Received: 12 February 2018 | Revised: 19 December 2018 | Accepted: 19 December 2018

DOI: 10.1002/dac.3912

RESEARCH ARTICLE

WILEY

Energy efficient virtual network embedding for federated software-defined networks

Mohamed Haji Dahir | Hadi Alizadeh | Didem Gözüpek

Department of Computer Engineering, Gebze Technical University, Kocaeli, Turkey

Correspondence
Didem Gözüpek, Department of Computer Engineering, Gebze Technical University, Kocaeli, Turkey.
Email: didem.gozupek@gtu.edu.tr

Funding information
The Scientific and Technological Research Council of Turkey (TUBITAK), Grant/Award Number: 114E245

Summary

In this paper, we focus on energy efficient virtual network embedding in federated (multidomain) software-defined networks (SDNs). We first formulate an optimization problem as an integer linear program (ILP) that minimizes the energy consumption of the network links, while at the same time adhering to the bandwidth and CPU requirements of the virtual network requests. We then propose a polynomial-time heuristic algorithm, which consists of three stages. In the first stage, the top SDN controller decides on whether to partition the virtual network request into smaller subrequests and give subrequests to multiple domains or give the enitre virtual network request to a single domain, while in the second stage, each SDN controller implements virtual network embedding in its own domain. Finally, in the third stage, the algorithm performs interdomain routing if partitioning decision had been made in the first stage. Our simulation results demonstrate that our proposed algorithm yields close performance to the solutions obtained by using the optimization software CPLEX that implements our ILP.

KEYWORDS

energy efficiency, green networks, heuristic algorithms, integer linear programming, multidomain networks, software-defined networks, virtual network embedding

1 | INTRODUCTION

Due to the enormous growth of Internet and continuous increase in Internet applications and services, deployment of new services is becoming increasingly more difficult, leading to the ossification of the Internet. To this end, the concept of "network virtualization" has been introduced. Network virtualization decouples the tasks of infrastructure providers (InPs) and service providers (SPs) by giving the responsibility of managing the physical and virtual networks (VNs) to InPs and SPs, respectively. The major goal of network virtualization is to overcome the current Internet ossification problem[1-4] by enabling the integration of new features as well as the cohabitation of multiple heterogeneous network architectures on a shared physical infrastructure.

A VN is mapped on top of one InP or is hosted among multiple InPs. Nevertheless, each VN can only be managed by one SP. Each SP creates and sustains its own VNs. Each VN is composed of a set of virtual nodes that are connected via virtual links. A fundamental problem in VNs is the VN embedding (VNE) problem, which deals with the mapping of physical resources to VN components. VNE can be separated into two mapping phases: node mapping, where virtual nodes have to be mapped to physical nodes, and link mapping, where virtual links connecting these virtual nodes have to be mapped to paths connecting the corresponding nodes in the substrate network (SN). VNE can be optimized according

to several parameters such as embedding cost, link bandwidth, and energy efficiency. Every virtual node is mapped to only one substrate node, and each virtual link is mapped to a substrate path consisting of one or more substrate links. [5]

Numerous variants of the VNE problem take the CPU processing power of the substrate nodes and the bandwidth requirements of the virtual links into consideration.[6-9] However, few works focus on making this assignment in an energy efficient way. Indeed, demand for energy is continuously increasing and research studies show that the energy consumption of information and communications technologies (ICTs) is also continuously increasing. This increase has both environmental and financial costs. Therefore, in order to redesign communication networks in an energy efficient way and to emphasize environmental friendliness, the concept of "green networks" has been introduced.[7-10] These requirements render energy efficiency in VNE a vital issue.

Software-defined networking (SDN) is a novel networking paradigm in which the forwarding hardware is decoupled from control decisions. In SDN, network intelligence is logically centralized in a software-based controller (the control plane), and network devices become simple packet forwarding devices (the data plane).[11] Some studies[2,12-15] focus on multidomain SDN, also known as federated SDN, where there exist multiple domains, a controller is placed in each of the domains, and a centralized global SDN controller orchestrates local controllers to gather and distribute resource state information. Consequently, in the data plane, the switches are partitioned into multiple domains, and the controller in the control plane configures the switches in its domain via OpenFlow protocol. The nature of the networking environment in federated SDN entails semi-centralized algorithms; in particular, a semi-centralized algorithm for VNE. To the best of our knowledge, this paper is the first one in the literature that proposes a semi-centralized heuristic algorithm for VNE in federated SDN.

Studies on green networks show that the power consumption of the network equipment is insensitive to the traffic load.[16] In fact, the energy consumption depends on whether the network element is active or not since the network elements consume energy even in idle state. Accordingly, in this paper, we first formulate an energy efficient VNE problem by minimizing the number of active network links and taking the CPU and bandwidth requirements of the VN requests (VNRs) into account. Motivated by the fact that federated SDN environment requires semi-centralized algorithms, we then propose a polynomial-time heuristic algorithm for federated SDN. The major merit of this work is our proposed algorithm since it is the first VNE heuristic for federated SDN. While other energy efficient VNE heuristics in the literature are mostly centralized, semi-centralized nature of our algorithm is its major distinctive feature.

On the other hand, works that have hitherto been conducted in the green networking domain have focused on the case where the entire network has a single owner and the goal is to reduce/minimize the total energy consumption. Networks in which different parts have different owners are called "multidomain networks" and most networks on the Internet have this property. In a federated SDN architecture, each domain can be managed by a different SDN controller. There are few works in the literature that focus on green multidomain networks; in fact, to the best of our knowledge, the only works in the literature that focuses on green multidomain networks are.[17,18] Therefore, our paper contributes also to the green multidomain networks literature.

The rest of the paper is organized as follows. In Section 2, we discuss a summary of related work. In Section 3, we present an integer linear program (ILP) formulation of our problem. We discuss our proposed heuristic algorithm in Section 4 and analyze its complexity in Section 5. We then present the simulation results in Section 6. Finally, Section 7 concludes the paper.

2 | RELATED WORK

Many works in the literature[2,5,6,19,20] deal with variants of VNE problem, which is known to be NP-hard. These variants have numerous constraints and objectives such as CPU, disk, bandwidth, and memory requirements of the substrate and virtual links, maximum length requirement for the virtual paths, the requirement on the maximum number of virtual nodes or links that can be assigned to a certain substrate node or link, and economical benefits.

