VNF placement at edge devices using Gurobi and Artificial Neural Network

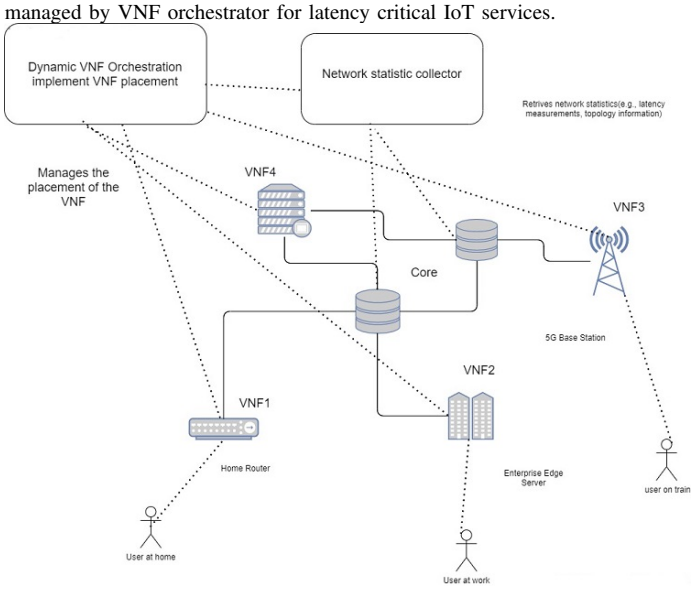
***Abstract*—VNF in edge devices is used to improve the response time, to avoid excessive use of core network and reduction in use of user-VNF end to end latency to a considerable extent, while holding Internet of things services in context to Network Function Virtualization (NFV). Many approaches have been proposed for VNF, although, the major concern is to reduce the numbers of servers run in a chain of VNFs for a particular service without putting considerations into network conditions, for example latency. In this paper, we utilize the optimal edge VNF placement problem as an Integer Linear Programming model that ensures minimum end to end latency and results in quality service without violating acceptance limit of latency. Latency beyond certain limits can be the reason for disruption and degradation in Internet of Things services. With the present time complexity of existing VNF placement algorithm being NP Hard, a new proposal of VNF placement strategy using Artificial Neural Network (ANN) trained by assignments solutions generated by Integer Linear Programming (ILP) model of optimal edge VNF placement method for smaller instances of VNFs. This method resolved problems for VNF assignment problems of edge devices with more number of VNFs with reducing time complexity to be linear and providing similar results as of ILP model in context to latency. To construct a completely virtualized environment, Virtual Network Functions (VNFs) can be linked or combined together as building blocks. On top of the hardware networking infrastructure, VNFs run on virtual machines (VMs). Virtual Network Functions (VNFs) put close users on the edge devices increase response time, prevent redundant usage of the core network, and reduce end-to-end user-to-VNF latency to a high ext latency. In this paper, we are using Gurobi to find the optimal VNFs placed on host by Integer Linear Programming(ILP). Later using the Artifical Nueral Network the best host with VNFs is compared.**

***Index Terms*—Virtual Network Function, Artificial Neural Network, Integer Linear Programming, deployment, edge devices.**

I. Introduction

In recent years, the penetration of Content Delivery Networks (CDN) has shown the advantages of implementing application-aware intelligence in the network. Continuing in this direction and building on Software-Defined Networking (SDN), it is now anticipated that future telecommunications networks will support low-latency, context-aware and user-specific services (also known as "value-added" services) that carry large volumes of traffic while handling them in a highly scalable and efficient way [1]. However, advances in virtualization technology allow these network functions to operate on commodity servers as software-only applications, and such network functions are called Virtualized Network Functions (VNFs). Network functions, such as firewall, session boundary controller, etc., are typically installed by network service providers on purpose-built physical appliances. It is difficult to implement and manage these hardware-dependent network functions and they are connected to a specific service, slowing down the introduction of new network services [2]. However, advances in virtualization technology allow these network functions to operate on commodity servers as software-only applications, and such network functions are called Virtualized Network Functions (VNFs).

Fig. 1. A high level architecture of VNF Placement at IoT edge devices, managed by VNF orchestrator for latency critical IoT services.



Industry efforts were initially carried out under the Network Function Virtualization (NFV) rubric (ETSI, n.d.) to express and standardize the context and support structures for such VNFs and later in an open source context inside community projects such as OPNFV (OPNFV: Open Platform for NFV), OSM (OpenSource MANO: OSM), and ONAPP (ONAP: Open Network Automation Platform). A modern way to plan, deploy, and control network services enables the execution of network functions as VNFs. As a network-related and hardware-independent software program, we describe a VNF. Technologies such as microservice, middleware, and distributed applications are all included in the scope of this analysis with this definition, even though the above-mentioned technologies existed before the VNF word appeared. It is anticipated that virtualization will accelerate service innovation, provide organizational flexibility and agility, and allow service providers to increase capital and operational efficiencies [3].

Lakehead University

NFs are the implementation of the program of a network service capable of operating over the NFV infrastructure. NFVI is the physical and/or virtual assets that enable the VNFs' implementation, and MANO includes the VNF and NFVI's orchestration and life cycle management. A standard deployment orchestration and life cycle administration is imposed for operational performance. Emerging network services (e.g. 5G-based) put substantial new demands on dynamic management networks to accommodate a dynamic, large-scale combination of a broader variety of services with different non-functional characteristics (e.g., reliability, power consumption).

Due to the complexities of service chains and the dynamics of emerging traffic trends, orchestration and positioning of VNFs has been one of the key NFV research challenges. Although several research projects have suggested solutions for static VNF placement (e.g. [4], [5]), we are unaware of any previous studies on VNF placement in a distributed edge NFV infrastructure. Consequently, most NFV resource orchestration schemes only calculate placement for a single service chain based on a static network state (topology and traffic) and do not recalculate placement to migrate VNFs as these temporal network properties change. Owing to a lack of dynamic orchestration and re-calculation of VNF placement, users often encounter "latency violations," in which VNFto-user E2E latency exceeds a desired value (for example, a video cache VNF has an upper bound of 100 ms for VNF-to-user E2E latency, but the real VNF-to-user E2E latency increases to 130 ms for a period of time due to network congestion). However, since VNF migrations cause temporary service interruption and high network overhead when moving their state through physical hosting devices, the number of VNF migrations caused by re-optimizing the location of VNFs must be held to a minimum [6].

