Energy-aware Service Function Chaining Embedding in NFV Networks

Lin, Rongping; He, Liu; Luo, Shan; Zukerman, Moshe

Published in:
IEEE Transactions on Services Computing

Published: 01/03/2023

Document Version:
Post-print, also known as Accepted Author Manuscript, Peer-reviewed or Author Final version

Publication record in CityU Scholars:
Go to record

Published version (DOI):
10.1109/TSC.2022.3162328

Publication details:

Citing this paper
Please note that where the full-text provided on CityU Scholars is the Post-print version (also known as Accepted Author Manuscript, Peer-reviewed or Author Final version), it may differ from the Final Published version. When citing, ensure that you check and use the publisher's definitive version for pagination and other details.

General rights
Copyright for the publications made accessible via the CityU Scholars portal is retained by the author(s) and/or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights. Users may not further distribute the material or use it for any profit-making activity or commercial gain.

Publisher permission
Permission for previously published items are in accordance with publisher's copyright policies sourced from the SHERPA RoMEO database. Links to full text versions (either Published or Post-print) are only available if corresponding publishers allow open access.

Take down policy
Contact lbscholars@cityu.edu.hk if you believe that this document breaches copyright and provide us with details. We will remove access to the work immediately and investigate your claim.
Energy-aware Service Function Chaining Embedding in NFV Networks

Rongping Lin, Liu He, Shan Luo, and Moshe Zukerman, Life Fellow, IEEE

Abstract—Network function virtualization (NFV) is a new networking paradigm based on decoupling network functions from dedicated hardware, so these network functions can be run as pieces of software on general-purpose computation servers, which are called virtual network functions. In addition to guarantee the service qualities provided by NFV networks comparable to those provided by traditional telecommunication networks, energy consumption becomes one of the challenges faced by NFV. This is due to a large number of general computation servers that consume a significant amount of energy. We address here the problem of how to provide an energy-aware service function chaining (SFC) embedding in NFV networks with a hierarchical resource allocation, where an SFC has a set of virtual network functions to be executed in a specific sequential order providing a specific network service. Assuming a dynamic traffic scenario, we introduce a problem an integer linear programming (ILP) and three polynomial heuristic algorithms for resource allocation. All three heuristic algorithms achieve energy savings by shutting down idle devices and balance the tradeoff between energy cost and SFC request acceptance ratio. Numerical results demonstrate the quality of the proposed heuristic algorithms in terms of acceptance ratio by comparing them with the ILP method and a method extended from an exiting algorithm despite the fact that they save energy.

Index Terms—Network function virtualization, service function chaining embedding, energy-aware

I. INTRODUCTION

According to Cisco’s report [1], worldwide IP traffic will reach 396 Exabytes per month in 2022, and it is also expected that the number of connected devices will exceed three times the global world population by that year, at the same time, machine to machine (M2M) applications across many industries accelerate Internet of Things (IoT) growth, and M2M connections will grow to 14.6 billion, which is more than half of the total number of global connected devices and connections by that time. The traffic generated by the IoT devices poses challenges for networks because of the extensive number and vast requirement differences of IoT traffic, where traffic flows require a range of network functionalities for different services. Usually, to provide sufficient network service capabilities and flexible resource management, network service providers (NSPs) are required to make major investments in equipment and human resources.

Network function virtualization (NFV) [2], [3] is a new networking paradigm that is based on decoupling network functions from hardware so that virtual network functions (VNFs) can be run in software on general-purpose computation equipment, e.g., X86 servers. With this technology, the network can efficiently handle the traffic with its computation scale-up/down abilities and fine granularity connection management. Specifically, in an NFV network, a network service is provided by a set of sequential VNFs, called service function chaining (SFC), that process data flow in a given order to provide a network service [4]–[8]. For example, SFCs can be used for surveillance camera video streaming. In such a case, the VNFs provide firewall, proxy, and video transcoder in this order. Note that these VNFs are pieces of software that can be optimally embedded to general computation devices, and lead to significant cost savings by avoiding the usage of specific hardware, then the deployment of network services becomes flexible and efficient. The NFV was initiated by the service operators to accelerate the deployment of new services, enable the flexibility of network operation and management. Various NFV solutions have been provided by traditional network equipment providers, such as Ericsson, Huawei, Cisco, Nokia, Juniper Network, Alcatel-Lucent, and other companies, including Red Hat, Amdocs, VMware and Ciena [9], [10]. These solutions indicate a promising future of the NFV marketing and potential revenues for the service providers. European Telecommunications Standards Institute (ETSI) has an industry specification group on the standardization of NFV. The fourth release of the NFV related standards has been published as NFV Release 4 [11], and in this release optimizing networking to ease the connectivity for VNFs and network services is emphasized which provides an opportunity for the solutions introduced in this paper to improve the efficiency of SFC.

The energy consumption of information and communication technology (ICT) was approximately 10% of electricity generation worldwide, reaching 1,500 TWh of electricity in 2013, and was predicted to double by 2023 [12]. In addition, the annual datacenter energy cost was estimated to exceed the datacenter CAPEX in the near future according to current trend [13], and it is also reported that energy consumption constitutes between 20% to 40% of network OPEX [14]. Reducing energy consumption is especially important in NFV networks, because a large number of energy-hungry computation servers are used to form virtualized computation pool, and the cost of energy consumption highly affects the network revenue.
In NFV networks, the VNFs that process traffic flows are based on the computation hardware, such as high volume servers, or general X86 computation servers. Meanwhile, the computation capacity is usually designed to have the ability to process the traffic peak, and it is not necessary to turn on all the servers for the full computation capacity when the traffic load is low. In Fig. 1, we present an example to illustrate the energy saving of SFC embedding in an NFV network, where two SFC requests colored as red and green are deployed in the network and resources are allocated. Terminals connect to the network through access networks and in the core network, a network node contains a switch for networking and computation devices for VNF provision. The red SFC request includes a firewall, intrusion detection system (IDS), proxy, and network address translation (NAT) sequentially, which are embedded into nodes A, B, D, and F, respectively. The green SFC request includes a firewall, deep packet inspection (DPI), and NAT sequentially, which are embedded into nodes A, E, and F, respectively. It is noted that if the DPI of the green SFC is deployed in node B that also hosts DPI, the server at node E will not have any workload, in which case, the servers associated with node E can be shut down for power saving. Accordingly, reduction of energy consumption can be achieved by using energy-aware SFC embedding algorithms.

The problem considered here is a new problem. Its key novelty is the consideration of a hierarchical resource allocation in the context of NFV. Although previous NFV publications provided resource allocation for SFC embedding, they only considered the bottom layer resource allocation of physical nodes. On the other hand, it is also important for the practical usage of NFV technology to optimally share VNF resources instantiated by physical nodes. The existing publications do not consider constraints associated with other resources that are linked to the individual resources that they optimized, which makes these approaches non-practical. For example, optimizing computational resources available in physical nodes without considering the particular servers in the node that provide these computational resources is insufficient because the nodes may have many servers and it is important to optimize the usage of these individual servers. Similarly, there may be multiple VNF instances of the same VNF type and it is important to optimize the usage of these individual VNF instances. To consider the resource allocation at the particular VNF of the same type is necessary for the practical network management. The SFC embedding considers only resource allocation in physical nodes, which provides only upper bounds for the performance of the practical SFC embedding problem, that in many cases (e.g. in data centers) involves a significant level of resource abstraction and virtualization, and therefore, can not always be applied in network management directly. However, the problem considered in this paper involves a 4-layer hierarchical resource allocation that addresses network resources abstraction and virtualization, where physical nodes, servers at individual physical nodes, types of VNFs in individual servers, and specific VNFs are all taken into consideration together. Noticing that in certain practical situations (e.g. data centers), the computation resource for SFC embedding is allocated in a hierarchical way. In NFV networks, a network node usually has multiple general-purpose computation servers deployed, and the VNF instantiation and usage are similar to the situation considered in this paper. In particular, the SFC embedding in a data center, where a node contains tens of servers, and the resource allocation for the SFC embedding is also hierarchical. Such a 4-layer hierarchical resource allocation for SFC embedding is therefore practical and is likely to be applied by network operators. Meanwhile, integer linear programming (ILP) is applied to model this energy-aware SFC embedding problem, where we design variables, constraints, and the coupled relationships among them to generate this ILP formulation. This ILP formulation is the first mathematical model of the SFC embedding problem considering the hierarchical resource allocation, where physical nodes, servers, type of VNF, and specific VNF are taken into consideration together and make resource allocation simultaneously.