The survey paper in Fischer et al[6] classifies the literature about VNE problem according to numerous parameters such as centralized/distributed and static/dynamic. In a centralized setting, one entity is responsible for making the embedding decisions. This entity has global knowledge of the network. Although this approach is advantageous for obtaining high solution quality, it has scalability problem in large networks since a single entity may be overwhelmed with a high number of requests. For instance, the works in other studies[21-25] follow a centralized approach for VNE. In comparison, a distributed approach entails multiple entities for the computation of the embedding solution. Since each entity deals with only a portion of the solution simultaneously, this approach has better performance in terms of scalability. However,

due to the lack of global knowledge, it may lead to embedding decisions with poor quality. For instance, the works in other studies[26-29] follow a distributed approach for VNE. On the other hand, a semi-centralized approach addresses this trade-off between quality of the embeddings and scalability by partitioning the solution stages among multiple entities, where each entity executes a centralized algorithm for its own computation, and coordination is achieved among these multiple entities to obtain the final solution. In a federated SDN scenario, each SDN controller performs the computations for its own domain while at the same time coordinating the final decision together with the other SDN controllers. Hence, the nature of the networking environment in federated SDN settings require semi-centralized algorithms.

Some works[7-10] in the literature focused on energy efficiency in VNE. For instance, Su et al[9] and Botero et al[7] formulated an ILP, which minimizes the number of active links and nodes. Chen et al[8] proposed an energy efficient VNE algorithm, which minimizes energy consumption. Guan et al[10] proposed an ant colony optimization-based energy efficient VNE to minimize energy consumption in multiple data centers for both computing and network resources. However, the algorithms in Chen et al[8] and Guan et al[10] are centralized algorithms, whereas the algorithm we propose in this paper has a semi-centralized nature and is applicable to federated SDN.

Some studies on virtualization in multidomain networks focused on virtual topology design in multidomain optical networks.[12,13] For instance, the work in Hong et al[12] proposed a survivable virtual topology mechanism for multidomain optical networks with the aim of minimizing total link cost of the virtual traffic requests. They designed heuristic algorithms for embedding virtual links into multidomain optical links using partition and contraction mechanisms and suggested a hierarchical SDN-based control plane to share information between domains. However, unlike our work, these studies do not focus on energy efficiency.

Some recent studies[2,14,15,30-34] have focused on the usage of SDN for VNE. In particular, the authors in Zhou et al[31] proposed a multidomain VN mapping mechanism for SDN using distributed architecture to achieve better efficiency and flexibility rather than the traditional policy-based VNE across multiple domains. However, the author solves subproblems of resource mapping with two major algorithms: first, the domain level VN mapping algorithm, and then, the intradomain VN mapping algorithm. The work in Chowdhury et al[2] introduced VNE algorithms interaction between the node mapping and the link mapping phases by proposing two VNE algorithms: deterministic VNE and randomized VNE. In another work, Gao et al[14] suggested various methods that find a pair of disjoint end-to-end paths that may cross multiple domains from source to destination and result in a minimum total cost of a pair of end-to-end paths. These strategies include methods for interdomain information exchange that hold the costs of disjoint paths within a domain. Unlike this paper, their approach ignores the bandwidth and CPU constraints in routing as well as energy efficiency. In Zaman et al,[32] authors present an SDN-based QoS-aware framework for metropolitan optical networks, while in Tegueu et al,[33] a dynamic proactive resource defragmentation scheme based on path migration is proposed in order to address the fragmentation problem inherent to online virtual links mapping in virtualized SDN-enabled infrastructures. Besides, an online VNE technique for scenarios where SDNs are used as the SN for hosting non-SDN VNs is presented in Osgouei et al.[34] Although the work in Osgouei et al[34] takes the bandwidth constraints into account, it does not focus on a scenario with multiple SDN controllers. Moreover, it does not take energy efficiency into account. In short, unlike this paper, other works in the literature that are concerned with multidomain VNE ignore energy efficiency.

The work in Zhong et al[15] proposed a splitting approach for the case where VN is to be deployed across multiple domains with the objective of minimizing embedding cost. The author considers the VN splitting problem and resolves the limitation of the existing methods by formulating a linear program with the objective of minimizing embedding cost. Thereby, the author proposes a max-flow/min-cut theory-based VN splitting problem together with a system clustering method to reduce the problem complexity, transforms the two-domain splitting problem into a max-flow/min-cut problem, and solves it efficiently by the shortest augmenting path algorithm. The major shortcoming of the work in Zhong et al[15] is that it ignores bandwidth and CPU requirements of the VN. To put it in a nutshell, unlike this paper, works in the literature that are concerned with multidomain VN splitting do not take energy efficiency into account.

Green multidomain networks is also a relatively unexplored area. To the best of our knowledge, the only works in the literature about green multidomain networks are Hou et al.[17,18] In Hou et al,[17] the authors focus on grooming in multidomain green optical networks. They propose a heuristic algorithm that establishes lightpaths by maximizing power efficiency and taking into account peak traffic distribution in multidomain networks. Authors in Hou et al[18] focus on a similar scenario. They present a grooming approach that involves both interdomain and intradomain provisioning of connection demands while at the same time yielding higher power efficiency and port savings. Unlike our work, the papers in Hou et al[17,18] do not focus on VNE.

In this work, we first formulate an optimization problem as an ILP, which minimizes the energy consumption of the network links, while adhering to the CPU and bandwidth requirement of the VNR. To address this problem, we then propose a semi-centralized heuristic algorithm for energy efficient VNE in federated SDN. In a nutshell, the contributions of this paper are as follows:

- To the best of our knowledge, our proposed algorithm is the first VNE heuristic in the literature for federated SDN.
- To the best of our knowledge, this paper is the first work on VNE in green multidomain networks.

3 | PROBLEM FORMULATION

In this section, we first describe the network model and then provide an ILP formulation of our problem. SN refers to the physical infrastructure, where nodes have processing power (CPU) capacity and links have bandwidth capacity. VN is the logical network that consists of virtual nodes that are connected via virtual links. Each VN has specific demands that have to be met, in particular, CPU power and bandwidth constraints. As in other studies,[7-10] energy conservation is obtained by putting links into sleep mode, ie, by minimizing the number of active links. The objective of our VNE problem is to map the VNRs to the SN resources such that the CPU and bandwidth constraints of the virtual nodes and links, respectively, are satisfied and the total energy consumption is minimized.