Recent research work has shown placing VNFs based on the decision of Artificial Neural Network (ANN) trained by optimal solutions approach [7]. It is first used to train an artificial neural network using the optimal edge VNF placement VNF assignment data, and then it is used to test the artificial neural network-based placement method using larger test instances with a large number of VNFs. Our solution gives the best edge VNF placement technique, which often guarantees the shortest total latency between users and VNFs. The optimal edge VNF placement strategy, which always ensures the complete minimum latency from users to VNFs, was studied in this paper [8]. The Integer Linear Programming(ILP)-based ANN-based approach for assigning VNFs to edge computers, where the training process includes optimal assignment solutions of different VNF and host pairs. Regardless of the size of training data samples, the VNF placement via ANN algorithm shows promising results in terms of latency while reducing the time complexity to be linear when compared to the optimal VNF placement process.

The following is how the rest of the paper is organised. Section I tell us about the introduction of the work done in this research paper. The flow of the related work has been shown in Section II. The system model is explained along with the mathematical model in Section III. In Section IV we will discuss the VNFs based on the decision of Artificial Neural Network to get an optimal solution. Section V contains the simulation results as well as an evaluation of the success of both placement strategies.At last, Section VI brings the paper to a conclusion.

II. Related Work

NFV stems from a change in mindset toward the virtualization of network components that are currently housed on hardware platforms. The consideration of the data forward plane and control plane becomes more important as networks evolve toward NFV, making it easier to handle existing services and create new ones. Many works, in particular, address the VNF-PEC problem with the aim of lowering the SC deployment cost. The computational and communication resources that a SC requires in order to be provisioned are usually expressed as deployment cost in the traditional VNF placement problem [9].

In inter-datacenter (inter-DC) elastic optical networks, investigation on how to implement VNF service chains (VNF-SCs) and perform task scheduling for bulk data transfers (EONs) [10]. In the case of the VNF-PEC problem [11] , the same goal was pursued. However, there may be monetary costs associated with leasing MEC and Cloud services [12] [13], which may vary depending on who owns the infrastructures (for example, various pricing models used by MEC and Cloud providers).

Another big problem during the SC deployment and VNF installation is the perceived contact delay. Until recently, the propagation delay was the most critical factor in minimising the overall delay [14]–[16]. However, there are other big delay contributors that must be considered in the sense of NFV. The processing time is another delay factor that should be kept to a minimum. This processing delay can be fixed [17] if the servers have sufficient ability to execute the VNFs, or it can be proportional to server utilisation [18]. Where the aim is to minimise the delay during VNF placement, both processing and propagation delays should be taken into account [19]–[21]. It's worth noting that the processing time requires a queueing time, which is heavily influenced by the number of users/IoT devices associated with each SC as well as their traffic rate [22]. State-space models paired with input controllers can be used to approximate the total processing delay in addition to queuing models [23]. All of these delays apply to all levels of IoT to Cloud communication and should be reduced together. Since many IoT applications are mission critical and delay sensitive, this is especially important in an IoT communication environment. Furthermore, the location of the VNFs in this context can have a huge effect on the overall delay [24]. VNFs may also be placed in infrastructure physical devices that are closer to the wireless devices to minimise contact delay [25]. Creating clusters of usable VNFs in the MEC layer, where the various requested SCs can be deployed, is one way to accomplish this [26]. The contact delay can be reduced by

reducing the number of clusters and correctly positioning the VNFs.

Another aim that is largely driven by infrastructure providers' requirements (e.g., MEC/Cloud providers) is to carefully assign and schedule available resources. When it comes to solving the VNF placement problem, path optimization, resource performance, and load balancing are all hot topics [27]. The aim is to increase resource availability, increase the number of SCs that can be deployed in the physical infrastructure at the same time, and simplify VNF placement of future SC requests to achieve the needed scalability. This target can be interpreted as minimising overall resource consumption in order to allow multiple heterogeneous SCs servicing heterogeneous IoT users to co-exist in the MEC layer [28,29] or minimising infrastructure resource idleness [30]. Load balancing can also be achieved by reducing bandwidth usage and minimising full connection utilisation [31]. This can be accomplished by using effective queuing and Quality of Service (QoS) modelling to minimise resource consumption during the optimization problem [32]. This is particularly important for the MEC provider because load balancing will keep its infrastructure from being overwhelmed, especially when it is dealing with a high volume of incoming traffic [33]. Furthermore, the manipulated traffic can be bursty and dynamically alter over time [30]. As a result, an effective load balancing scheme can comfortably handle any incoming traffic bursts and spikes. This solution has a number of advantages, including the avoidance of Service Level Agreement (SLA) breaches and the elimination of the need to use external Cloud services that could cross a potentially costly transport network.

If both Cloud and MEC resources are open, another goal may be to figure out how to best offload and redirect traffic between the two layers [34]. This is especially important for applications that produce a lot of data and need to make the best decision possible about where to process it. Of course, the MEC layer can lack the infrastructure required to provide networking services for such massive volumes of data.As a result, a first decision can be taken in the MEC to decide which traffic should be forwarded to the Cloud in real time so that the appropriate VNFs can be applied [35]. The choice of which VNFs to deploy to the Cloud is critical, as continuous data transmission to the Cloud will result in expense, privacy, and storage overheads [36]. When conducting traffic offloading and load balancing at the same time, another essential activity occurs. A vertical solution can be considered in this case by correctly offloading wireless traffic to a group of cloudlets located at the network's edge [37]. Real-time VNF placement solutions should be developed to optimise traffic offloading. This allows traffic to be diverted on the fly based on the aggregated traffic stream's computational and communication needs. The objective feature should take into account the complex existence of the streams and allocate VNFs wisely to efficiently offload traffic to the Cloud while meeting any position and latency requirements placed by the IoT application.