There may be communication overheads incurred by this 4-layer hierarchical resource allocation. However, the total resource demand for intra-communications is normally negligible compared to the total capacity available for such traffic. In a network node, multiple services are deployed and connected by an Ethernet switch providing high rate connections (e.g. 10G), where the LAN is used for the connections within the node. Meanwhile, there may be multiple VNFs in a server, and each of the VNFs is hosted by a virtual machine or a container, then a virtual switch with high rate switching (e.g. Open vSwitch can provide 10G switching rate) is applied for the LAN connections for VNFs. All these connections incur communication overheads, but these overheads are negligible given the high rate connections provided within a LAN. Accordingly, in this paper, for simplicity, the intra-communication overhead is not considered. In addition, the computing overhead increases because of the additional complication introduced by the 4-layer hierarchy when we optimize the resource allocation because the number of decision variables increases, and their optimal values must be computed simultaneously as compared to the existing single-layer resource allocation decisions. However, the computation of the resource allocation of SFC embedding is usually done at the network controller, which provides a sufficient computing capacity. Accordingly, in this paper, this computing overhead is also not considered for simplicity.

In contrast to the existing SFC embedding research publi-
cations that provide efficient embedding methods for a given set of SFC requests in a traditional network for a long holding time, in this paper, we consider terminals in the NFV network that generate SFC requests in a dynamic way, where requests arrive at the network randomly and leave the network when their service is completed, i.e., such SFC requests require a finite network service time. The main contributions of this paper are as follows.

1) The new energy-aware SFC embedding problem with a hierarchical resource allocation in the dynamic traffic scenario is investigated, and both the SFC embedding and the network operations (including NFV consolidation, NFV instantiation, and device operations) are optimized. In this hierarchical resource allocation, physical nodes, servers at individual physical nodes, types of VNFs in individual servers, and specific VNFs are taken into consideration together and resource allocations are decided simultaneously for the first time.

2) An ILP formulation for the above-mentioned problem is provided, where the objective is to minimize energy cost to accommodate a new SFC request arrival. In this formulation, the hierarchical resource allocations are simultaneously optimized which provides a new benchmark for the network performance.

3) Three polynomial heuristic algorithms are proposed to efficiently solve the problem, where the original complex problem is decomposed into multiple simple shortest path problems on an auxiliary graph that associates costs of VNF embedding, consolidation, and instantiation as link costs.

4) We provide extensive numerical results for performance evaluation of our proposed algorithms. They demonstrate that our heuristic algorithms give results that are close to those obtained by the ILP method for a small size test network. We also demonstrate the performance advantages of these heuristic algorithms over a method extended from an exiting algorithm for a realistic-sized network.

The remainder of this paper is organized as follows. In Section II, we provide background on existing work. In Section III, an ILP formulation for the SFC embedding problem is provided. Three algorithms for energy efficient SFC embedding are provided in Section IV. In Section V, numerical results are provided. Finally, the paper is concluded in Section VI.

II. RELATED WORK

We begin our coverage of the related work by discussing the virtual network embedding (VNE) problem because of its similarity to the SFC embedding problem considered in this paper. Both problems include virtual node embedding, i.e., mapping the virtual nodes (or VNFs) to physical nodes, and virtual link embedding, i.e., mapping the virtual links to physical paths [15], [16]. This similarity allows the reuse of certain ideas and techniques used for the VNE problem for the SFC embedding problem. In the VNE problem, a virtual network (VN) consists of virtual nodes and links, and the problem is to embed a VN into a physical network [17]–[24]. However, it is not straightforward to apply methodologies used for the VNE problem to the SFC embedding problem, because of the fundamental difference between these two problems. This difference stems from the fact that the virtual nodes in SFC embedding perform different network functions, like firewall, IDS, or NAT, and the virtual nodes in VNE represent only general computation resource requirements. In addition, virtual link embedding in SFC cascades all VNFs in an ordered sequence and virtual link embedding in VNE is performed according to the VN topology.

Most of the existing published work on NFV aims to provide end-users with high service quality levels, such as, reliability and delay [5], [25]–[29]. Several publications addressed the energy consumption problem in the context of NFV. Huin et al. [30] considered a static traffic scenario with multiple SFC requests given a priori, and proposed a column generation algorithm to minimize the energy consumption in SFC embedding. However, the on-off state of VNFs on servers are dynamically changing and this effect has not been considered in [30]. Therefore, the work of [30] cannot be directly applied to the problem considered in the present paper, where SFC requests arrive according to a dynamic process under which the on-off state of the VNFs is variable. Eramo et al. [31] investigated VNF migration policies in the SFC reconfiguration problem, and minimized the cost in terms of energy consumption cost and reconfiguration cost, where a Markov decision process based migration method was proposed. Bolla et al. [32] introduced a distributed approach based on an open source framework for the SFC embedding problem, which considered power managements in a proposed green abstraction layer. Mijumbi et al. [33] considered three NFV related applications, virtualized evolved packet core, virtualized radio access network, and virtualized customer premises equipment, for which they proposed energy saving approaches. Pham et al. [34] minimized the energy consumption in the problem of the virtual network function placement with the Markov approximation approach. Although the work in [31]–[34] considered energy saving in NFV networks, their objectives are different from those of this paper as they did not consider energy-aware resource allocations in the entire SFC embedding problem that includes VNF embedding and virtual link embedding.

There are publications that considered the optimal VNF and virtual link embedding in the SFC embedding problem. Kuo et al. [35] considered a static traffic scenario in the SFC embedding problem, where a set of SFC requests were given beforehand, and an iterative method was proposed to maximize the acceptance of demands based on the empty-loaded network that has no traffic been accommodated and with all VNF instances be ready for usage. However, this method can not be directly applied to our problem, where VNF instances are turned on/off dynamically according to the arrivals and departures of SFC requests. In [36], different types of VNFs are simply considered in terms of the different amounts of computation and storage resources, then an SFC embedding problem transforms into the problem of finding servers that have sufficient computation and storage resources that can satisfy the SFC request. This resource allocation did not consider types of VNFs in a server and multiple VNF
instances of the same VNF type, which is quite different from the 4-layer resource allocation in our problem. The resource allocation of SFC embedding in [37] only considers general computation resource allocation for VNFs at servers, and did not consider other layer resource allocations similarly. By comparison, in this paper we consider a hierarchical resource allocation, where physical nodes, servers at a physical node, types of VNFs in a server, and specific VNFs are all taken into consideration together to optimize resource allocation simultaneously. Accordingly, as mentioned in Section I, when existing algorithms focus their optimizations on specific individual resources, they do not consider constraints associated with other resources that are linked to the individual resources that they optimize. To the best of our knowledge, we present here the first work that investigates the SFC embedding problem where this more practical hierarchical resource allocation is considered. Among the existing publications on SFC embedding optimization [35]–[37] that are characterized by optimizing the resources required by a single request or a set of requests that arrive at once, we select the algorithm of [35] as a benchmark for comparison. The choice of the algorithm of [35] is made because the model assumption of [35] is closest to ours among [35]–[37]. In particular, the model of [35] considers various VNF types in resource provisioning as in our model, while the other two simplify the VNF resources as general computation/storage resources. The work of [35] partially solves our problem for dynamic traffic scenarios, so it will be extended in this paper to be applicable to our problem for the performance comparison purpose in Section V.