Table 1 shows the input parameters for our ILP, while Table 2 describes the decision variables. An SN can be described as a graph $G^s = (V^s, E^s)$, where V^s is the set of substrate nodes and E^s is the set of substrate links. In addition, a VNR l can be described as a graph $G^{vl} = (V^{vl}, E^{vl})$, where V^{vl} is set of virtual nodes and E^{vl} is the set of virtual links. As in Chowdhury et al,[2] we construct an augmented graph $G^a = (V^a, E^a)$, where V^a is set of augmented nodes and E^a is the set of augmented links, by combining both SN and VNs as follows (see Figure 1):

$$V^a = V^s \cup \bigcup_{l \in [L]} V^{vl}$$

$$E^a = E^s \cup \bigcup_{l \in [L]} \{(a, a') | a \in V^s, a' \in V^{vl}\}\text{ , where [L] = \{1,2,. … ,L\}.}$$

TABLE 1 Input parameters

Input parameter	Explanation
$G^s = (V^s, E^s)$	Substrate network topology.
$G^{vl} = (V^{vl}, E^{vl})$	Virtual network request l .
$G^a = (V^a, E^a)$	Augmented graph topology.
w_{ij}	Power consumption of substrate link (i, j).
p_i	CPU capacity of substrate node i.
C_{ij}	Bandwidth capacity of substrate link (i, j).
D^{ml}	CPU capacity required by virtual node m of VNR l.
B^{mn}	Bandwidth required by virtual link (m, n).
(s^{ml}, t^{nl})	Virtual link (m, n) of VNR l corresponds to a pair of vertices s^{ml} (source vertex) and t^{nl} in the augmented graph
L	Total number of VNRs.

Abbreviation: VNR, virtual network request.

TABLE 2 Decision variables

Decision variable	Explanation
x_i^{ml}	$= \begin{cases} 1, & \text{if virtual node } m \text{ of VNR } l \text{ is embedded on substrate node } i. \\ 0, & \text{otherwise} \end{cases}$
y_{ij}^{mnl}	$= \begin{cases} 1, & \text{if virtual link } (m, n) \text{ of VNR } l \text{ is assigned to a substrate path that uses physical link } (i, j). \\ 0, & \text{otherwise} \end{cases}$
f_{ij}^{mnl}	The amount of flow of virtual link (m,n) of VNR l passing through substrate link (i,j).
x_{ij}	$= \begin{cases} 1, & \text{if the substrate link (i,j) is active.} \\ 0, & \text{otherwise} \end{cases}$

Abbreviation: VNR, virtual network request.

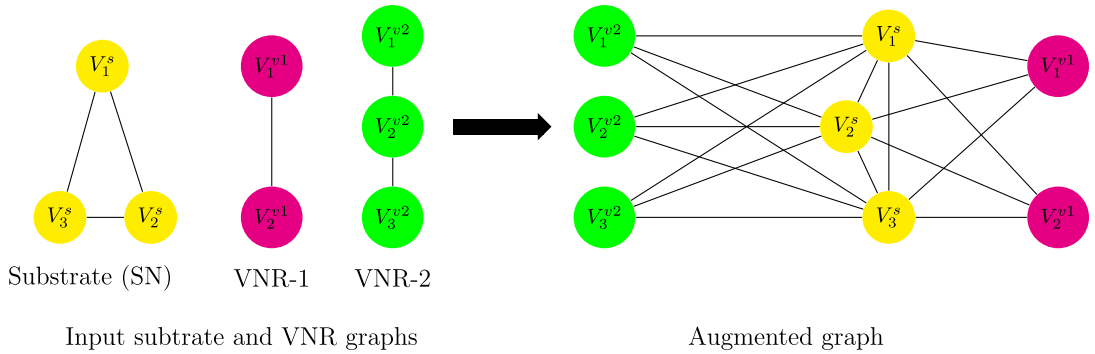

FIGURE 1 Example for an augmented graph. VNR, virtual network request

Objective function:

The objective function minimizes the total power consumption of the SN by considering the power consumption of active links (when $x_{ij} = 1$):

$$\min \quad \left(\sum_{i=1}^{|V^s|} \sum_{j=i}^{|V^s|} w_{ij} \times x_{ij} \right). \tag{1}$$

Constraints:

Constraints (2) and (3) are link power consumption constraints, where M is a very large integer. Constraint (2) guarantees that if a nonzero flow pass through substrate link (i,j), then $x_{ij} = 1$. Constraint (3) ensures that substrate link (i,j) is inactive ($x_{ij} = 0$) if no flow passes through it:

$$M \times x_{ij} \geq f_{ij}^{mnl}; \quad \forall i,j \in V^s, \forall l \in [L], \forall m,n \in V^{vl}, \tag{2}$$

$$x_{ij} \leq \sum_{l=1}^{L} \sum_{m=1}^{|V^{vl}|} \sum_{n=1}^{|V^{vl}|} f_{ij}^{mnl}; \quad \forall i,j \in V^s. \tag{3}$$

Constraint (4) ensures that the total CPU demand of the virtual nodes embedded to the same substrate node cannot exceed the CPU capacity p_i of substrate node i:

$$\sum_{l=1}^{L} \sum_{m=1}^{|V^{vl}|} D^{ml} \times x_i^{ml} \leq p_i; \quad \forall i \in V^s. \tag{4}$$

Constraint (5) guarantees that each substrate node can host at most one virtual node from the same VNR:

$$\sum_{m=1}^{|V^{vl}|} x_i^{ml} \leq 1; \quad \forall i \in V^s, \forall l \in [L]. \tag{5}$$

Constraint (6) ensures that every virtual node is embedded to exactly one substrate node:

$$\sum_{i=1}^{|V^s|} x_i^{ml} = 1; \quad \forall l \in [L], \forall m \in V^{vl}. \tag{6}$$

Flow conservation constraints:

In flow conservation constraints, a path is found for each source and destination pair (s^{ml}, t^{nl}), which are nodes in the augmented graph corresponding to virtual nodes m and n of VNR l. Constraint (7) forces the net flow of intermediate nodes to be zero, while constraint (8) ensures that each flow originates from its source node in the augmented graph with the required bandwidth B^{mn} assigned to it. Constraint (9), on the other hand, ensures that each flow terminates at its destination node in the augmented graph:

$$\sum_{j=1}^{|V^s|} f_{ij}^{mnl} - \sum_{j=1}^{|V^s|} f_{ji}^{mnl} = 0; \quad \forall i \in V^s, \forall l \in [L], \forall m,n \in V^{vl}, \tag{7}$$

$$\sum_{j=1}^{|V^s|} f_{s^{ml}j}^{mnl} - \sum_{j=1}^{|V^s|} f_{js^{ml}}^{mnl} = B^{mn}; \quad \forall l \in [L], \forall m, n \in V^{vl}, \tag{8}$$

$$\sum_{j=1}^{|V^s|} f_{t^{nl}j}^{mnl} - \sum_{j=1}^{|V^s|} f_{jt^{nl}}^{mnl} = -B^{mn}; \quad \forall l \in [L], \forall m, n \in V^{vl}. \tag{9}$$