III. Placement of edge VNF with optimal solution

As an Integer Linear Program (ILP) to calculate the latency-optimal allocation of VNFs in next-generation edge networks, we introduce and formalize the Edge VNF placement issue in this section.As an Integer Linear Program (ILP), we implement and formalize the Edge VNF placement question to determine the optimal latency allocation of VNFs in next-generation edge networks. In edge devices or the distant cloud, the optimal Edge VNF placement strategy aims to find the optimal latency VNF assignments. Network providers first plan to position the VNFs closer to the consumer on edge devices to ensure maximum latency for users. If the edge computers, however, are out of room and do not support the VNFs, then the internal cloud of the provider is used to host the VNFs. VNFs can be hosted on any physical server, such as an edge system close to the user or in a distant cloud, thanks to recent developments in virtualization and NFV. The operators aim to position VNFs in close proximity to their users in order to have the best possible VNF-to-user E2E latency, by first using edge devices that are close to the user and falling back to hosting VNFs in the internal cloud of the provider when edge devices are out of capability.

A. System Model

All the variables used for the formulation are shown in TableI. We represent the physical network as an undirected Model

$$M = (Host, Link, User),$$

where Host, Link and User denote the set of hosts, the links between them, and the users in the topology, respectively.

$$Network = \{n_1^1, n_2^2, n_3^3 ..... n_n^n\}$$

Network indicate the network functions allocated to every VNF connected to each *User*. We assume that VNFs can be deployed on any physical host in this graph, implying that all physical hosts have the necessary CPU, memory, and I/O resources to support VNFs. Furthermore, we presume that all links have a physical bandwidth limit that is factored into the placement. We present requirements like memory for each network because there are different kinds of VNFs for example firewalls or resource-intensive Deep Packet Inspection modules. In addition to the compute requirements, all VNFs must meet the Service Level Agreements (SLA) of the service provider, which are denoted with an θ_i.

Similarly, all VNFs specify the bandwidth reservations that must be made along the path from their user (since some VNFs are heavily used by users, while others are not). A latency value of lat_m is assigned to each link in the network. Summing all the lat_m of links between a VNF n_i and a host h_j, which is represented by Lat_{ij}, gives the total latency from a user to VNF. In the case of VNF n_i^o (VNF associated with user U_o) being hosted at h_j, these bandwidth requirements are materialised using the derived parameter ban_ij, which denotes the bandwidth required along the path p_k. Finally we get a

TABLE I
TABLE OF PARAMETERS FOR THE SYSTEM MODEL

Network Parameters	Description
M = (Host, Link, User)	Undirected model of network
Host = { h_1, h_2,.....h_H }	Host available in the network
Link = { e_1, e_2.....e_M }	All the links in the network
User = { u_1, u_2.....u_U }	Users associated with the network
lat_{ij}	Latency of the networks
cap_j	Hardware capacity of the host
C_{ij}	Capacity of the link between the host and the user
Network Parameters	**Description**
Network = $\{n_1^1, n_2^2, n_3^3 ..... n_n^n\}$	Network function that needs to be allocated
req_i	Host requirement for each VNF $n_i \in$ Network
θ_i	Maximum latency tolerance limit of each VNF $n_i \in$ Network
Decision Variable	**Description**
x_{ij}	binary variable defining the best VNF and Host pair for the network
Derived Parameter	**Description**
Lat_{ij}	Total latency between the VNF and Host in a network
ban_{ij}	bandwidth required along the path

binary variable defining the best VNF and Host pair for the current network:

$$x_{ij} = \{_1^0 \tag{1}$$

B. Problem Formulation for Optimal Placement

We need to find an acceptable allocation of all VNFs that minimises the total expected end-to-end latency from all users to their respective VNFs, given a set of users U, a set of VNF hosts H, a set of individual VNFs N, and a latency matrix lat_{ij}. VNF placement at the host will give experince minimum latency over any user. The objective function of the model has been formulated using Integer Linear Programming(ILP) can be represented by:

$$min \sum_{n_i \in Network} \sum_{h_j \in Host} x_{ij} \times Lat_{ij} \tag{2}$$

The object function looks for yij values when adhering to the following constraints:

$$C1: \sum_{n_i \in Network} x_{ij} \times req_i \leq cap_j, \forall_{h_j \in Host} \tag{3}$$

$$C2: \sum_{h_i \in Host} x_{ij} \times Lat_{ij} \leq \theta_i, \forall_{n_i \in Network} \tag{4}$$

$$C3: \sum_{h_i \in Host} x_{ij} = 1, \forall_{n_i \in Network} \tag{5}$$

$$C4: \sum_{h_i \in Host} x_{ij} \times ban_{ij} \leq C_{ij}, \forall_{n_i \in Network} \tag{6}$$

$$C5: x_{ij} \in \{0, 1\} \tag{7}$$

Due to finite hardware resources, we can only host a limited number of VNFs on each host. Constraint (1) ensures that hosts' hardware constraints are respected. While deploying latency-sensitive VNFs, Constraint (2) ensures that the user's maximum tolerance delay threshold is not exceeded. Constraint (3) states that each virtual network function (VNF) must be assigned to exactly one of the hosts (for example, a nearby edge device or the Cloud). As a result, each VNF can be assigned to only one of the hosts, which can be local edge devices or the cloud. Thus constraint (4) considers the bandwidth requirements of virtual network functions (ban_{ij}) and link capacities (cap_j) along a path, it ensures that none of the physical links along the path are overloaded. Lastly, Constraint (5) denotes that the value of decision variable x_{ij} will be either 0 or 1, depending on whether n_i is hosted at h_j or not. The value of x_{ij} will be 1 if n_i is hosted at h_j; otherwise, it will be 0. All of the parameters discussed above were adapted from previous research [8] [40]–[42].

IV. VNF PLACEMENT USING ILP

The ILP model in the experiment takes randomly generated values as in Algorithm 1. These generated values are hence used to get the optimized values of the VNF placement to host h_j. Using gurobi ILP placement strategy, we get a tuple value of optimized values of placed VNF. As mentioned earlier, the constraints give you a optimized latency VNF placement. The experiment gives each VNF n_i to Host h_j. So if there are 10 host and 100 VNFs, the 10 host gets the best VNF placed that reduces the latency. The ILP model has the worst case due to the exponential timeit takes to get the solution. Hence, we use the ANN to find the next solution in the experiment.