Some publications consider SFC embedding problems in IoT networks. Kouah et al. [38] considered an energy-aware SFC embedding problem in IoT networks, formulated the problem as a mixed integer linear program, and proposed a genetic algorithm to solve the problem. Chien et al. [39] investigated the SFC embedding problem in IoT networks, where the types and priorities of services required by terminal devices were considered, and a heuristic algorithm was proposed to reduce the data transmission time and balance the workload of VNFs. Wang et al. [40] considered an SFC orchestration in an IoT network, and provided a linear programming formulation as well as an approximate algorithm for resource optimization considering a sharp IoT traffic increase and avoiding resource idleness. However, all these publications considered static traffic scenarios where the SFC requests are given and there is no network operation decision in a long time perspective.

With the introduction of edge computing that complements cloud computing, there is recent work on SFC embedding in edge computing networks. Thanh et al. [41] investigated the SFC embedding problem for IoT applications in edge-cloud environments, where a proposed SFC embedding strategy under dynamic traffic considered resource efficiency, energy efficiency and network performance. Zheng et al. [42] investigated a hybrid SFC embedding problem to minimize latency in edge computing networks where forward and backward traffic may carry different content and different SFCs are traversed. Benefiting from the development of machine learning, machine learning methods are used in the SFC embedding problem to obtain a cost-effective solution. Fu et al. [43] proposed a deep reinforcement learning based scheme for the SFC embedding problem to handle the problem complexity and dynamic traffic scenarios. To apply a machine learning method, the Markov assumption was made for the SFC embedding. Liu et al. [44] provided two deep reinforcement learning based methods for the dynamic embedding problem under the assumption of the Markov decision process in small and large network sizes. There are other existing recent publications on SFC embedding in multi-domain networks. Liu et al. [45] provided a distributed algorithm to embed segments of an SFC in different domains for the multi-domain SFC embedding problem. This algorithm guarantees fairness among domains while balancing loads among domains and preserving the privacy and autonomy of individual ones. Toumi et al. [46] proposed a centralized method to embed SFC in multi-domain networks where the minimal information of individual domain infrastructure is disclosed. This method aims to optimize end-to-end latency, bandwidth, and deployment cost. Kibalya et al. [47] investigated the SFC embedding problem while considering a limited information disclosure of domains in multi-domain networks, and proposed a multi-stage graph based algorithm that involved a limited number of domains to reduce the execution time. Our previous work [48] on SFC embedding considered multi-domain networks in which individual domain information is not disclosed. Algorithms based on the column generation method were proposed to embed SFC requests in a dynamic traffic scenario.

In this paper, we study the SFC embedding problem with special consideration to energy efficiency, where the dynamic traffic scenario is considered. The objective is to minimize the power consumption of a realistic size network in a way that will not decrease the acceptance ratio of SFC requests significantly below that obtained under the ILP method. Specifically, the algorithm tries to not only successfully embed SFC requests but also shut down devices (including computation servers and network devices) to reduce energy consumption. The approach of shutting down idle devices or putting them in sleep mode to save energy has been reported in various publications [49]–[54] that can not apply directly to NFV. Then, extensions and modifications of the existing solutions considering resource usage constraints are required for them to be applied in NFV. For example, multiple devices are dependent of each other in resource usage if these devices embed the same SFC, then the shutting down of devices is highly related the details of SFC embedding.

III. ILP FORMULATION

The computation server may have a comparable idle power that may be more than 50% of the highest power usage, but the CPU utilization is zero [22], [55]. It is necessary to merge workloads to reduce the overhead from idle powers and increase resource utilisations.

In an NFV network modeled as a graph $G = (N^s, L^s)$, where $N^s$ is the set of physical nodes and $L^s$ is the set of physical links, a set of servers $T_k$ is attached to network node $k \in N^s$ and computational capacity $C^F_k$ is provided at server $t \in T_k$. Each server can instantiate different VNFs with different computational resource consumptions. Meanwhile, multiple VNFs
In the following, we provide and describe the constraints of reducing the power cost of each SFC embedding. The power cost of an SFC embedding is defined as the weighted power increment of physical links, which is denoted by $A_{kt}$, where $i$ is the network function, $k$ is the node and $t$ is the server node of the network. If this VNF is newly instantiated in the server, there may be multiple VNFs of the same type that are newly instantiated, and we use $L_{kt}$ to denote the set of VNFs of type $s$ instantiated in server $t$ at node $k$, which is set to \{1, 2, ..., $[C_k^t / \Delta_s]$\}, where $\Delta_s$ is the computational resources required to instantiate a VNF of type $s$, then the specific VNFs can be selected. If there are existing VNFs that can accommodate the network function, we use $D_{kt}$ to denote the set of existing VNFs of type $s$ in server $t$ at node $k$, and $U_{kt,d}$ to denote residual processing capability of the $d$th existing VNF of type $s$. The optimal embedding decision is the optimal combination considering these resource limitations. Accordingly, the problem with hierarchical resource allocations in the NFV network is a challenging problem.

To shut down the servers for energy saving and satisfy the embedding of the SFC requests, the instantiation of VNFs and the network function embedding in the server should be carefully designed. Meanwhile, a VNF in the server can be shared by multiple SFC requests if the computational capacity is satisfied. To arrange different SFC requests for VNF sharing can be a challenging problem for high resource utilizations. In this section, an ILP formulation is provided for resource allocation for a single SFC request arrival in a dynamic scenario, and embedding solution is derived by the ILP if applicable; otherwise, this SFC request is blocked because of lack of availability of network resources. The notations used in the formulation are defined in Table I.