Constraint (10) ensures that total amount of flow on substrate link (i, j) must not exceed its bandwidth capacity C_{ij}:

$$\sum_{l=1}^{L} \sum_{m=1}^{|V^{vl}|} \sum_{n=1}^{|V^{vl}|} f_{ij}^{mnl} \leq C_{ij}; \quad \forall i, j \in V^s. \tag{10}$$

Recall that s^{ml} and t^{nl} represent the source and terminal vertices in the augmented graph corresponding to the virtual link (m, n) in VNR l. For a given substrate node i, constraints (11) and (12) model the relationship between $y_{s^{ml}i}^{mnl}$ and x_i^{ml} as well as the one between $y_{it^{nl}}^{mnl}$ and x_i^{nl}. Corresponding to a particular virtual link, the virtual node that is assigned to a substrate node can be a source or a terminal vertex. If there is a nonzero flow from/to the nodes corresponding to the virtual nodes in the augmented graph, then constraints (11) and (12) ensure that the pertinent virtual node is assigned to that substrate node. Constraint (11) guarantees that if for some substrate node i, there exists a nonzero flow from a source node s^{ml} in the augmented graph ($y_{s^{ml}i}^{mnl} = 1$), then the virtual node m, which is represented by s^{ml} in the augmented graph, is assigned to substrate node i. Similarly, constraint (12) guarantees that if there exists a nonzero flow from a terminal node t^{nl} in the augmented graph (virtual node n in VNR l) to a substrate node i, then the virtual node n is assigned to substrate node i. Note that a virtual node can be a source vertex for some incident virtual link and a terminal vertex for some other incident virtual link.

$$x_i^{ml} = y_{s^{ml}i}^{mnl}; \quad \forall i \in V^s, \forall s^{ml} \in V^a, \forall l \in [L], \forall m, n \in V^{vl} \tag{11}$$

$$x_i^{nl} = y_{it^{nl}}^{mnl}; \quad \forall i \in V^s, \forall t^{nl} \in V^a, \forall l \in [L], \forall m, n \in V^{vl} \tag{12}$$

Constraints (13), (14), and (15) avoid flow splitting, ie, they ensure that the flow corresponding to a virtual link passes through a single path in the SN. Constraints (13), (14), and (15) avoid path splitting for the source node, terminal node, and intermediate nodes, respectively:

$$\sum_{j=1}^{|V^s|} y_{ij}^{mnl} = 1; \quad \forall l \in [L], \forall m, n \in V^{vl}, i = s^{ml}, \tag{13}$$

$$\sum_{j=1}^{|V^s|} y_{ji}^{mnl} = 1; \quad \forall l \in [L], \forall m, n \in V^{vl}, i = t^{nl}, \tag{14}$$

$$\sum_{j=1}^{|V^s|} y_{ij}^{mnl} = \sum_{j=1}^{|V^s|} y_{ji}^{mnl} \leq 1; \quad \forall i, j \in V^s \setminus \{s^{ml}, t^{nl}\}, \forall l \in [L], \forall m, n \in V^{vl}. \tag{15}$$

Constraints (16) and (17) model the relationship between $y_{s^{ml}j}^{mnl}$ and $f_{s^{ml}j}^{mnl}$. In particular, constraint (16) forces $y_{s^{ml}j}^{mnl}$ to be zero when the corresponding flow variable $f_{s^{ml}j}^{mnl}$ is zero, and constraint (17) forces $y_{s^{ml}j}^{mnl}$ to be 1 when $f_{s^{ml}j}^{mnl}$ is nonzero:

$$y_{s^{ml}j}^{mnl} \leq f_{s^{ml}j}^{mnl}; \quad \forall j \in V^s, \forall s^{ml} \in V^a, \forall l \in [L], \forall m, n \in V^{vl}, \tag{16}$$

$$f_{s^{ml}j}^{mnl} \leq M \times y_{s^{ml}j}^{mnl}; \quad \forall j \in V^s, \forall s^{ml} \in V^a, \forall l \in [L], \forall m, n \in V^{vl}. \tag{17}$$

Constraints on decision variables:

$$x_i^{ml}, x_{ij}, y_{ij}^{mnl} \in \{0,1\} \quad \forall i,j \in V^s, \forall l \in [L], \forall m,n \in V^{vl}, \tag{18}$$

$$f_{ij}^{mnl} \in Z^+ \cup \{0\} \quad \forall i,j \in V^s, \forall l \in [L], \forall m,n \in V^{vl}. \tag{19}$$

4 | PROPOSED HEURISTIC ALGORITHM

In a large network, network management becomes infeasible when there is a single SDN controller. To this end, federated SDN, where each SDN controller is responsible for the network management in its own domain is require. Figure 2 shows an example architecture of the federated SDN we focus on with four domains. Each SDN controller interacts with the top SDN controller through a northbound interface (NBI) located on the top SDN controller. NBI is used to generate actions in the controller such as computation of the routing paths or the (un)provisioning of services. On the other hand, the communication with the network equipment is established via southbound interfaces (SBI) from each SDN controller. The federated architecture demonstrates how a set of resources can be divided through different SDN controllers, while internally all SDN controllers are coordinated by the hierarchically top SDN controller. Top SDN controller is aware of the nodes in each SDN controller and the bandwidth level of the links interconnecting the SDN controllers (interdomain links).

The nature of the networking environment in federated SDN necessitates semi-centralized heuristic algorithms. In a large network, having a centralized algorithm with a single SDN yields not only a scalability problem in terms of computational overhead but also the problem of single point of failure. In this section, we propose a polynomial-time semi-centralized heuristic algorithm for the problem we have formulated in (1) to (19). We design our algorithm for a federated SDN architecture with an arbitrary number of SDN domains. The general outline of our proposed algorithm is as follows. In our federated model, the top SDN performs VN assignment, ie, it decides on whether to split a VNR into partitions and send each partition to an SDN domain or to send the entire VNR to one of the domains. Since splitting a VNR is also an NP-hard problem,[1] we have developed a polynomial-time heuristic algorithm which performs both VN assignment and VNE. While VN assignment is performed by the top SDN, VNR embedding is performed by the controllers in SDN domains. Besides, virtual links can be mapped to interdomain links or intradomain links. In the top SDN controller, the first step is to determine whether to make VN partitioning or not. If both ends of a virtual link are assigned to the same SDN controller, then this SDN controller handles the path assignment for this virtual link. Otherwise, there exist