V. VNF PLACEMENT USING ANN

The ILP model takes exponential time in the worst case [38], which is inefficient when dealing with a large number of VNFs for huge IoT services. According to a research [39] [42], in terms of the number of samples, an ANN has linear time complexity. As a result, using an ANN to predict the best VNF placement solution has the potential to reduce optimization time and improve user experience.

Algorithm 1: Stimulation and training

Result: Final matrix D_{ij}

$D_{ij} \leftarrow \{\}$
for *epoch < cases of networks* **do**
  Randomly generate systems : Lat_{ij}, ban_{ij}, C_{ij}, cap_j, req_i, θ_i
  x_{ij} is an optimized value of the problem defined using the constraint solve by Integer Linear Programming (ILP)
  for *i = 1,2,3,...Network* **do**
    j $\leftarrow argmax(x_{ij})$
    k $\leftarrow$ get a random number from {1,2...m} - {j}
    D $\leftarrow$ D $\cup\{(X_{ij},1),(X_{ik},0)\}$
    $cap_j \leftarrow cap_j$ - req_j
    $C_{ij} \leftarrow C_{ij}$ - ban_{ij}
  end for
end for
Normalize each feature in Xij where (X_{ij} , x_{ij}) $\in$ D between 0 and 1.0
Train the ANN on every (X_{ij} , x_{ij}) pair in D

We use an existing ILP solver to optimise the simulated VNF placement problems and record the solutions in order to generate labelled data for ANN training. Smaller instances, such as 10 to 45 hosts and 100 to 450 VNFs, have been considered for training the ANN. In section VI, you can find more information about the training data and simulation parameters.

Algorithm 2: VNF placement via ANN

Input: Lat_{ij}, ban_{ij}, C_{ij}, cap_j, req_i, θ_i
Result: R
$R \leftarrow \{\}$
for *i = 1,2,3,...Network* **do**
  $C \leftarrow$ array of H elements
  for *j = 1,2,3,...m* **do**
    if $Lat_{ij} > C_{ij}$ ***or*** $req_i > cap_j$ ***or*** $ban_{ij} > \theta_i$ **then**
      $M_j \leftarrow 0$
    else
      $M_j \leftarrow$ fun(X_{ij})
    end if
  end for
  k $\leftarrow argmax_{ij}$ M_j place n_i on h_k)
  $cap_k \leftarrow cap_k$ - req_i
  $C_{ik} \leftarrow C_{ik}$ - ban_{ik}
end for

If we describe the ANN as a function fun, we get fun(X_{ij}) = γ_{ij}, where 0 $\leq\gamma_{ij} \leq$ 1.x_{ij} is the training lable for x_{ij}. To reduce the mean squared error between x_{ij} and γ_{ij}, we optimise the ANN parameters. The aim of the ANN is to solve the VNF placement problem in a sequential manner. The ANN's input features are defined as an octuple X_{ij} = ($\{cap_k$—1 $\leq k \leq H\}, \{\theta_{ik}$—1 $\leq k \leq H\}, \{Lat_{ik}$—1 $\leq k \leq H\}, Lat_{ij}$, req_i, C_i, ban_{ij}, θ_{ij}), where i and j denote the current VNF n_i and the host h_j, respectively. As a consequence, we interpret fun(X_{ij})'s performance as a trust score for placing VNF n_i on host h_j. The simulation and training phase is depicted in Algorithm 1. Algorithm 2 depicts the process for leveraging the qualified ANN to optimise VNF placement.

VI. EXPERIMENTAL RESULT

Fig. 2. Running time comparison for 100 hosts

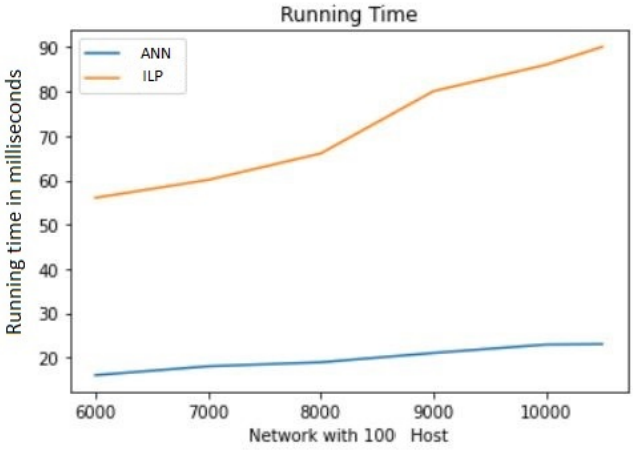
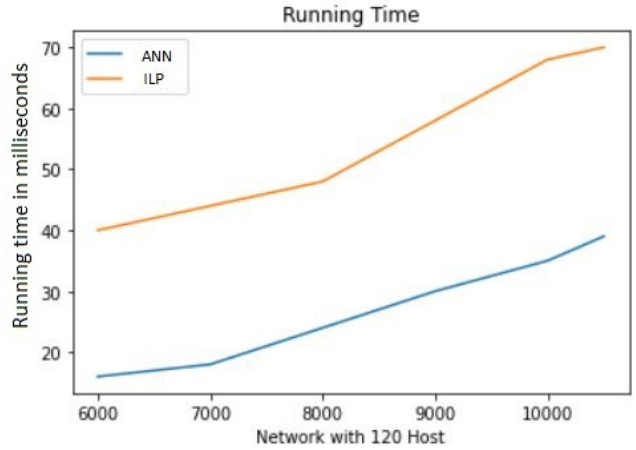


Fig. 3. Running time comparison for 120 hosts



The optimal edge VNF placement has been implemented, and an ANN has been trained with the optimal VNF to host (edge devices available in the MEC or IoT Layer) assignment solutions provided by the optimal edge VNF placement. The performance of our proposed ANN placement strategy is then compared to that of the optimal edge VNF placement and a gurobi ILP placement strategy in terms of latency and running