The objective is to minimize the weighted power increment incurred by the SFC embedding, and the weight parameters are the electricity prices. It is formulated as follows.

\[
\sum_{k \in N^s} \epsilon_k \left\{ X_k (1 - \alpha_k) P_k + PL_k \sum_{m \in \text{adj}(k)} X_{mk} (1 - \gamma_{mk}) + \sum_{n \in \text{adj}(k)} X_{kn} (1 - \gamma_{kn}) \right\} + \sum_{k \in N^s} \sum_{t \in T_k} \epsilon_k \left\{ X_k^t (1 - \beta_k^t) P_k + \Delta P_k t P(i) A_{kt} \right\} + \sum_{m n \in L^s} \epsilon_{mn} X_{mn} (1 - \gamma_{mn}) PL K_{mn}.
\]

The formulation achieves energy savings in the network by reducing the power cost of each SFC embedding. The power increment comes from four aspects: power of switch, line card, server, power required by VNFs and power of physical links. In the following, we provide and describe the constraints of our ILP formulation.

\[
\sum_{m \in \text{adj}(k)} M_{mk}^{ij} - \sum_{n \in \text{adj}(k)} M_{kn}^{ij} = \sum_{t \in T_k} A_{kt}^j - \sum_{t \in T_k} A_{kt}^i \quad \forall i, j \in L^c, k \in N^s.
\]

The constraint (2) has four cases according to values of variables at the right-hand of the equation: 1) Both of $A_{kt}^i$ and $A_{kt}^j$ are equal to 1, then the right-hand side is equal to zero. If both network functions are mapped to the same server, the values of $A_{kt}^i$ and $A_{kt}^j$ are 1 simultaneously. Accordingly, the right-hand side of (2) equals zero, which ensures that for node $k$, the number of incoming physical links used is equal to the number of outgoing physical links used. While considering the objective function of minimizing the power consumption, none of physical links will be traversed by the virtual link $(i, j)$. 2) $A_{kt}^i = A_{kt}^j = 0$. Then the right-hand side is equal to zero. Neither $i$ nor $j$ is mapped to the server $t$ at node $k$, which makes the right-hand side of (2) equal to zero. The number of incoming used physical links of node $k$ equals the number of outgoing used physical links, which makes $k$ as an intermediate node; 3) The right-hand side is equal to 1 ($A_{kt}^i = 1$ and $A_{kt}^j = 0$). This implies that the network function $j$ is mapped to server $t$ at node $k$. Then, at node $k$, the number of used incoming physical links is larger than the number of used outgoing physical links by 1; 4) The right-hand side equals -1 ($A_{kt}^i = 0$ and $A_{kt}^j = 1$). In this case, the network function $i$ is mapped to server $t$ at node $k$. Accordingly, at node $k$, the number of used outgoing physical links is larger than that of used incoming physical links by 1.

\[
\sum_{k \in N^{f(i)} \cap T_k} A_{kt}^i = 1 \quad \forall i \in L^c.
\]

The constraint (3) ensures that each network function must be embedded in a server that can handle this network function.

\[
\sum_{i, j \in L^c} M_{mn}^{ij} \leq p_{mn} \quad \forall m n \in L^s.
\]

The constraint (4) ensures that the available bandwidth of physical link $(m, n)$ should be no less than the bandwidth consumption of this SFC request.

\[
P(i) A_{kt}^i \leq \sum_{d \in D_{kt}^{(i)}} t_{f(i)}^{f(i)} V_{kt,d}^i + \sum_{i \in L^{(i)}} C_{f(i)} H_{kt,l}^i \quad \forall i \in L^c, k \in N^s, t \in T_k.
\]

The constraint (5) ensures that if network function $i$ is mapped to server $t$ at node $k$, either existing VNF $f(i)$ or a newly instantiated VNF will be used. If an existing VNF is selected, there should be enough residual processing capacity, otherwise, a newly instantiated VNF will provide the full capacity. The constraint (6) ensures that at most one method:
TABLE I
SUMMARY OF NOTATIONS

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N^s )</td>
<td>Set of physical nodes, where a node contains a switch and servers</td>
</tr>
<tr>
<td>( L^s )</td>
<td>Set of physical links</td>
</tr>
<tr>
<td>( N^c )</td>
<td>Set of network functions</td>
</tr>
<tr>
<td>( L^c )</td>
<td>Set of virtual links between two adjacent functions</td>
</tr>
<tr>
<td>( G^c(N^c, L^c) )</td>
<td>Directed graph represents the physical network with nodes ( N^c ) and links ( L^c )</td>
</tr>
<tr>
<td>( oadj(k) )</td>
<td>Outgoing adjacent nodes of node ( k ), where there is a link from ( k ) to each node of ( oadj(k) )</td>
</tr>
<tr>
<td>( iadj(k) )</td>
<td>Incoming adjacent nodes of node ( k ), where there is a link from each node of ( iadj(k) ) to ( k )</td>
</tr>
<tr>
<td>( f(i) )</td>
<td>Type of network function ( i, i \in N^c )</td>
</tr>
<tr>
<td>( P(i) )</td>
<td>Computation requirement of network function ( i, i \in N^c )</td>
</tr>
<tr>
<td>( S )</td>
<td>All types of network functions in ( N^c ), ( S = { { f(i) \mid i \in N^c } } )</td>
</tr>
<tr>
<td>( C_s )</td>
<td>Processing capacity of a newly instantiated VNF ( s, s \in S )</td>
</tr>
<tr>
<td>( \Delta_s )</td>
<td>Computation needed to instantiate a VNF ( s, s \in S )</td>
</tr>
<tr>
<td>( T_k )</td>
<td>Set of servers at node ( k, k \in N^s )</td>
</tr>
<tr>
<td>( C_k )</td>
<td>Number of available computation of server ( k ) at node ( k, k \in N^s )</td>
</tr>
<tr>
<td>( b_{ij} )</td>
<td>Bandwidth requirement of virtual link ( (i, j), (i, j) \in L^c )</td>
</tr>
<tr>
<td>( p_{mn} )</td>
<td>Available bandwidth of physical link ( (m, n), (m, n) \in L^s )</td>
</tr>
<tr>
<td>( L_{kt}^i )</td>
<td>Set of VNF ( i ) can be instantiated at server ( t ) at node ( k ), is set to ( { 1, 2, ..., [C_k^i/\Delta_s] } ), ( s \in S, t \in T_k, k \in N^s )</td>
</tr>
<tr>
<td>( D_{kt}^i )</td>
<td>Set of existing VNF ( s ) in server ( t ) at node ( k, s \in S, t \in T_k, k \in N^s )</td>
</tr>
<tr>
<td>( U_{kt,d}^i )</td>
<td>Residual processing capability of the ( i )th existing VNF ( s ) in server ( t ) at node ( k, s \in S, d \in D_{kt}^i, t \in T_k, k \in N^s )</td>
</tr>
<tr>
<td>( e_k )</td>
<td>Electricity price at node ( k, k \in N^s )</td>
</tr>
<tr>
<td>( \epsilon_{mn} )</td>
<td>Electricity price at link ((m, n), (m, n) \in L^s )</td>
</tr>
<tr>
<td>( P_k )</td>
<td>Idle power of switch ( k ), ( k \in N^k )</td>
</tr>
<tr>
<td>( P_{Lk} )</td>
<td>Power of line card in switch ( k ), ( k \in N^k )</td>
</tr>
<tr>
<td>( PL_{km,n} )</td>
<td>Power of physical link ((m, n), (m, n) \in L^s )</td>
</tr>
<tr>
<td>( P_{id} )</td>
<td>Idle power of server ( t ) at node ( k, t \in T_k, k \in N^s )</td>
</tr>
<tr>
<td>( \Delta P_{kt} )</td>
<td>Power increment for an unit resource usage on server ( t ) at node ( k, t \in T_k, k \in N^s )</td>
</tr>
<tr>
<td>( N )</td>
<td>Set of nodes in which servers can handle function ( l, N^l \in N^s, l \in S )</td>
</tr>
<tr>
<td>( \omega_{ki} )</td>
<td>On-off state of switch at node ( k ), referred to as switch ( k ) afterward, ( k \in N^s )</td>
</tr>
<tr>
<td>( \beta_k )</td>
<td>On-off state of server ( t ) at node ( k, t \in T_k, k \in N^s )</td>
</tr>
<tr>
<td>( \gamma_{mn} )</td>
<td>On-off state of link ((m, n), (m, n) \in L^s )</td>
</tr>
<tr>
<td>( M_{mn}^{ij} )</td>
<td>Virtual link ((i, j)) traverses physical link ((m, n), (m, n) ) or not</td>
</tr>
<tr>
<td>( A_{kt}^i )</td>
<td>Function ( i ) is mapped to server ( t ) at node ( k, t \in T_k, k \in N^s )</td>
</tr>
<tr>
<td>( V_{kt,d}^i )</td>
<td>Function ( i ) is consolidated into the ( i )th existing VNF ( f(i) ) in server ( t ) at node ( k ) or not, ( i \in N^c, d \in D_{kt}^i, t \in T_k, k \in N^s )</td>
</tr>
<tr>
<td>( H_{kt,l}^i )</td>
<td>Function ( i ) is supported by the ( l )th newly instantiated VNF ( f(i) ) in server ( t ) at node ( k ) or not, ( i \in N^c, l \in L_{kt}^i, t \in T_k, k \in N^s )</td>
</tr>
<tr>
<td>( X_{kt,l} )</td>
<td>The ( l )th instantiation of VNF ( s ) in server ( t ) at node ( k ) is used by the SFC request or not, ( s \in S, l \in L_{kt}^i, t \in T_k, k \in N^s )</td>
</tr>
<tr>
<td>( X_k^s )</td>
<td>Switch ( k ) is used by the SFC or not, ( k \in N^s )</td>
</tr>
<tr>
<td>( X_k^t )</td>
<td>Server ( t ) at node ( k ) is used by the SFC or not, ( t \in T_k, k \in N^s )</td>
</tr>
<tr>
<td>( X_m,n )</td>
<td>Link ((m, n), (m, n) ) is used by the SFC or not, ((m, n) \in L^s )</td>
</tr>
</tbody>
</table>