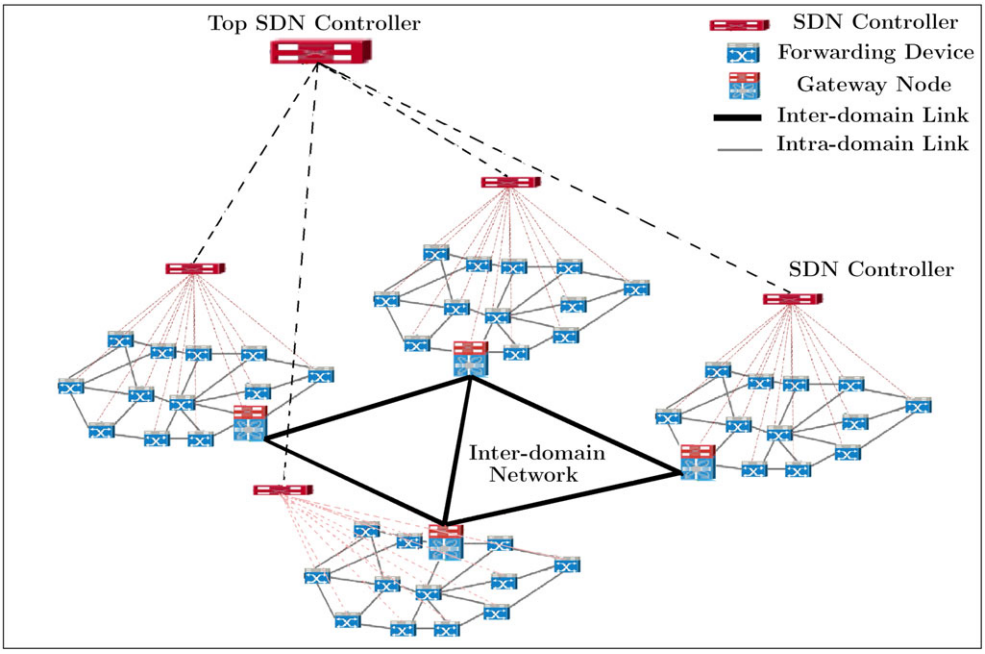

FIGURE 2 The architecture of the considered federated SDN with four SDN domains. SDN, software-defined network

some virtual links whose endpoints are assigned to different SDN domains. Our algorithm replaces each of these virtual links by three links and makes the path assignment accordingly (will be explained in detail).

4.1 | VN assignment

We refer to our heuristic algorithm as *VNE in Federated SDN* (VNEFSDN). Table 3 shows the input parameters used by VNEFSDN. VNEFSDN consists of three phases: *VN assignment*, *VNE*, and *interdomain routing*. Phase 1 shows the pseudocode of VN assignment performed by the top SDN, phase 2 shows the pseudocode of VNE executed by the domain controller, whereas phase 3 shows the pseudocode of performing interdomain routing.

VN assignment procedure, which is performed by the top SDN, takes VNR G_v, Γ_s^i, C_t^i, C^i, W^i, G_{id}, C^{id}, W^{id}, and k as input. The top SDN requests the Γ_s^i and C_t^i from SDN domains. In line 2, the assignment procedure sorts the vertices of the VNR in descending order according to the value of b_v and stores the result in set B_v. It then initializes the values of p^i to zero, where p^i points to the first element in Γ_s^i (lines 3-5).

Starting from line 6, the algorithm checks all virtual nodes v in set B_v in order to assign them to the SDN domains. If the total capacity γ_{p^i} of the substrate links incident to substrate node p^i can satisfy the bandwidth requirement of virtual node v, the algorithm adds SDN i to the set S containing candidate domains for virtual node v (lines 7-9). If S is empty, the algorithm returns infeasible solution (lines 12 and 13). Otherwise, the algorithm calculates D_v^i value for each i in S, selects the SDN domain(s) with the maximum value of D_v^i, and puts them in set Q (line 14-16). If there is only one SDN set Q, then virtual node v is assigned to that SDN domain (lines 17 and 18). In case where Q has more than one SDN, the algorithm selects the SDN with the maximum value C_t^i to be assigned to the virtual node v (lines 19-22).

In lines 24 to 31, the algorithm makes the assignment of VNR to SDN domains if VN partitioning has occurred. As a result, there might be some virtual links whose endpoints lie in different domains. In such a situation, our heuristic algorithm makes the assignment as if there are three virtual links instead of the original virtual link by creating an additional virtual node assigned to each domain, referred to as virtual gateway node, in order to form a path that combines the

TABLE 3 Input parameters for VNEFSDN

Input Parameter	Explanation
$G_v = (V^v, E^v)$	Graph representing the VNR
$G_s^i = (V^s, E^s)$	Graph representing the substrate network topology of SDN domain i
b_e	Bandwidth requirement of virtual link e
$b_v = \sum_{e=vv',v' \in V^v} b_e$	Total bandwidth of the links incident to virtual node v
B_v	Set of VN vertices sorted in descending order of b_v
c_e	Capacity of link e
$C_t^i = \sum_{e \in E^s} c_e$	Total capacity of the links of SDN domain i
$\gamma_s = \sum_{e=ss',s' \in V^s} c_e$	Total capacity of the links incident to substrate node s
Γ_s^i	Set of vertices of SDN domain i sorted in descending order of γ_s
$D_v^i = \sum_{e=vv', v' \in V_a^i} b_e$	Sum of the bandwidth demands of the virtual links which are incident to virtual node v and whose endpoints (v') have hitherto been assigned to SDN domain i ($v' \in V_a^i$).
V_a^i	Set of virtual nodes assigned to SDN domain i
G_{id}^i	Set of virtual nodes assigned to SDN domain i
$\mathbf{C}^i$	Matrix containing capacities of the links of substrate network i
$\mathbf{W}^i$	Matrix containing power consumption of the links of substrate network i
$G_{id} = (V_{id}, E_{id})$	Graph representing the topology of the interdomain routing network.
G^{id}	Matrix containing the capacities of the links of the interdomain network
W^{id}	Matrix containing the power consumption of the links of the interdomain network
ϵ	Current power consumption of the network
k	A parameter for Yen's algorithm
Y	Total number of domains

Abbreviations: SDN, software-defined network; VNEFSDN, VN Embedding in Federated SDN; VN, virtual network; VNR, VN request.

assignments in the different partitions. For instance, suppose that after VN partitioning, the endpoints of virtual link uu' are in different partitions; eg, u is in V_i^a and u' is in V_j^a. Line 25 shows how our algorithm makes the VN partitioning. The subgraph induced by V_i^a (represented by $G^v[V_i^a]$) includes virtual links whose endpoints are in the same SDN domain. However, $\{uv_g^i | u \in V_i^a, u' \in V_j^a, i \neq j, uu' \in E^v\}$ shows the virtual links whose endpoints (u and u') are in SDN domains i and j. Considering these two possible types of virtual links, the top SDN forms the VN partitions and sends them to the SDN domain(s). Furthermore, in the case of VN partitioning, the traffic between the virtual nodes assigned to the different SDN domains is created as a set of source and destination pairs denoted by (s, t) and is given to the Inter_domain_Routing function (line 30).

