource consumption. Different input data, network environment and available resources make model's training time vary considerably. Even using the same data and training the same model in an static environment with fixed parameter settings, the AI model's training time may also vary if we train the model twice given that the gradient descent of neural networks is stochastic. Secondly, the storage space usage d is uncertain in the model training process. For example, an AI model usually converges after 100 training epochs, and we usually save the trained model at that epoch for use. However, in another training process for this model, model convergence does not occur at the 100th epoch, so the model will be stored from the 100th epoch to the actual convergence epoch as we have empirically set the model saving from the 100th epoch. Consequently, the storage space usage increases for this training process. Thirdly, model performance quality q is also an uncertain factor. Since the gradient descent of neural networks is stochastic, and neural networks usually have more than $10e4$ parameters, the optimal point achieved by the model in each training is difficult to be the same although they are close. Therefore, in order to carry out proper AI model placement, these three uncertainties have to be considered, including computing resource consumption, storage space usage d, and model performance quality q.

Considering the above-mentioned uncertainties in the placement strategy shown in Eqs. (1)-(4), we formulate the optimization of AI model placement as follows:

$$\arg\min_{x\in\Omega}\Big\{\sum_{h\in H}\Big[\sum_{a\in s}(W_h^c\cdot r_a^c\cdot f_c(t)) + W_h^g\cdot r_a^g\cdot f_g(t))\cdot x_{ah} + W_h^e\cdot y_h\Big] + \sum_{i\in L}\sum_{a\in s} W_{net}\cdot b_a^s\cdot x_{ah}\Big\} \tag{5}$$

$s.t.$

$$\sum_{a\in s} r_a\cdot x_{ah} \le y_h\cdot r_h, \forall h\in H, r\in R \tag{6}$$

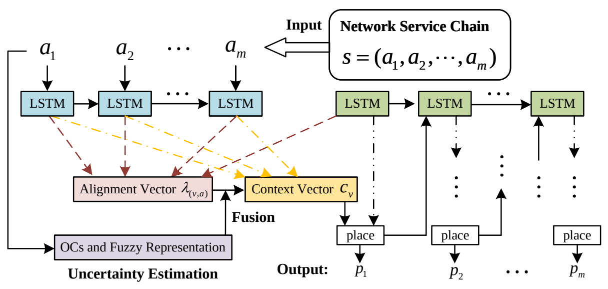


Fig. 1: The framework of our AI placement model.

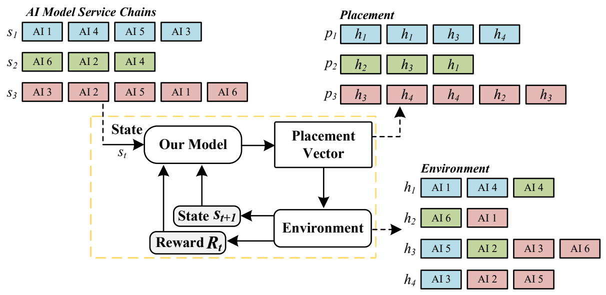


Fig. 2: The learning framework for the S2S model with RL environment.

$$\sum_{a\in s} b_a^s \cdot x_{ah} \leq g_i \cdot b_i, \forall i \in L \tag{7}$$

$$\sum_{h\in H}\sum_{a\in s} l_a \cdot x_{ah} + \sum_{i\in L}\sum_{a\in s} l_i^s \cdot x_{ah} \leq l^s, \forall h \in H, i \in L \tag{8}$$

$$\sum_{a\in s} d_a \cdot x_{ah} \cdot f_d(t) \leq y_h \cdot d_h, \forall h \in H, d \in D \tag{9}$$

$$\sum_{a\in s} q_a \cdot x_{ah} \cdot f_q(t) \geq q_{sla}, \forall h \in H \tag{10}$$

where d_h is the storage space available in host servers and q_{sla} is the minimum achieved model performance given by the Service Level Agreement (SLA). $f_c(t)$ and $f_g(t)$ are power consumption uncertainty functions for CPU and GPU. $f_d(t)$ is the uncertainty function for storage space usage. $f_q(t)$ is the model performance uncertainty function. The above three functions all use model training time t as the independent variable. In addition, the function mapping relationship for uncertainties cannot be given a specific mathematical expression, but can be learned through neural networks.