existing VNF or newly instantiated VNF, will be selected. These two constraints not only indicate the embedding decision of the SFC request but also the operation of the network, especially the consolidation of VNF and the instantiation of VNF.

\[
\sum_{i \in N^c, f(i) = s} P(i)V_{kt,d}^i \leq U_{kt,d}^s \quad \forall s \in S, k \in N^s, t \in T_k, d \in D_{kt}^s. \quad (7)
\]

\[
\sum_{i \in N^c, f(i) = s} P(i)H_{kt,l}^i \leq C_s \quad \forall s \in S, k \in N^s, t \in T_k, l \in L_{kt}^s. \quad (8)
\]

In this problem, we consider that a VNF may be repeated more than one time in an SFC, and VNF embedding can be consolidated into the existing VNF instance under capacity limitations. The constraint (7) ensures that the required processing resource is no larger than the available processing capability. The constraint (8) ensures that the processing resource usage of newly instantiated VNF is constrained.

\[
\sum_{i \in N^c, f(i) = s} H_{kt,l}^i \leq |N^c|X_{kt,l}^s \quad \forall s \in S, k \in N^s, t \in T_k. \quad (9)
\]

\[
\sum_{s \in S, f(i) = s} X_{kt,l}^s \Delta_s \leq C_k^l \quad \forall k \in N^s, t \in T_k. \quad (10)
\]

The constraints (9) and (10) calculate the number of newly instantiated VNFs for this arriving traffic request. A new VNF instance may be shared by multiple VNFs of the SFC request, because we allowed a VNF may be repeated more than one time in an SFC, so the constraint(9) ensures the newly instantiated VNF can be properly shared. The constraint (10) ensures the computation resources used by instantiating VNFs should be no larger than the available computation resources at that server.

\[
\sum_{i \in L^s} A_{kt}^i \leq |N^s|X_k^t \quad \forall k \in N^s, t \in T_k. \quad (11)
\]
that requests are being accommodated based on the first come first served (FCFS) principle which is considered acceptable and reasonable in many real-world systems. In other words, we try our best to serve every incoming request first, and only then we aim to allocate resources such that power increment is minimized.

IV. HEURISTIC ALGORITHM

In the dynamic traffic scenario, we can apply the above ILP formulation to derive the embedding solution for the SFC request arrival. If the ILP problem is feasible and the optimal solution is obtained, the SFC request is accepted, otherwise, this SFC request is blocked. However, the SFC embedding is an NP-hard problem [16], and the computation time of the ILP problem may be too long and cannot scale to large size problems. In this section, efficient heuristic algorithms are proposed to approximately solve the problem for scalability, energy consumption and blocking ratio simultaneously, and the SFC embedding problem is divided into multiple small problems in the heuristic algorithm. In particular, every small problem comprises a virtual link embedding problem and two VNFs embedding problems at the two ends of the virtual link.

To embed a virtual link and its two VNFs at the two ends of the virtual link, an auxiliary graph is built for converting the embedding problem into a shortest path routing problem. The details of the auxiliary graph design are introduced in Algorithm 1. Fig. 2 provides an example of the auxiliary graph design, where Fig. 2(a) shows the original network topology, and Fig. 2(b) shows the auxiliary graph. The physical nodes \( N^s \) in the original graph are duplicated as \( N^s \) and \( N^{s'} \), so node \( A \) becomes two nodes \( A \) and \( A' \). Physical links are also duplicated where an original link from \( m \) to \( n \) becomes two new links from \( m \) to \( n' \) and from \( m' \) to \( n' \). In Fig. 2(b), the dashed lines connect the physical nodes and network function \( i \) or \( j \), where \( i \) and \( j \) are two VNFs of the virtual link, respectively, and the usage of a dashed line indicates VNF embedding into the node accordingly. The solid lines are the duplications of the physical links in the original network topology, and a path consisting of the solid line is the embedding of a virtual link that connects two VNFs. We also note that to consider the situation where two VNFs are embedded into the same node, in the auxiliary graph, dotted lines are added where a node connects to both \( i \) and \( j \), such as link \( (i, C) \) and \( (C, j) \) in Fig. 2(b). This type of links are added under conditions that the node (actually, the servers attached to) has enough capability to host both VNFs of \( i \) and \( j \). The shortest path algorithm can be applied to derive a route from \( i \) and \( j \) in the auxiliary graph. Then, the shortest path route contains both two VNF embeddings and the virtual link embedding information.

The link cost of the auxiliary graph is assigned according to the power consumption of the devices. Specifically, Equation (14) gives details of link costs: 1) the solid link is the physical links of the original network, and the link cost is set to the power cost of turning on the link and the switch at the end node; 2) the cost of the dashed link between physical nodes and \( i \) or \( j \) are set to the power cost values if the VNF is embedded in the node. It is noted that the dashed
Algorithm 1: Build auxiliary graph

Input: Current network $G^c(N^c,L^c)$, VNFs at $i$ and $j$, temporary embedding solution $T_{solution}$.
Output: Auxiliary graph $G^e$.
BEGIN:
1 Auxilary graph $G^e \leftarrow \emptyset$.
2 Add nodes $N^e \cup N^u \cup \{i,j\}$ into $G^e$.
  //Add solid lines
3 for $(m,n) \in L^s$ do
4 \hspace{1em} If $p_{mn} \geq b_{ij}$, add link $(m,n')$ and link $(m',n')$ into $G^e$.
end
  //Add dash lines
5 If VNF at $f(i)$ has been embedded into a node $t$ in $T_{solution}$ already, add link $(i,t)$, goto line 8.
6 for $n \in N^s$ do
7 \hspace{1em} Find a server in node $n$ with the minimal power cost to provide enough computation for VNF $f(i)$, add link $(i,n)$.
end
8 If VNF at $f(j)$ has been embedded into a node $t$ in $T_{solution}$ already, add link $(t,j)$, goto line 11.
9 for $n \in N^s$ do
10 \hspace{1em} Find a server in node $n$ with the minimal power cost to provide enough computation for VNF $f(j)$, add link $(n',j)$.
end
  //Add dotted lines
11 If both VNF $f(i)$ and VNF $f(j)$ have been embedded into a node $t$ in $T_{solution}$ already, add link $(t,j)$, goto line 14.
12 for $n \in N^s$ do
13 \hspace{1em} Find a server in node $n$ with the minimal power cost to provide enough computation for both VNF $f(i)$ and VNF $f(j)$, add link $(n,j)$.
end
14 Set the link cost $c_{mn}$ in $G^e$ according to (14).
END