In this case, the total energy consumption equals the sum of the energy consumption of interdomain links and active links in the SDN domain (line 30).

On the other hand, if there is no VN partition, the entire VNR given to a single SDN domain (line 32). In this case, the total energy consumption is equal to the energy consumption resulting from the VN assignment in only one SDN domain.

Each SDN controller performs VNE procedure on its input VNR and returns the amount of energy consumed by the SDN domain to the top SDN controller. The top SDN controller calculates the total energy consumption of the federated system by using the energy values returned by the SDN domains.

Phase 1: VN Assignment

1 **procedure** VN_Assignment($G^v, \Gamma_s^i, C_t^i, C^i, W^i, G_{id}, C^{id}, W^{id}, k$)
2 Sort V^v in descending order of b_v and store it in B_v;
3 **for** *all SDN domain i* **do**
4 $p^i \leftarrow 0$; // p^i points to the first element in Γ_s^i
5 **end for**
6 **for** *all* $v \in B_v$ **do**
7 **for** *all SDN domain i* **do**
8 **if** $(b_v \leq \gamma_{p^i})$ **then**
9 $S \leftarrow S \cup \{i\}$; // add SDN i to the set S containing candidate domains for virtual node v
10 **end if**
11 **end for**
12 **if** $(|S| = 0)$ **then**
13 **return** $InfeasibleVNassignment$;
14 **else**
15 calculate $D_v^i = \sum_{\substack{e=vv' \\ v' \in V_a^i}} b_e$ for each $i \in S$, select the SDN domain(s) with the maximum D_v^i value and put them in candidate set Q;
16 **end if**
17 **if** $(|Q| = 1)$ **then**
18 add the virtual node v to V_a^q and $p_q \leftarrow p_q + 1$, where q is the index of the SDN domain in Q;
19 **else**
20 select the SDN with the maximum C_t^i value among the SDNs in Q and put its index in q;
21 add v to V_a^q and $p_q \leftarrow p_q + 1$;
22 **end if**
23 **end for**

24 **if** VN partitioning is done
25 create $G_i^v(V_i^v, E_i^v)$ where $V_i^v = V_i^a \cup v_g^i$ and $E_i^v = E(G^v[V_i^a]) \cup \{uv_g^i | u \in V_i^a, u' \in V_j^a, i \neq j, uu' \in E^v\}$;
26 create $F_{id} = \{(i, j) | u \in V_i^a, u' \in V_j^a, i \neq j, uu' \in E^v\}$;
27 **for** *all SDN domains assigned a VN partition* **do**
28 $energy \leftarrow energy +$ VN_Embedding($G_s^i, G_i^v, C^i, W^i, k$)
29 **end for**
30 **return** $energy +$ Inter_domain _Routing($F_{id}, G_{id}, C^{id}, W^{id}, k$);
31 **else**
32 **return** VN_Embedding(G_s^q, G^v, C^q, W^q, k); // q: index of the SDN selected to embed the entire VNR
33 **end if**

4.2 | VNE function

Phase 2 describes the VNE procedure. This procedure takes VNR (G^v), the topology of the SDN domain (G^s), network capacity (**C**), energy consumption of the network links (**W**), and the parameter for Yen's k-shortest path algorithm (k) as input. For each virtual link, the algorithm creates an augmented graph by connecting the end nodes of the virtual links to the substrate nodes which have previously not been assigned to other virtual nodes (line 4). Then, the algorithm executes Yen's k-shortest path algorithm[35] to find a set containing k-shortest paths for the virtual nodes at the ends of that virtual link (line 5). Yen's k-shortest path algorithm[35] computes single source k-shortest loopless paths for a graph with nonnegative edge costs by employing any shortest path algorithm to find the best path and then proceeding to find $k-1$ deviations of the best path. The output of Yen's algorithm is given to the Select_Best_Path() function (line 6). This function checks all paths in this set and among the paths that satisfy the network capacity constraint (feasible solutions), it selects the one which uses the active (powered on) links.

Phase 2: VN Embedding

```
1  procedure VN_Embedding(G^v, G^s, C, W, k)
2      X ← sort E^v in non-ascending order according to B_v.
3      for each e = vv' ∈ X do
4          Create augmented graph G^s'(V^s', E^s'), where V^s' = V^s ∪ {v, v'} and
           E^s' = E^s ∪ {e = vu|u ∈ V^s and u has not been assigned to other virtual nodes before}∪
           {e = v'u|u ∈ V^s and u has not been assigned to other virtual nodes before};
5          S ← Yen_K_ShortestPath(G^s', v, v', C, k);
6          p ← Select_Best_Path(S);
7          if (p ≠ ∅) then
8              Assign v to the first and v' to the last substrate node in path p;
9              Assign virtual link vv' to path p
10             Update(C,ϵ);
11         else
12             return InfeasibleSolution;
13         end if
14     end for
15 return ϵ;
```

Phase 3: Inter-domain Routing

```
1  procedure Inter_domain_Routing(F_id, G_id, C^id, W^id, k)
2      for each (s, t) ∈ F_id do
3          S ← Yen_K_ShortestPath (G_id, s, t, C^id, W^id, k);
4          p ← Select_Best_Path(S);
5          if (p ≠ ∅) then
6              Assign path p to pair (s, t)
7              Update(C^id, ϵ);
8          else
9              return InfeasibleSolution;
10         end if
11     end for
12 return ϵ;
```

If the Select_Best_Path() function finds a path for the virtual link vv', the VNE function assigns v to the first and v' to the last substrate node in that path and updates ϵ and **C** (lines 7-10). If all virtual links in a VNR are accommodated and the virtual nodes at the endpoints of these links are assigned to substrate nodes, the algorithm calculates the energy consumption of the network and returns it (line 15). However, if the algorithm fails in finding a path for even one of the virtual links in a VNR, it returns an infeasible solution (line 12).

4.3 | VN interdomain routing function

Phase 3 describes the VN interdomain routing procedure. This procedure takes VN gateway demand (F_{id}), the topology of the interdomain routing network (G_{id}), link capacities of the interdomain network (C^{id}), energy consumption of the links of the interdomain network (W^{id}), and the parameter for Yen's k-shortest path algorithm (k) as input. For each set of source and destination pairs denoted by (s, t) in F_{id}, the algorithm executes Yen's k-shortest path algorithm to find a set containing k-shortest paths for the virtual nodes at the ends of that virtual link (line 3). The output of Yen's algorithm is given to the Select_Best_Path() function (line 4). This function checks all paths in this set and among the paths that satisfy the network capacity constraint (feasible solutions), it selects the one which uses the active (powered on) links.