IV. Learning to Optimize for AI Model Placement

A. Sequence-to-Sequence Model for Placement

For AI model placement tasks, an NS chain $s = (a_1, a_2, \cdots, a_m)$ which includes the sequence of AI models is the input. We aim to find the optimal server positions $p_s = (p_1, p_2, \cdots, p_m)$ for each AI model in s to place, following the stochastic policy $\pi_\theta(p_s|s)$. Since NS chains have different lengths $m \in (1, 2, \cdots, M)$, we adopt a sequence-to-sequence model following [2], [14] which is devised to perform optimization problems with chains of different lengths.

The sequence-to-sequence (S2S) model adopts an encoder-decoder architecture featuring stacked Long Short-Term Memory (LSTM) cells, as illustrated in Fig. 1. The decoder, following the attention LSTM model proposed by Bahdanau et al. [15], aligns with the input sequence's length, producing host placement outputs for each introduced component (i.e., AI model) from the encoder. The decoder's hidden state $\rho_v = f(\rho_{v-1}, \overline{\rho}_{v-1}, c_t)$ evolves based on its prior state and attention over the encoder's hidden states. The context vector c_t is computed as the sum of weighted hidden states of the input sequence, determined by alignment scores. Specifically, for the decoder's output step v, the context vector is given by the equation:

$$c_v = \sum\nolimits_a \lambda_{(v,a)} \cdot \overline{\rho}_a, \tag{11}$$

where $\lambda_{(v,a)}$ defines the weight of each source hidden state in the decoding process. This variable-size alignment vector aligns with the source sequence length and is calculated by scoring the current target hidden state of the decoder ρ_v with each source hidden state $\overline{\rho}_a$:

$$\lambda_{(v,a)} = softmax\big(\omega \tanh(\xi_v \rho_v + \xi_a \overline{\rho}_a)\big), \tag{12}$$

where ω, ξ_v, and ξ_a are weight matrices learned in the alignment model. The prediction baseline $b_\theta(s)$ is determined by an auxiliary network, specifically an LSTM encoder linked to a multilayer perceptron (MLP) output layer. This network predicts the penalized energy cost produced by the agent following the current policy, serving as a value approximator based on the environmental state.

For S2S neural network learning, we use a RL environment as shown in Fig. 2. Given an AI model based service chain as the state, the S2S model is used for the policy network to generate the placement vector (the action). And then, the network environment can give the reward for S2S model training. The constraints shown in Eqs. (6)-(10) are set to the environment, so the model can determine whether the constraints are met by interacting with the environment.

B. Orthonormal Certificates

The Orthonormal Certificate, as introduced in [4], serves as an epistemic uncertainty estimator specifically designed for neural networks. This set of certificates, denoted as $C = (C_1, \cdots, C_k)$, operates on a deep neural network model expressed as $y = f(\phi(x))$, where ϕ denotes a deep feature extractor from the input data x, and f represents the neural network function. The high-level representations of training samples are organized into a dataset denoted as $\Phi = \phi(x_i)_{i=1}^n$, where n signifies the total data volume. Each training certificate C_j is essentially a straightforward neural network trained to map the dataset Φ to zero, accomplished through the minimization of a loss function l_c. As defined in [4], the estimation of epistemic uncertainty, denoted as $u_e(x)$, can be expressed as follows:

$$u_e(x) \overset{\text{def}}{=} \left\| C^{\mathrm{T}} \phi(x) \right\|^2, \tag{13}$$

where the calculation details of C can be obtained from [4].