Algorithm 2: A heuristic algorithm

Input: An arriving SFC request $G^r(N^r,L^r)$, and current network $G^c(N^c,L^c)$.
Output: An embedding solution $result$ in the current network.
BEGIN:
  //small problem sorting
1 for $i \in N^c$ do
2 \hspace{1em} for $s \in S$, $k \in N^s$, $t \in T_k$, $d \in D_{kt}^d$ do
3 \hspace{2em} If $U^s_{kt,d} \geq P(i)$, count$_i$ = count$_i$ + 1.
end
  //small problem solving
4 for $ij \in L^c$ do
5 \hspace{1em} weight$_{ij}$ = $P(i)/\text{count}_i + P(j)/\text{count}_j$
6 \hspace{1em} $L$.push($ij$, weight$_{ij}$).
end
7 Sort $L$ in the descending order of weight$_{ij}$.
  //small problems solving
8 for $ij \in L^c$ do
9 \hspace{1em} Build an auxiliary graph $G$ by Algorithm 1.
10 \hspace{1em} Find the shortest path $P$ from $i$ to $j$ on $G$.
11 \hspace{1em} If $P$ exists, $result \leftarrow result \cup P$. Otherwise, link $ij$ cannot be embedded, return false.
end
12 Update network resources according to $result$.
END

$t^* = \arg \min_{t \in T_m}((1 - \beta_m)P_m + \Delta P_{mt}P(i)) + (1 - \alpha_m)P_m$

Value 0 of the indication function implies that the same server can host both $i$ and $j$, then the idle power of the server is counted only once.

$$c_{mn} = \begin{cases} (1 - \gamma_{mn})(\epsilon_n PL_m + \epsilon_m PL_n + \epsilon_{mn} PLK_m) \\
+ \epsilon_n(1 - \alpha_m)P_n, \quad m \in N^s, n \in N^u \end{cases}$$

$$\min_{t \in T_m} \epsilon_m((1 - \beta_m)P_m + \Delta P_{mt}P(i)) + \epsilon_n(1 - \alpha_m)P_n$$

$$m = i, n \in N^s$$

$$\min_{t \in T_m} \epsilon_m((1 - \beta_m)P_m + \Delta P_{mt}P(j)),$$

$$m = i, n \in N^s$$

$$m \neq i, n \in N^s$$

$$\min_{t \in T_m} \epsilon_m((1 - \beta_m)P_m + \Delta P_{mt}P(j)),$$

$$m \in N^s, n = j.$$ (14)

As discussed before, the SFC embedding problem is NP-hard, and it is also found that the ILP formulation provided by this paper still has the complexity issue where there may be a large number of variables and constraints to denote network states. With the auxiliary graph and the link cost setting, the embedding decision based on network states has been largely simplified, which also can be seemed as the reduction of network states. Specifically, in (14), the computation server with the minimal power cost is selected as the potential server in a physical node for embedding and set costs to the cor-
responding links, this significantly simplifies the embedding decision and reduces network states. For convenience, the information of the minimal power cost computation server for each type of VNF embedding has been stored at physical nodes, which reduces the minimal searching complexity at the cost of small storage spaces.

The details of the proposed heuristic algorithm are provided in Algorithm 2. This heuristic algorithm contains the executions of Algorithm 1 for auxiliary graph building. In the proposed algorithm, an SFC request is embedded part-by-part in terms of small problems as described above. In this paper, three different embedding sequences of small problems are investigated for algorithm performance: 1) descending order of \( P(i) + P(j) \), sorted by descending order of the computation requirements of \( i \) and \( j \). Usually, VNFs with larger computation requirements may lead to a higher power, and this ordering may reduce the energy consumption as these VNFs has priority to be embedded in the low energy consumption places; 2) ascending order of \( \text{count}_i + \text{count}_j \), sorted by ascending order of the number of feasible embedding places where VNFs of \( i \) and \( j \) can be hosted, respectively. This ordering prioritizes the VNFs that has limited embedding choices to improve the acceptance ratio; 3) descending order of \( \frac{P(i)}{\text{count}_i} + \frac{P(j)}{\text{count}_j} \), sorted by descending order of the ratios of values in 1) and 2), which contains both ideas of 1) and 2). This ordering prioritizes VNFs with fewer embedding choices and more computation resource requirements. For the sake of brevity, in Algorithm 2, only the third sorting method described above is used. According to the three different sorting methods described above, three different heuristic algorithms are provided, and we name them as Heuristic-power, Heuristic-resource, and Heuristic-power-resource, for sorting method 1), 2) and 3), respectively. Then, the Algorithm 2 shown in the paper is the exact details of the Heuristic-power-resource.

In Algorithm 2, From lines 1 to 3, the number of feasible embedding places for each VNF without using new VNFs is calculated, that is, how many existing VNFs that can handle the VNF embedding. From lines 4 to 7, small problems are stored in set \( L \) associated with weight, and ordered accordingly. For other two sorting methods described, we can change weight value to be \((p(i) + p(j))\) for the first sorting method, and \((-\text{count}_i - \text{count}_j)\) for the second sorting method, respectively.

In line 8 of Algorithm 2, starting from the beginning of \( L \), each small problem is processed. After the auxiliary graph is built in line 9, the shortest path algorithm is applied to derive the path in line 10, which gives the minimal power consumption. If any one of the small problems is failed, this SFC request is blocked. After all small problems obtain embedding solutions, the network resources are updated according to the embedding solutions result, and then the SFC is successfully embedded.

Algorithm Complexity and Feasibility

The entire embedding procedure contains \(|L^c|\) executions of Algorithm 1 and one execution of Algorithm 2. The complexity contributed by \(|L^c|\) executions of Algorithm 1 is \(O((|L^c||L^c|+|N^c||N^c|))\), where the first term comes from solid line adding, and the second term comes from dash lines and dotted lines adding (the minimal power cost of device for embedding each type of VNF has been stored in each physical node, then the searching of the minimal item is avoided). The complexity of Algorithm 2 is \(O(|L^c|\log |L^c| + |L^c||N^c|^2)\), where the first term comes from sorting operations, and the second term is from the \(|L^c|\) executions of the shortest path algorithm (Dijkstra algorithm) on the auxiliary graph \( G^c \). Therefore, the complexity in total of the heuristic algorithms is \(O((|L^s||L^c| + |N^s||N^c| + |L^c|\log |L^c| + |L^c||N^c|^2)\).

As discussed, this heuristic algorithm may be implemented with one of the three different sorting procedures, so it becomes three algorithms (Heuristic-power, Heuristic-resource, and Heuristic-power-resource), and the complexity we just derived applies to all three algorithms. Accordingly, they are all polynomial algorithms and are applicable to large scale problems. Since the original SFC embedding problem is NP-hard, we convert the original problem into an approximate problem that can be solved by the algorithm with polynomial complexity. This algorithm can be used in large size problems. The quality of the solutions of the heuristic algorithms is measured by the so-called approximation errors, which is the performance difference between the heuristic algorithms and the ILP of the original problem that provides the optimal embedding solution for an SFC request in the next section.