If the Select_Best_Path() function finds a path for the virtual link, then it assigns path p to pairs (s, t) and updates ϵ as well as C^{id} (lines 5-7). If all virtual links in a VN gateway demand are accommodated, then the algorithm calculates the energy consumption of the network and returns it (line 12). However, if the algorithm fails in finding a path for even one of the pairs (s, t) in a VN gateway demand, it returns an infeasible solution (line 9).

5 | TIME COMPLEXITY

In this section, we present the worst-case time complexity analysis of our proposed heuristic algorithm. The number of substrate nodes is $|V^s|$, the number of substrate links is $|E^s|$, the number of virtual nodes in the VNR with maximum size is $|V^n|$, the number of virtual links is $|E^n|$, the number of VNRs is L, the number of domains is Y, and k is the parameter for Yen's k-shortest path algorithm.

In VN assignment phase, the nodes of a VNR V^v is sorted in descending order of b_v (line 2). This sorting operation is done in $O(|V^n| \cdot log|V^n|)$ time, while storing the result in B_v takes constant time for a single VNR. Lines 3 to 5 are performed for all SDN domains, yielding $O(Y)$ complexity. Lines 6 to 23 of our algorithm explain how a virtual node v is assigned to one of the SDN domains. For a single virtual node, the condition in line 8 is checked for all SDN domains, and the candidate set S is created. The complexity of lines 7 to 11 is $O(Y)$. Afterwards, in the worst case, the condition in line 12 does not hold, and line 14 is executed. Here, for each candidate SDN domain in S, the value of D_v^i is calculated. The maximum size of S is equal to the number of all SDN domains. Taking into account that the calculation of D_v^i for each SDN domain takes $(|V^n| - 1)$ iterations in the worst case, the complexity of lines 14 to 16 is $O(Y \cdot |V^n|)$. In the worst case, the condition in line 17 is not satisfied, and lines 19 to 22 are executed. Here, considering that the value of C_t^i is calculated in constant time for each candidate in set Q and the maximum size of Q is equal to the size of S, the complexity of lines 19 to 22 is $O(Y)$. Therefore, the worst-case complexity of VN assignment for a single VN of maximum size $|(V^n)|$ is $O\left(|V^n|(Y + Y \cdot |V^n| + Y)\right) = O\left(Y \cdot |V^n|^2\right)$.

In lines 24 to 33, the algorithm decides on whether to deal with the case of partitioning (lines 25-30) or the case of assigning the entire VNR to a single SDN domain (line 32). The worst case happens in the case of partitioning since extra operations to deal with interdomain links need to be done. In line 25, VNEFSDN creates subrequests, adds a virtual gateway to each subrequest, which takes $O(Y)$ time, and created E_i^v for each subrequest, which takes $(|V^n| - 1)$ iterations in the worst case, leading to the complexity $O(Y \cdot |V^n|)$. Thus, the overall complexity of line 25 is $O(Y + Y \cdot |V^n|)) = O(Y \cdot |V^n|)$. Furthermore, VNEFSDN creates the set of source and destination pairs F_{id} for interdomain links in line 26. Since each virtual link with its endpoints assigned to different SDN domains leads to a source and destination pair in F_{id}, the worst-case complexity of line 26 is $O(|E^n|)$. By denoting the complexity of *VN_Embedding* function (line 28) as C_{emb} and the complexity of Inter_domain_routing function (line 30) as C_{idr}, the complexity of our algorithm for a single VNR C_v is as follows:

$$C_v \in O\left(Y \cdot |V^n|^2 + |E^n| + Y \cdot C_{emb} + C_{idr}\right). \tag{20}$$

VN_Embedding function sorts the virtual links in the VNR in $O(|E^n| \cdot log|E^n|)$ time. Then, for each virtual link, it creates an augmented graph (line 4) in $O(|V^s|)$ time and applies Yen's k-shortest path algorithm whose worst-case complexity is $O(k \cdot |V^{s'}| \cdot (|E^{s'}| + |V^{s'}| \cdot log(|V^{s'}|)))$ according to Martins Ernesto and Pascoal Marta.[36] Since $|V^{s'}| = |V^s| + 2$ and in the worst case $|E^{s'}| = |E^s| + 2|V^s|$, the worst-case complexity of Yen's k-shortest path algorithm can be rewritten as $O(k \cdot |V^s| \cdot (|E^s| + |V^s| \cdot log(|V^s|)))$ for each virtual link. In line 6, Select_Best_Path function checks the feasibility of the k paths found by Yen's k-shortest path algorithm by considering two constraints: link capacities and node capacities. Both constraints are checked for all links of a path. The worst case occurs when a path has the maximum possible length, namely $(|V^s| - 1)$. Thus, the complexity of the path selection is $O(k \cdot |V^s|)$. Considering that lines 7 to 14 are executed in

constant time, and the sorting (line 2) and Yen's k-shortest path algorithm (line 5) are the dominant tasks, the complexity of *VN_Embedding* function is as follows:

$$C_{EMB} \in O\left(|E^n| \cdot \left(log|E^n| + k \cdot |V^s| \cdot (|E^s| + |V^s| \cdot log(|V^s|)\right)\right). \tag{21}$$

The major tasks in *Inter_DomainRouting* function are Yen's k-shortest path algorithm (line 3) and Select_Best_Path function (line 4), which are executed for each source and destination pair in F_{id}. Similar to the *VN_Embedding* function, Yen's k-shortest path algorithm is the dominant task and since $|F^{id}| = |E^v|$ in the worst case, the complexity of *Inter_DomainRouting* function is as follows:

$$C_{IDR} \in O\left(|E^n| \cdot k \cdot Y \cdot \left(|E^{id}| + Y \cdot log(Y)\right).\right) \tag{22}$$

Considering that the total number of VNRs is L, by plugging 21 and 22 in 20, the overall worst-case time complexity of VNEFSDN is as follows:

$$C_{VNEFSDN} \in O\left(L \cdot Y \cdot \left(|V^n|^2 + |E^n| \cdot \left(log|E^n| + k \cdot |V^s| \cdot (|E^s| + |V^s| \cdot log(|V^s|)\right) + |E^n| \cdot k \cdot \left(|E^{id}| + Y \cdot log(Y)\right)\right)\right). \tag{23}$$