TABLE II: Network Infrastructure Environment Settings

Network Environment with 10 Host Servers:

Host No.	1	2	3	4	5	6	7	8	9	10
CPU / GPU Capacity	10	9	8	7	6	6	6	6	6	6
Disk Size (GB)	10	10	10	8	8	8	6	6	6	6
Link Bandwidth (Mbps)	1000	1000	500	400	300	300	300	300	300	300
Link Lantency (ms)	30	50	10	50	50	50	50	50	50	50

Network Environment with 20 Host Servers:

Host No.	1	2	3	4	5	6	7	8	9	10
CPU / GPU Capacity	10	10	9	9	8	8	7	7	6	6
Disk Size (GB)	10	10	10	10	10	10	8	8	8	8
Link Bandwidth (Mbps)	1000	1000	1000	1000	500	500	400	400	300	300
Link Lantency (ms)	30	30	50	50	10	10	50	50	50	50
Host No.	11	12	13	14	15	16	17	18	19	20
CPU Capacity	6	6	6	6	6	6	6	6	6	6
Disk Size (GB)	8	8	6	6	6	6	6	6	6	6
Link Bandwidth (Mbps)	300	300	300	300	300	300	300	300	300	300
Link Lantency (ms)	50	50	50	50	50	50	50	50	50	50

TABLE III: AI Model Profiles

AI Model No.	1	2	3	4	5	6	7	8
Required CPU / GPU Capacity	4+Δ_1	3+Δ_2	3+Δ_3	2+Δ_4	2+Δ_5	2+Δ_6	1+Δ_7	1+Δ_8
Required Disk Size (GB)	4+Θ_1	4+Θ_2	3+Θ_3	3 + $\Theta_4$1	2+Θ_5	2+Θ_6	1+Θ_7	1+Θ_8
Required Bandwidth (Mbps)	100	80	60	20	20	20	20	20
Required Lantency (ms)	100	80	60	20	20	20	20	20
Task Completion Rate (%)	80+Υ_1	80+Υ_2	80+Υ_3	80+Υ_4	80+Υ_5	80+Υ_6	80+Υ_7	80+Υ_8

In our task, the training process of AI models generates some uncertain factors such as the CPU consumption uncertainty $f_c(t)$, the GPU consumption uncertainty $f_g(t)$, the storage usage uncertainty $f_d(t)$, and the model performance uncertainty $f_q(t)$. To estimate these uncertainties, we add the OCs-based estimator as shown in Eq. (13) to the S2S decoder as illustrated in §IV-A. Specifically, we consider the uncertainty influence into the context vector c_v as follows:

$$c_v = \sum_a \lambda_{(v,a)} \cdot \overline{\rho}_a \cdot F\big\{u_e(x_a)\big\}, \tag{14}$$

where $F\big\{u_e(x_a)\big\}$ is the fuzzy logic-based uncertainty embedding which we will detail in the next section.

C. Fuzzy Logic

For the input NS chain $s = (a_1, a_2, \cdots, a_m)$, we obtain the epistemic uncertainty estimation u_e for each AI model based on the formulation in Eq. (13). As detailed in §IV-A, the context vector c_v encapsulates alignment scores for AI models. Thus, we use fuzzy logic to incorporate the uncertainty estimation u_e into the context vector c_v, effectively accounting for uncertainties in AI model placement.

1) Fuzzy Representation for Uncertainties

Given the epistemic uncertainty estimation u_e, we denote l_u as the layer number, f_i^u is the i-th node, and o_i^u is the corresponding fuzzy output. To compile uncertainty into a neural network, we adopt the fuzzy representation proposed in [16] to assess the degree to which an input node belongs to a specific fuzzy set. Therefore, the i-th fuzzy neuron $n_i(\cdot) : R \to [0,1]$ maps the k-th input as the fuzzy degree:

$$o_i^u = n_i\left(f_i^u\right) = e^{-(f_i^u - \mu_i)^2/\sigma_i^2} \quad \forall i. \tag{15}$$

The Gaussian membership function with mean μ and variance σ^2 is utilized in this representation.

2) Fusion Part

TABLE IV: Experimental Results of Our Method with Different Number of NS Requests, Hosts, and AI Models of a Service Request

No. of NS Requests	No. of Hosts	Service Length	NS Request Accept Ratio	No. of NS Requests	No. of Hosts	Service Length	NS Request Accept Ratio
64	10	12	98.4%	128	10	12	97.6%
		14	85.9%			14	82.0%
		16	46.9%			16	43.8%
		18	12.5%			18	12.5%
	20	20	95.3%		20	20	95.3%
		24	93.7%			24	92.2%
		28	70.3%			28	68.7%
		30	35.9%			30	37.5%

To combine uncertainty estimations with context vectors, we also need to convert the context vector c_v into high-level representations. Therefore, the representation output o_i^c for the i-th node f_i^c of the context vector c_v is given as follows:

$$o_i^c = \frac{1}{1 + e^{-f_i^c}}. \tag{16}$$

Then, we can fuse uncertainty estimations with context vectors as follows:

$$F\big\{c_v, u_e\big\} \overset{\text{def}}{=} w_i^c o_i^c + w_i^u o_i^u + b_i, \tag{17}$$

where w_i^c and w_i^u are the weight coefficients and b_i is the bias coefficient.