After the execution of the Algorithm 2, the solution obtained may be non-optimal for the SFC request embedding because of the approximation problem solving. However, the solution is feasible because at each step of the algorithm (including multiple executions of Algorithm 1), the feasibility of the solution has been guaranteed by the feasible embedding selection for each VNF.

When network state changes, e.g., because some SFC requests leave the network, other existing deployed SFC may become non-optimally embedded, then re-embedding these SFC may improve the network performance. This re-embedding problem requires that the selection of re-embedded SFCs, new embedding solutions, the cost of service disruptions, and other constraints are taken into consideration and it is beyond the scope of this paper. In this paper, for simplicity, we only consider the energy-aware SFC embedding problem, and we do not consider re-embedding together with our embedding problem.

V. NUMERICAL RESULTS

In this section, we compare the performance of the ILP problem solving and three proposed algorithms. In the simulation, the SFC request arrivals are assumed to follow a Poisson process with arrival rate \( \lambda \) and the mean of holding time of a request is a negative exponential distribution with a mean \( 1/\mu \) for this common used case, then the network load is calculated as \( \lambda/\mu \) (Erlangs). We also assume there are hosts attached to network nodes, where hosts are user terminals that generate SFC requests for the network. All these hosts are independent and the merged SFC requests generated by hosts follow a Poisson process. For a given SFC request,
the number of VNFs follows a uniform distribution, and bandwidth and computation requirements of an SFC request are also uniformly distributed. We repeat an experiment 11 times at a specific network load, and use the results to show the error bars of 95% confidence intervals based on Student’s t-distribution. Table II gives the parameter settings used in the six-node network and the Usnet network (shown in Fig. 3).

From Figs. 4 to 6, we present results of our three proposed heuristic algorithms (Heuristic-power, Heuristic-resource, and Heuristic-power&resource) and the ILP method. These three heuristic algorithms are similar except that they use different sorting methods that decide the embedding order of the virtual links of the SFC request. The ILP method represents the solution of the ILP problem provided in the paper, which obtains the optimal solution for an SFC embedding request. Then a small performance difference to the ILP method indicates efficiencies of the proposed heuristic algorithms.

Fig. 4 presents performance comparisons that have been based on the network depicted in Fig. 3(a). In Fig. 4(a), we compare the acceptance ratio values. We observe that the acceptance ratio obtained using the ILP formulation to derive the SFC embedding is the highest. Among the heuristic algorithms (which have very close acceptance ratios), the Heuristic-resource algorithm has the highest acceptance ratio, followed by the Heuristic-power&resource algorithm and the lowest acceptance ratio is obtained by the Heuristic-power. In Fig. 4(a), when network load increases, the acceptance ratios of the four methods decrease, this is because more SFC requests compete for the limited network resources, and it leads to more SFC requests being blocked for the resource shortage. In Fig. 4(b), the electricity costs are compared, and the order is the same as that in Fig. 4(a), where ILP has the highest electricity cost, followed by the Heuristic-resource algorithm, the Heuristic-power&resource algorithm and the Heuristic-power algorithm. In Fig. 4(b), when network load increases, the electricity costs of the four methods decrease, this is because higher network loads mean denser SFC request arrivals, and in our experiments, the number of SFC requests is fixed, then a shorter network running time leads to a less energy consumption. The energy consumption is affected by time durations, then we focus on performance differences between methods at fixed network loads. In Figs. 4(a) and 4(b), the heuristic-resource algorithm obtains an acceptance ratio lower than the ILP by about 2.2%, and electricity cost lower than the ILP by about 2.4%. The heuristic-power algorithm obtains an acceptance ratio lower than the ILP by about 3.1%, and electricity cost lower than the ILP by about 3.5%. Note that the three heuristic algorithms have both the similar electricity cost and the similar acceptance ratio. The three algorithms have three different subproblem sorting strategies, which are the three different balances between electricity cost and acceptance ratio.

The numbers of turned on links and servers (called on-link and on-server afterward) that sampling during the experiments

<table>
<thead>
<tr>
<th>TABLE II</th>
<th>PARAMETER SETTINGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>Six-node</td>
</tr>
<tr>
<td>( \lambda ) (arrival rate)</td>
<td>0 - 9.0</td>
</tr>
<tr>
<td>( 1/\mu ) (mean of service time)</td>
<td>1</td>
</tr>
<tr>
<td>Number of servers at a node</td>
<td>unif(1,4)</td>
</tr>
<tr>
<td>Number of hosts attached to a node</td>
<td>unif(20,50)</td>
</tr>
<tr>
<td>Computation of a server</td>
<td>40</td>
</tr>
<tr>
<td>Link bandwidth</td>
<td>30</td>
</tr>
<tr>
<td>Idle power of server</td>
<td>200 W</td>
</tr>
<tr>
<td>Idle power of switch</td>
<td>100 W</td>
</tr>
<tr>
<td>Power of line card</td>
<td>10 W</td>
</tr>
<tr>
<td>Power of physical link</td>
<td>20 W</td>
</tr>
<tr>
<td>Electricity price at node</td>
<td>25 cents</td>
</tr>
<tr>
<td>Electricity price at link</td>
<td>25 cents</td>
</tr>
<tr>
<td>Types of VNFs</td>
<td>5</td>
</tr>
<tr>
<td>Computation to instantiate VNFs</td>
<td>10,12,14,16,18</td>
</tr>
<tr>
<td>Number of VNFs in an SFC</td>
<td>unif(2,4)</td>
</tr>
<tr>
<td>Computation requirement</td>
<td>unif(1.10)</td>
</tr>
<tr>
<td>Bandwidth requirement</td>
<td>unif(1.3)</td>
</tr>
<tr>
<td>Number of requests</td>
<td>10000</td>
</tr>
<tr>
<td>Number of experiments</td>
<td>11</td>
</tr>
</tbody>
</table>
of the four algorithms are also compared in Figs. 4(c) and 4(d). The ILP has the highest on-link number, and the other three heuristic algorithms have the similar on-link number in Fig. 4(c). This is because a link is usually traversed by multiple paths of different node pairs, and if a link is turned off, the associated paths are disrupted which prevents finding the best SFC embedding solution. Meanwhile, the power of a link is far lower than that of a server in practice. Based on the high computation complexity, the ILP method can derive better embedding solutions with the cost of more physical links. In Fig. 4(d), the ILP has the highest on-server number and each of the other three heuristic algorithms has a similar on-server number that the ILP does. This is because the ILP achieves the highest acceptance ratio which implies that more SFC requests are accommodated, then more servers are needed for computational resources. The reason that all the four algorithms have similar on-server numbers is that the servers are the main contributor to energy consumption, and the ILP and our three algorithms focus on the reduction of the number of active servers which lead to similar on-server numbers. Besides on-link and on-server comparisons, the on-link utilizations and on-server utilizations are also compared in Figs. 4(e) and 4(f). According to the results, the ILP has both the highest on-link and the highest on-server utilization, and the three heuristic algorithms have similar values. Accordingly, we can observe that a high resource sharing leads to a high utilization, and fewer devices are needed to be turned on for a given traffic load, so energy saving is achieved. In Figs. 4(c) to 4(f), when network load increases, the plots of four methods go up, this is because when network load increases, more SFC requests compete for a limited network resource, then more links/servers will be turned on and the utilization will be increased accordingly.