Fig. 4. Running time comparison for 140 hosts

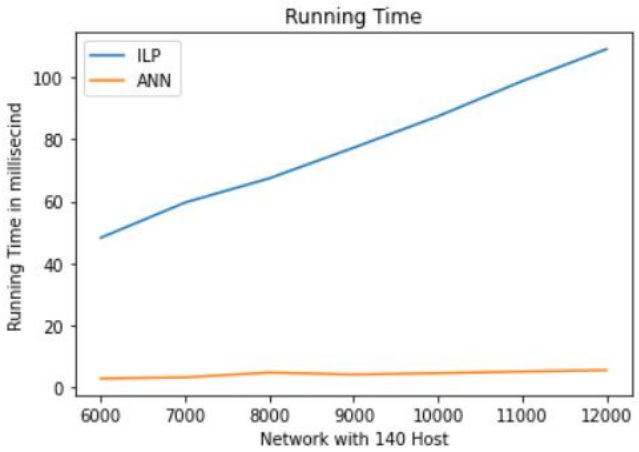
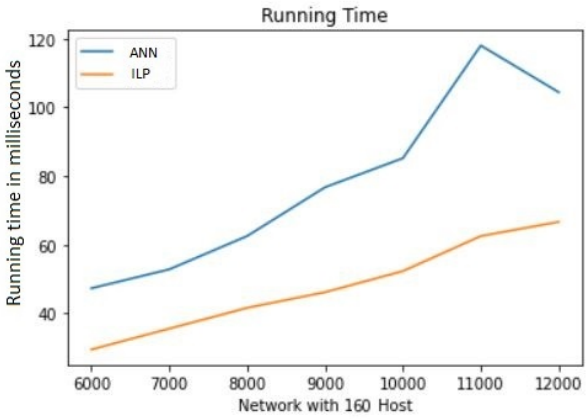


Fig. 5. Running time comparison for 160 hosts



time. The gurobi ILP VNF assignment method will always choose the best available VNF to host assignment at the time, which may turn out to be very latency inefficient in the long run. At each point, the greedy placement strategy selects the n_i and h_j pair with the locally optimal latency Lat_{ij} from the set of candidate solutions. The gurobi ILP model, on the other hand, could fail to find the best solution, possibly even offering the worst possible VNF to host assignment, resulting in significantly higher total latency than in the best case.

While optimal edge VNF placement using ILP often results in better latency benefits, the time complexity remains exponential for large numbers of VNFs and hosts [43]. VNF placement using the proposed ANN technique, on the other hand, reduces the time complexity to linear, lowering computational costs significantly for large-scale VNFs to hosts assignments.

For running the simulations, we have used an Intel(R) Core(TM) i5-1035G1 CPU for 1.00GHz to 1.19 GHz machine with UHD Graphics 4GB Memory. We used the GPU to run the placement algorithm in the case of ANN placement experiments. This proposed placement method's efficiency can be enhanced even further by running it on several CPU threads in parallel. Since different IoT services can have different tolerable latency limits C_{ij}, the latency violation limit of each VNF I has been generated between 30 and 100 milliseconds.

Fig. 6. Look of the raw data ANN

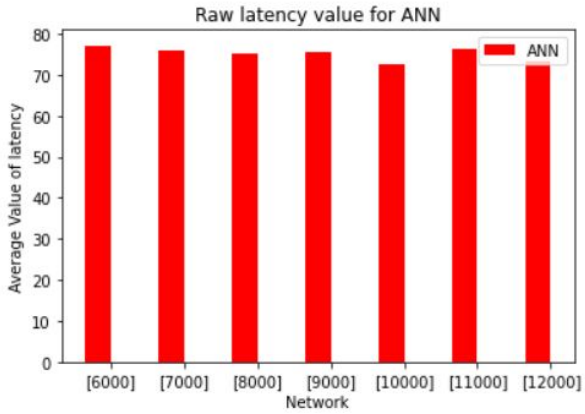
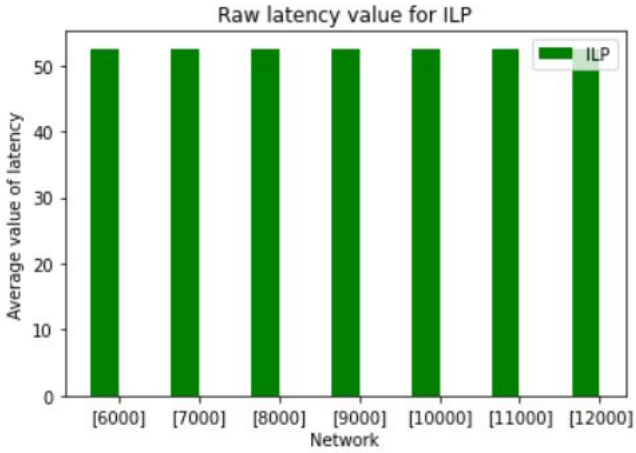


Fig. 7. Look of raw data ILP



Due to the dynamic power ownership of different IoT devices, the values of VNF requirements (req_i) and host capacity (cap_j) in case VNF n_i is put at h_j have been considered to be in the range of 10 - 50 and 10 - 800, respectively. Furthermore, a VNF n_i installed on host h_j can have a bandwidth requirement (ban_{ij}) of 5 to 10, whereas the hosts' bandwidth capacity (θ_{ij}) is 50 to 100. Due to the fact that the latency values on the links differ for a variety of reasons, Lat_{ij} values have been produced at random and range from 5 to 100 milliseconds. All of the parameters listed have been adapted from previous work [40]–[42]. Fig. 6-9 depicts the cumulative latency deviation of the greedy placement strategy and the ANN-based placement strategy from the optimum edge placement in percentage scale with respect to the delay variance. In this case, latency variance refers to the difference between the optimum latency value obtained by ILP and the latency values of ANN and greedy placement mechanisms. This experiment was performed with 100, 120, 140, 160 hosts, respectively, in accordance with 6000, 7000, 8000, 9000, 10000, and 12000