V. Evaluation

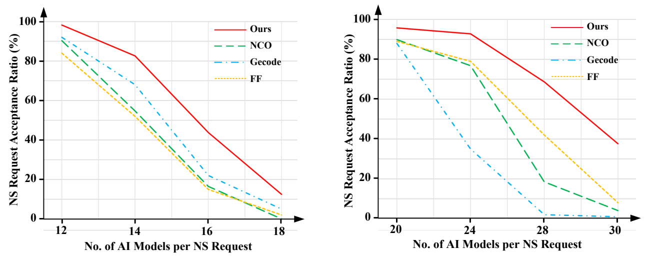
A. Experimental Settings

In this paper, for the sake of illustration, we consider the infrastructure with two different sizes with 10 host servers and 20 host servers respectively for evaluation; the proposed solution is able to handle any network size. The detailed settings of network infrastructure environments are given in Table II. The AI model profiles are given in Table III, where Δ_i, $i \in \{1, 2, \cdots, 8\}$, Θ_j, $j \in \{1, 2, \cdots, 8\}$ and Υ_k, $k \in \{1, 2, \cdots, 8\}$ are the uncertain variables randomly generated by two distributions including the "normal" distribution and the "uniform" distribution using Python "numpy" library. Power consumption parameters in Eq. (5) are given as follows: $W_h^c = W_h^g = 200$, $W_h^e = 100$ and $W_{net} = 0.1$. For model parameters, the learning rate is set to 0.0001, the LSTM layer number is set to $(1, 3, 4)$ and the LSTM hidden size is set to $(32, 64, 128)$ according to [2], [14].

B. Comparison Baselines and the Evaluation Metrics

Baselines: We evaluate our method in comparison with three baseline models, including NCO [2], Gecode solver [17] and First Fit (FF) heuristic algorithm [18]. The NCO is a deep RL method for VNF placement Optimization without uncertainty considerations. The Gecode solver is an algorithm based on the classical Branch and Bound paradigm. The FF algorithm is designed with the heuristic first fit strategy.

The Metric: The NS request acceptance ratio is used as the evaluation metric in the experiment which is defined as the ratio of the number of successfully placed requests to the total number of requests. For example, if 100 requests are received and the number of successfully placed is 50, then the acceptance ratio is 50%.



(a) (b)

Fig. 3: Comparisons with NCO [2], Gecode [17] and FF [18] with 128 NS requests in (a) 10 hosts and (b) 20 hosts.

C. Model Performance with Different Requests

To evaluate the proposed model, we first generate 64 NS requests to place them one by one. With the network environment with 10 hosts, we evaluate a chain consisting of 12, 14, 16, and 18 AI models in sequence. With the network environment with 20 hosts, we choose a chain consisting of 20, 24, 28, and 30 AI models. From the results in Table IV, we can see that our model can achieve more than 90% successful placement for AI models with the considered uncertainties at ⟨ Length 12, Host number 10 ⟩ and ⟨ Length 20 and 24, Host number 20 ⟩ for both 64 and 128 requests. This illustrates that our model has good deployment capabilities for AI models with uncertain properties. To reflect the level of deployment capabilities, we conduct comparative experiments between the proposed method and other related models in the next section.

D. Comparison with Other Models

To further validate the proposed approach, we compare our model with the NCO [2], Gecode solver [17] and the First Fit (FF) heuristic algorithm [18]. The experiment is conducted using 128 NS requests, and the results are shown in Fig. 3. The results illustrate that the placement successful ratio of our method is significantly better than that of the other three methods when dealing with network service requests with different number of AI models, because our model has an estimate of the AI model uncertainties and takes it into account in resource allocation.