To illustrate how the algorithms take advantage of variable electricity prices, we provide more simulation results for the six-node network, where the electricity price is set to various electricity prices with \( \sigma = 5 \), respectively. Figs. 5(a) and 5(b), acceptance ratios and electricity costs achieved by the algorithms are provided under the various electricity prices with \( \sigma = 5 \) and \( \sigma = 10 \) cents, respectively. Figs. 5(c) and 5(d) show acceptance ratios and electricity costs under the various electricity price with \( \sigma = 10 \), respectively. Comparing Figs. 4(a), 5(a), and 5(c), we observe minimal changes in the acceptance ratios, the trends (as traffic increases) and in the order of the acceptance ratio values for the different algorithms. This is because the algorithms' first aim is to accommodate each SFC request without considering costs (or prices) at all. Only then, they try to reduce electricity costs. However, we observe a significant cost benefit that is achieved by the algorithms as the price variability increase from comparing Figs. 4(b), 5(b) and 5(d). We also observe from this comparison that there is no significant difference in trend (as traffic increases) and in the order of electricity costs of algorithms. The explanation for the significant cost benefit achieved by our algorithms with an increase of price variability is that with increased price variability in the network, our algorithms have the opportunity to embed SFCs into physical nodes and links that have lower electricity prices to minimize the objective function value, which reduces electricity cost accordingly.

To further illustrate the ability of the algorithms to achieve cost reduction in scenarios involving variability of electricity price, the average electricity costs of different algorithms under three standard deviation values (the value 0 represents the case where the price is fixed to 25 cents) are listed in Table III. From the table, we can again see that higher price variabilities lead to lower electricity costs while the same acceptance ratio is maintained.

<table>
<thead>
<tr>
<th>Standard deviation</th>
<th>0</th>
<th>5</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILP</td>
<td>82.22</td>
<td>76.92</td>
<td>68.69</td>
</tr>
<tr>
<td>Heuristic-resource</td>
<td>80.28</td>
<td>75.38</td>
<td>67.17</td>
</tr>
<tr>
<td>Heuristic-power&amp;resource</td>
<td>79.75</td>
<td>75.08</td>
<td>66.98</td>
</tr>
<tr>
<td>Heuristic-power</td>
<td>79.31</td>
<td>74.91</td>
<td>66.72</td>
</tr>
</tbody>
</table>

We also demonstrate the performance of the three heuristic algorithms as compared to the [35]-based benchmark for the USnet network as shown in Fig. 3(b). However, certain extensions and modifications are made on the relevant algorithm of [35] to make it applicable to our problem in this paper. Specifically, to make it applicable to the dynamic traffic scenario, this algorithm must be executed for a single arriving SFC request at a time, because it performs static resource allocation for a single request. Accordingly, the SFC embedding of [35] can not be optimized with consideration of all SFC requests together. Also note that the algorithm of
that when the network load increases, more SFC requests are considered in the model of [35], and for comparison, we can assume that what we consider to be the empty-loaded VNFs are switched off to save energy in the model of [35]. It is also noted that the ILP method cannot be applied in this scenario because the ILP problem solving is intractable for large networks due to its excessive running time.

In Fig. 6(a), the acceptance ratios of the four algorithms (our three heuristic algorithms and the [35]-based benchmark) decrease as the network load increases. This is explained by the fact that higher network load leads to more blocking as discussed in Fig. 4(a). As observed in Fig. 6(a), the acceptance ratio of the benchmark is lower than that of each of our three algorithms. In Fig. 6(b), the electricity costs of the four algorithms are compared, and our three algorithms achieve lower costs than the benchmark when network load is from 14 erlangs to 22 erlangs, but the benchmark achieves the lowest cost in the range of 24–30 erlangs. The reason for this is that when the network load increases, more SFC requests are blocked and there is no need to turn on links and servers, then no additional energy is consumed. It is noted that this is not an efficient way as there is high request blocking, and the unused resources imply low resource utilization. Fig. 6(c) to Fig. 6(f) show the average on-links, average on-servers, link utilization and server utilizations of the four algorithms. From the results presented in these figures, we observe that the benchmark considers more links and servers turned on than each of our three algorithms does, but the utilization of the turned on devices is the lowest among the four algorithms. Meanwhile, the performance-ranking orders of acceptance ratios, electricity costs, average on-links, average on-servers, link utilization, and server utilization of our three algorithms observed by the results presented in Figs. 6(a)–6(f) are the same as those obtained in the case of the six-node network presented in Figs. 4(a)–4(f), respectively. These comparisons show that our algorithms minimize the energy consumption while maintaining high acceptance ratios, and for each one of our three algorithms, we observe the relevant tradeoff balance between acceptance ratio and electricity cost.

VI. CONCLUSION

We have considered the energy-aware SFC embedding problem with the integration of multiple hierarchical resource allocations in NFV networks in a dynamic traffic scenario where SFC request arrivals follow a Poisson process. Both SFC embedding and network operations, including VNF consolidation, instantiation and device operations, are jointly considered in the problem. An ILP formulation with the objective of minimizing the power cost for an arriving SFC request has been provided. Three similar heuristic algorithms based on network state reduction have been also provided for scalability, and three algorithms achieved three different tradeoff balances between acceptance ratio and electricity cost in the SFC embedding at dynamic traffic scenarios. The numerical results demonstrate that the heuristic algorithms have achieved similar performance as the ILP method (where the acceptance ratio is lower only by 2.2% to 3.1%, and the electricity cost is lower by 3.5%) for a six-node network, and also demonstrate their performance advantages over a method extended from an existing algorithm for the USNet network. In certain applications, there may be multiple SFC requests arriving at the same time, then future work may focus on energy-aware SFC embedding with a batch of SFC request arrivals. Another potential future work is to use meta-heuristic methods, such as genetic algorithm, to solve this optimization problem, and this method may obtain a better solution than our heuristic algorithms with a potentially longer computation time.

REFERENCES


Rongping Lin received the Ph.D. degree from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, 2013. He is currently an Associate Professor with the School of Information and Communication Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu, P.R. China. His research interests include network optimization, networking protocol design, and network function virtualization.

Liu He is pursuing the M.S. degree with the School of Information and Communication Engineering, UESTC, P.R. China. Her research interest includes optimization and network function virtualization.

Shan Luo received the Ph.D. degree in information engineering from Nanyang Technological University in 2014. She is currently an Associate Professor with the School of Aeronautics and Astronautics, UESTC, P.R. China. Her research interests include wireless communications and optimization.

Moshe Zukerman (M’87-SM’91-F’07-LF’20) received the B.Sc. degree in industrial engineering and management, the M.Sc. degree in operations research from the Technion-Israel Institute of Technology, Haifa, Israel, and the Ph.D. degree in engineering from University of California, Los Angeles, in 1985. He was an independent consultant with the IRI Corporation and a Postdoctoral Fellow with the University of California, Los Angeles, in 1985-1986. In 1986-1997, he was with Telstra Research Laboratories (TRL), first as a Research Engineer and, in 1988-1997, as a Project Leader. He also taught and supervised graduate students at Monash University in 1990-2001. During 1997-2008, he was with The University of Melbourne, Victoria, Australia. In 2008 he joined City University of Hong Kong as a Chair Professor of Information Engineering, and a team leader. He has also served as CityU’s Acting CIO since December 2020. He has over 300 publications in scientific journals and conference proceedings. He has served on various editorial boards such as Computer Networks, IEEE Communications Magazine, IEEE Journal of Selected Areas in Communications, IEEE/ACM Transactions on Networking and Computer Communications.