Fig. 8. Comparision of latency deviation for 100 host

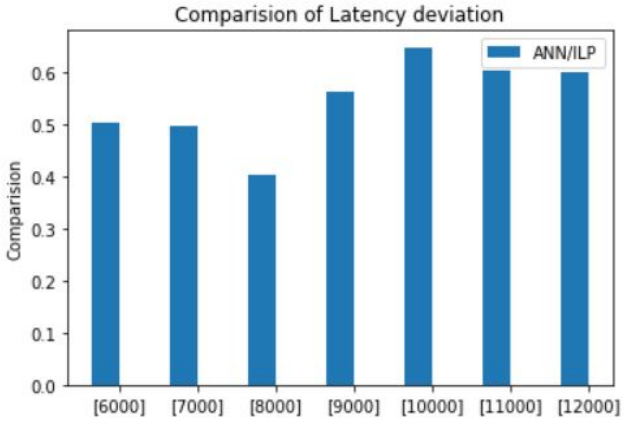


Fig. 9. Comparision of latency deviation for 120 host

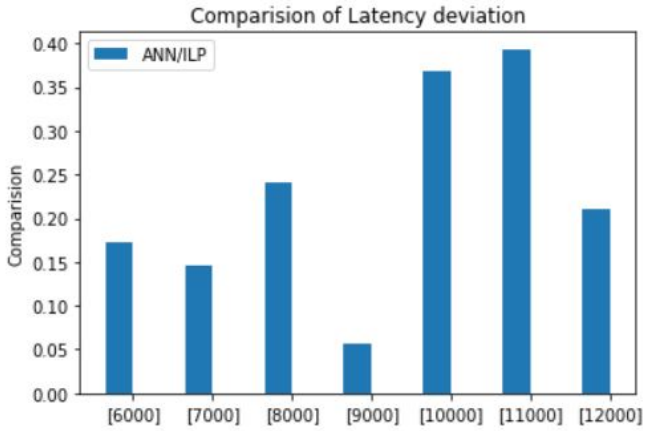


Fig. 10. Comparision of latency deviation for 140 hosts

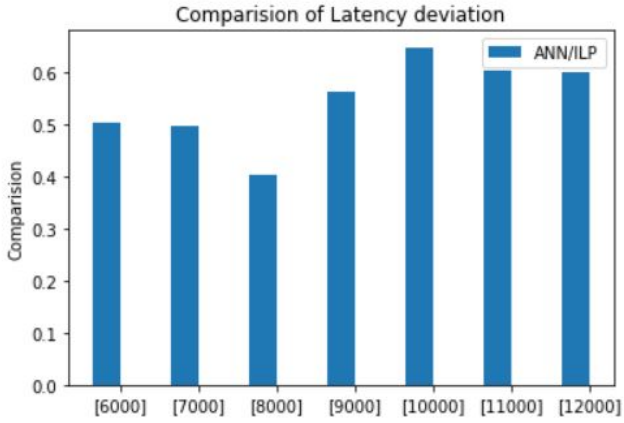
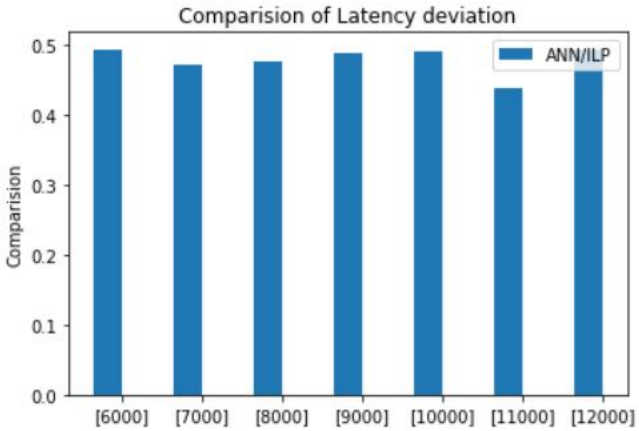


Fig. 11. Comparision of latency deviation for 160 hosts



VNFs. As compared to the optimal edge placement strategy, the ANN placement strategy shows promising total latency results that are similar to the optimal, with a delay deviation from the optimal average latency ranging from to 0.5%. The gurobi ILP placement method, on the other hand, raises the total latency variance from the optimal case by up to 27% while producing at least three times more total latency than VNF placement using ANN.

We set the host numbers to 100, 120, 140, and 160 for the next experiment, with the number of VNFs ranging from 6000 to 10000. Fig 2-5 As compared to the ideal edge VNF placement with an increasing number of VNFs, the execution time for ANN placement strategy with GPUs increases the running time by a large amount (50 percent - 60 percent for ANN placement strategy using two GPUs and 65 percent - 85 percent for ANN placement strategy using four GPUs). Though managing large scale VNF placement, the patterns in the increase of running time for optimal edge placement illustrate why it performs poorly in terms of running time compared to VNF placement using ANN. In terms of percentage, the (Gurobi - ILP)/ILP and (ANN-ILP)/ILP reflect the discrepancies or deviations of latency values obtained by greedy and ANN placement strategies from optimal latency values found by ILP. The number of VNFs used in this experiment ranged from 6000 to 10000, which is substantially more than the instance size of VNFs used in the training process. In contrast to the ILP solver, the total latency encountered when positioning all of the VNFs using the ANN strategy provides very fair cumulative latency results (0.0 percent - 0.6 percent deviated from optimal only). On the other hand, the gurobi ILP placement strategy will result in a average latency of up to 40-60 from ILP, which is less than the average latency variance caused by ANN. As a result, as the number of VNFs increases, an ANN-based placement strategy achieves very similar total latency results as the best edge VNF placement strategy while maintaining a much shorter execution period by making faster placement decisions.

VII. CONCLUSION

We argued in this paper that, in order to maintain low end-to-end latency from users to their VNFs in a constantly changing network environment, VNFs must be moved to optimal locations multiple times over the course of an NFV system's lifetime. The goal has been to develop a quicker al-

ternative to the ILP model of optimal edge placement. Because realtime IoT applications (such as Storage Incompatibility Detection, Perimeter Access Control, Forest Fire Detection, Smart Roads, Patient Surveillance, and Ultraviolet Radiation Detection) require the VNF orchestrator to make VNF to host assignment decisions as quickly as possible, the VNF orchestrator is required. On a simulated nationwide network topology with real-world latency features and various latency-sensitive VNFs running, we tested the proposed VNF orchestrator. As a result, we used Integer Linear Programming(ILP) to solve the optimal edge VNF placement problem. Because this placement strategy takes exponential time, the trade-off here is between time complexity and latency minimization to ensure optimal latency. As a result, we developed a new placement approach based on the optimal VNF to host assignments created by the ILP model of optimal edge VNF placement for smaller instances, where the training phase is completed based on the optimal VNF to host assignments generated by the ILP model of optimal edge VNF placement for smaller instances [42]. When dealing with a large number of VNFs and hosts, this placement technique is better because the time complexity is linear, which greatly reduces the computational complexity. Additionally, this placement strategy achieves promising results that are similar to the optimum edge placement method's latency minimization capability.