VI. Conclusion

This paper addressed the critical challenge of AI model placement in next-generation networks, particularly in the context of 6G, where VNFs heavily rely on AI implementation. The inherent uncertainties introduced by AI models, including varying computing resource requirements, changing performance, and unpredictable storage requirements, make optimal model placement more complex compared to conventional software based VNFs. To tackle this problem, we proposed a novel neural combinatorial optimization strategy, employing a S2S neural network trained in an RL environment. To handle the introduced uncertainties, we incorporated OCs and fuzzy logic as uncertainty estimators to enhance the S2S model. Our experimental results demonstrate the effectiveness of the proposed strategy, showcasing competitive performance in addressing the challenges associated with AI model placement in the presence of epistemic uncertainties. This research contributes valuable insights to the ongoing development of next-generation networks by providing a robust and adaptive solution for optimal AI model placement.

Acknowledgements

This work has been partially sponsored by the UK GOV DSIT (FONRC) project REASON.

References

[1] W. Attaoui, E. Sabir, H. Elbiaze, and M. Guizani, "Vnf and cnf placement in 5g: Recent advances and future trends," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2023.
[2] R. Solozabal, J. Ceberio, A. Sanchoyerto, L. Zabala, B. Blanco, and F. Liberal, "Virtual network function placement optimization with deep reinforcement learning," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 292–303, 2020.
[3] A. Marotta, F. D'andreagiovanni, A. Kassler, and E. Zola, "On the energy cost of robustness for green virtual network function placement in 5g virtualized infrastructures," *Computer Networks*, vol. 125, pp. 64–75, 2017.
[4] N. Tagasovska and D. Lopez-Paz, "Single-model uncertainties for deep learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.
[5] Y. T. Woldeyohannes, A. Mohammadkhan, K. K. Ramakrishnan, and Y. Jiang, "Cluspr: Balancing multiple objectives at scale for nfv resource allocation," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1307–1321, 2018.
[6] Y. Chen and J. Wu, "Latency-efficient vnf deployment and path routing for reliable service chain," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 1, pp. 651–661, 2021.
[7] F. Malandrino, C. F. Chiasserini, G. Einziger, and G. Scalosub, "Reducing service deployment cost through vnf sharing," *IEEE/ACM Transactions on Networking*, vol. 27, no. 6, pp. 2363–2376, 2019.
[8] H. Ren, Z. Xu, W. Liang, Q. Xia, P. Zhou, O. F. Rana, A. Galis, and G. Wu, "Efficient algorithms for delay-aware nfv-enabled multicasting in mobile edge clouds with resource sharing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 9, pp. 2050–2066, 2020.
[9] V. Mai, K. Mani, and L. Paull, "Sample efficient deep reinforcement learning via uncertainty estimation," *arXiv preprint arXiv:2201.01666*, 2022.
[10] X. Wang, C. Wu, F. Le, and F. C. Lau, "Online learning-assisted vnf service chain scaling with network uncertainties," in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, 2017, pp. 205–213.
[11] Y. Xie, S. Wang, B. Wang, and L. Luo, "Migration aware virtual network function placing and routing in uncertain environment," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6.
[12] S. Agarwal, F. Malandrino, C.-F. Chiasserini, and S. De, "Joint vnf placement and cpu allocation in 5g," in *IEEE INFOCOM 2018-IEEE conference on computer communications*. IEEE, 2018, pp. 1943–1951.
[13] J. Pei, P. Hong, M. Pan, J. Liu, and J. Zhou, "Optimal vnf placement via deep reinforcement learning in sdn/nfv-enabled networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 263–278, 2019.
[14] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.
[15] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.
[16] Y. Deng, Z. Ren, Y. Kong, F. Bao, and Q. Dai, "A hierarchical fused fuzzy deep neural network for data classification," *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 4, pp. 1006–1012, 2017.
[17] C. Schulte, M. Lagerkvist, and G. Tack, "Gecode," *Software download and online material at the website: http://www. gecode. org*, pp. 11–13, 2006.
[18] S. Kumaraswamy and M. K. Nair, "Bin packing algorithms for virtual machine placement in cloud computing: a review," *International Journal of Electrical and Computer Engineering*, vol. 9, no. 1, p. 512, 2019.