Artificial neural networks can be used for similar optimization research problems, as evidenced by the promising results in terms of reducing the running time of VNF orchestration and delivering results that are similar to optimal latency. In the future, we intend to use ANN to expand the VNF placement strategy, taking into account various QoS levels for different types of users (for example, premium subscribers have a lower latency violation limit than regular charged users) as well as other network dynamics.

REFERENCES

[1] A. Osseiran, F. Boccardi, V. Braun, K. Kusume, P. Marsch, M. Maternia, O. Queseth, M. Schellmann, H. Schotten, H. Taoka et al., "Scenarios for 5g mobile and wireless communications: the vision of the metis project", IEEE Communications Magazine, vol. 52, no. 5, pp. 26-35, 2014.

[2] C. Zhang, H. P. Joshi, G. F. Riley, and S. A. Wright, "Towards a virtual network function research agenda: A systematic literature review of vnf design considerations," Journal of Network and Computer Applications, vol. 146, p. 102417, 2019.

[3] M. Chiosi et al., Network functions virtualisation: An introduction benefits enablers challenges call for action, 2012.

[4] H. Moens and F. De Turck, "VNF-P: A model for efficient placement of virtualized network functions," in CNSM. IEEE, 2014, pp. 418–423.

[5] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in CNSM, 2015 11th International Conference on. IEEE, 2015, pp. 50–56.

[6] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella, "Opennf: Enabling innovation in network function control," in ACM SIGCOMM Computer Communication Review, vol. 44, no. 4. ACM, 2014, pp. 163–174.

[7] M. Emu, P. Yan and S. Choudhury, "Latency Aware VNF Deployment at Edge Devices for IoT Services: An Artificial Neural Network Based Approach", in 2020 IEEE International Conference on Communications Workshops (ICC Workshops), 2020-06, p.1-6.

[8] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, latency-optimal vnf placement at the network edge," IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, pp. 693–701, 2018.

[9] Sahhaf, S.; Tavernier, W.; Rost, M.; Schmid, S.; Colle, D.; Pickavet, M.; Demeester, P. Network Service Chaining with Optimized Network Function Embedding Supporting Service Decompositions. Elsevier Comput. Netw. 2015, 93, 492–505.

[10] W. Lu, L. Liang, and Z. Zhu, "On vnf-sc deployment and task scheduling for bulk-data transfers in inter-dc eons," in 2017 IEEE/CIC International Conference on Communications in China (ICCC), Oct 2017, pp. 1–4.

[11] Antevski, K.; Martín-Pérez, J.; Molner, N.; Chiasserini, C.F.; Malandrino, F.; Frangoudis, P.; Ksentini, A.; Li, X.; SalvatLozano, J.; Martínez, R.; et al. Resource Orchestration of 5G transport networks for vertical industries. In Proceedings of the 2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Montreal, QC, Canada, 9–12 September 2018; arXiv:1807.10430.

[12] Benkacem, I.; Taleb, T.; Bagaa, M.; Flinck, H. Optimal VNFs placement in CDN Slicing over Multi-Cloud Environment. IEEE J. Sel. Areas Commun. 2018, 36, 616–627.

[13] Chiaraviglio, L.; D'Andreagiovanni, F.; Siderotti, G.; Melazzi, N.B.; Salsano, S. Optimal Design of 5G Superfluid Networks: Problem Formulation and Solutions. In Proceedings of the IFIP/IEEE International Conference on Innovation in Clouds, Internet and Networks (ICIN), Paris, France, 20–22 February 2018; pp. 1–8.

[14] Pham, C.; Tran, N.H.; Ren, S.; Saad, W.; Hong, C.S. Traffic-aware and Energy-efficient VNF Placement for Service Chaining: Joint Sampling and Matching Approach. IEEE Trans. Serv. Comput. 2018, 99, 1.

[15] Wen, T.; Yu, H.; Sun, G.; Liu, L. Network Function Consolidation in Service Function Chaining Orchestration. In Proceedings of the IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, 23–27 May 2016; pp. 1–6.

[16] Leivadeas, A.; Falkner, M.; Lambadaris, I.; Kesidis, G. Resource Management and Orchestration for a Dynamic Service Chain Steering Model. In Proceedings of the IEEE Conference on Global Communications (GLOBECOM), Washington, DC, USA, 4–8 December 2016; pp. 1–6.

[17] Feng, H.; Llorca, J.; Tulino, A.M.; Molisch, A. Dynamic Network Service Optimization in Distributed Cloud Networks. In Proceedings of the IEEE International Conference on Computer Communications (INFOCOM), Workshops on Software-Driven Flexible and Agile Networking, San Francisco, CA, USA, 10–15 April 2016; pp. 300–305

[18] Savi, M.; Tornatore, M.; Verticale, G. Impact of Processing Costs on Service Chain Placement in Network Functions Virtualization. In Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN), San Francisco, CA, USA, 18–21 November 2015; pp. 191–197.

[19] Lombardo, A.; Manzalini, A.; Riccobene, V.; Schembra, G. An analytical tool for performance evaluation of software defined networking services. In Proceedings of the IEEE Network Operations and Management Symposium (NOMS), Krakow, Poland, 5–9 May 2014; pp. 1–7.

[20] Luizelli, M.C.; Cordeiro, W.L.C.; Buriol, L.S.; Gaspary, L.P. A fix-and-optimize approach for efficient and large scale virtual network function placement and chaining. Elsevier J. Comput. Commun. 2017, 102, 67–77.

[21] Addis, B.; Belabed, D.; Buet, M.; Secci, S. Virtual Network Functions Placement and Routing Optimization. In Proceedings of the IEEE International Conference on Cloud Networking (CloudNet), Niagara Falls, ON, Canada, 5–7 October 2015; pp. 171–177.

[22] Otokura, M.; Leibnitz, K.; Koizumi, Y.; Kominami, D.; Shimokawa, T.; Murata, M. Application of Evolutionary Mechanism to Dynamic Virtual Network Function Placement. In Proceedings of the IEEE International Conference on Network Protocols (ICNP), Workshops on Control, Operation and Application in SDN Protocols (CoolSDN), Singapore, 9–11 November 2016; pp. 1–6, doi:10.1109/ICNP.2016.7784475.

[23] Leontiou, N.; Dechouniotis, D.; Denazis, S.; Papavassiliou, S. A hierarchical control framework of load balancing and resource allocation of cloud computing services. Elsevier J. Comput. Electr. Eng. 2018, 67, 235–251.

[24] Hegyi, A.; Flinck, H.; Ketyko, I.; Kuure, P.; Nemes, C.; Pinter, L. Application Orchestration in Mobile Edge Cloud: Placing of IoT Applications to the Edge. In Proceedings of the IEEE International Workshops on Foundations and Applications of Self* Systems, Augsburg, Germany, 12–16 September 2016; pp. 230–235.

[25] Leivadeas, A.; Falkner, M.; Lambadaris, I.; Kesidis, G. Dynamic Virtualized Network Function Allocation in a Multi-Cloud Environment. In Proceedings of the IEEE International Conference on Telecommunications (ICT), Thessaloniki, Greece, 16–18 May 2016; pp. 1–5.

[26] Nam, Y.; Song, S.; Chung, J.M. Clustered NFV Service Chaining Optimization in Mobile Edge Clouds. IEEE Commun. Lett. 2017, 21, 350–353.

[27] Ma, W.; Beltran, J.; Pan, Z.; Pan, D.; Pissinou, N. SDN-Based Traffic Aware Placement of NFV Middleboxes. IEEE Trans. Netw. Serv. Manag. 2017, 14, 528–542.

[28] Zanzi, L.; Giust, F.; Sciancalepore, V. M2EC: A Multi-tenant Resource Orchestration in Multi-Access Edge Computing Systems. In Proceedings of the IEEE International Conference on Wireless Communications and Networking Conference (WCNC), Barcelona, Spain, 15–18 April 2018; pp. 1–6.

[29] Alleg, A.; Ahmed, T.; Mosbah, M.; Riggio, R.; Boutaba, R. Delay-aware VNF placement and chaining based on a flexible resource allocation approach. In Proceedings of the IEEE International Conference on Network and Service Management (CNSM), Tokyo, Japan, 26–30 November 2017; pp. 1–7.

[30] Wang, J.; Qi, H.; Li, K.; Zhou, X. PRSFC-IoT: A Performance and Resource Aware Orchestration System of Service Function Chaining for Internet of Things. IEEE Internet Things J. 2018, 5, 1400–1410.

[31] Cao, J.; Zhang, Y.; An, W.; Chen, X.; Sun, J.; Han, Y. VNF-FG Design and VNF Placement for 5G Mobile Networks. Springer J. Inf. Sci. 2017, 60, 1–15.

[32] Jemaa, F.B.; Pujolle, G.; Pariente, M. QoS-aware VNF placement Optimization in Edge-Central Carrier Cloud Architecture. In Proceedings of the IEEE International Conference on Global Communications Conference (GLOBECOM), Washington, DC, USA, 4–8 December 2016; pp. 1–7.

[33] Carpio, F.; Dhahri, S.; Jukan, A. VNF Placement with Replication for Load balancing in NFV networks. In Proceedings of the IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; pp. 1–6.

[34] Munoz, R.; Vilalta, R.; Yoshikane, N.; Casellas, R.; Martinez, R.; Tsuritani, T.; Morita, I. Integration of IoT, Transport SDN and Edge/Cloud computing for Dynamic Distribution of IoT Analytics and Efficient Use of Network Resources. IEEE J. Lightwave Technol. 2018, 36, 1420–1428.

[35] Lingen, F.V.; Yannuzzi, M.; Jain, A.; Irons-Mclean, R.; Lluch, O.; Carrera, D.; Perez, J.L.; Gutierrez, A.; Montero, D.; Marti, J.; et al. The Unavoidable Convergence of NFV, 5G, and Fog: A Model-Driven Approach to Bridge Cloud and Edge. IEEE Commun. Mag. 2017, 55, 28–35.

[36] Yannuzzi, M.; Lingen, F.V.; Jain, A.; Lluch, O.; Flore, M.M.; Carrera, D.; Perez, J.L.; Montero, D.; Chacin, P.; Corsaro, A.; et al. A New Era for Cities with Fog Computing. IEEE Internet Comput. 2017, 21, 54–67.

[37] Jia, M.; Liang, W.; Xu, Z.; Huang, M. Cloudlet Load Balancing in Wireless Metropolitan Area Networks. In Proceedings of the IEEE International Conference on Communications (INFOCOM), San Francisco, CA, USA, 10–14 April 2016; pp. 1–9.

[38] M. R. Garey. and D. S. Johnson, Computers and Intractability; A Guide to the Theory of NP-Completeness. New York, NY, USA: W. H. Freeman Co., 1990.

[39] R. Livni, S. Shalev, and O. Shamir, "On the computational efficiency of training neural networks," in Advances in Neural Information Processing Systems 27. Curran Associates, Inc., 2014, pp. 855–863.

[40] K. S. Ghai, S. Choudhury, and A. Yassine, "A stable matching based algorithm to minimize the end-to-end latency of edge nfv," Procedia Computer Science, vol. 151, pp. 377 – 384, 2019.

[41] K. S. Ghai, S. Choudhury, and A. Yassine, "Efficient algorithms to minimize the end-to-end latency of edge network function virtualization," Journal of Ambient Intelligence and Humanized Computing, 01 2020.

[42] Emu, M., Yan, P., Choudhury, S. (2020). Latency Aware VNF Deployment at Edge Devices for IoT Services: An Artificial Neural Network Based Approach. 2020 IEEE International Conference on Communications Workshops (ICC Workshops), 1–6. https://doi.org/10.1109/ICCWorkshops49005.2020.9145242

[43] M. Abu-Lebdeh, D. Naboulsi, R. Glitho, and C. W. Tchouati, "On the placement of vnf managers in large-scale and distributed nfv systems," IEEE Transactions on Network and Service Management, vol. 14, no. 4, pp. 875–889, Dec 2017.