Since $|E^{id}| \in O(Y^2)$, $|E^n| \in O(|V^n|^2)$, and $|E^s| \in O(|V^s|^2)$, the overall worst-case time complexity of VNEFSDN can be simplified as follows:

$$C_{VNEFSDN} \in O\left(L \cdot Y \cdot \left(|V^n|^2 \cdot (log|V^n| + k \cdot |V^s|^3) + |V^n|^2 \cdot k \cdot Y^2\right)\right). \tag{24}$$

6 | PERFORMANCE EVALUATION

Optimization software CPLEX[37] can provide efficient solutions to ILP problems. Whenever CPLEX finds an optimal solution, it indicates this fact in its output. If it takes too much time for CPLEX to converge to optimality, a tolerance value can be provided to the software such that the computation terminates once a solution within the given percentage of the optimal solution is found. The parameter *epgap* serves this purpose. Therefore, the ILP in (1) to (19) can be solved using CPLEX. However, in a real-life scenario, a commercial optimization software may not be available. Furthermore, CPLEX is suitable for centralized settings where all the computation occurs at a single entity. Therefore, it is not suitable for federated SDN settings. Accordingly, CPLEX solutions serve as a reliable baseline for comparison in the performance evaluation of our proposed semi-centralized heuristic algorithm for federated SDN networks.

In this section, we present the numerical evaluation of the results obtained by the execution of our ILP using CPLEX and the results of our proposed heuristic algorithm. In our experiments, we consider the impact of network size, link capacity, and the number of VNRs on energy consumption and ratio of feasible solutions. Since the problem in (1) to (19) is formulated for the first time in this paper, no other heuristic algorithm that addresses this problem exists in the literature. Therefore, we are able to compare the performance of our heuristic algorithm only with the CPLEX solutions. Furthermore, note here that ILP formulation is not for a federated scenario; its main purpose is to serve a baseline for comparison with our proposed heuristic algorithm, which is for federated SDN. As the problem size gets larger, CPLEX running times become prohibitively high. For these cases, an upper time limit can be set to CPLEX and hence if CPLEX finds an optimal solution within this time limit, then it returns an optimal solution; otherwise, it returns the solution it has found up until that time as well as the resulting gap value. This way, CPLEX solutions that are either optimal or near optimal can be obtained so that we a baseline to compare our heuristics with can be constructed. The default value of the gap parameter (epgap) is 0.0001, and it can take any value between 0.0 and 1.0. We do not set a time limit in any of the experiments. Hence, all of the obtained results are within the default gap value, which is 0.0001.

In our experiments, we use both synthetic and real topologies. While in the case of synthetic topologies, as in other studies,[1,6,19] we create both substrate and VN topologies via topology generator GT-ITM, in the case of real topologies, we partition EU-Nobel and GEANT topologies into multiple domains and use GT-ITM only for generating VN topologies. In each experiment, we execute the simulations in a federated SDN with five domains, each with the same number of substrate nodes. For the energy consumption of each intradomain physical link, we set a value uniformly randomly generated between 50 and 100 Joule. We set the interdomain link costs to approximately twice the energy consumption of intradomain link costs, ie, to 250 Joule. [13]

As for VN topologies, the number of virtual nodes in each VNR is uniformly randomly distributed between 2 and 10. CPU capacities of virtual nodes are chosen uniformly randomly between 0 and 10 processing power units. We set the required bandwidth of virtual links to values that are uniformly randomly distributed between 0 and 10 bandwidth units. The number of VNRs is 5 and the average VN graph connectivity is fixed to 0.5; in other words, each pair of virtual nodes is connected with a probability of 0:5. These values are the same in all experiments for both real and synthetic topologies except the one where the impact of that particular parameter is evaluated, in which case, the value of that parameter (for instance, the number of VNRs) is varied.

In Figure 3, we consider the impact of the number of substrate nodes on energy consumption. We vary the total number of substrate nodes from 15 to 40. We randomly generate 50 instances of each SN. We randomly generate link capacities between 100 and 150 Mbps, which is a range for which both federated and ILP scenarios yield feasible solutions for all SN sizes. Figure 3 shows that as the size of the SN increases, the amount of energy consumed in both federated and centralized scenarios decreases. We observe that the results achieved by the federated scenario are close to those achieved by the ILP scenario, demonstrating the satisfactory performance of our semi-centralized heuristic algorithm. Moreover, for the SN sizes 15 and 20, the size of each SDN domain (3 and 4, respectively) is smaller than the VNR size (5 in our case). Thus, in order to accommodate the VNR, both ILP and our heuristic have to partition the VNR and use interdomain links. However, for SN sizes that are at least 25, the probability that both algorithms accommodate the VNR in a single SDN domain without the need to use interdomain links increases. Thus, we observe a drop in the amount of energy consumption for SN sizes greater than 20.

In Figures 4 to 6, we examine the impact of the number of VNRs on energy consumption. In all experiments, we set the bandwidth request of VNR links uniformly randomly between 100 and 150 Mbps in order for both federated and CPLEX scenarios to achieve feasible solutions for all VNRs. In Figure 4, we consider 50 randomly generated instances of SN topologies, each with 50 substrate nodes. Figure 4 demonstrates that as the number of VNRs increases, the energy consumed by both federated and CPLEX scenarios increases as well. This increase is natural since a higher number of VNRs implies more virtual links and therefore necessitates the usage of more paths on the SN to accommodate the virtual links, yielding higher energy consumption. Moreover, we observe that our heuristic algorithm demonstrates close

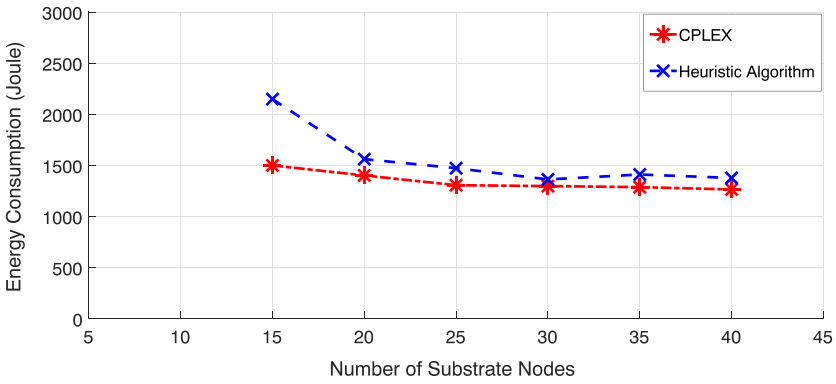

FIGURE 3 Energy consumption with respect to the total number of substrate nodes

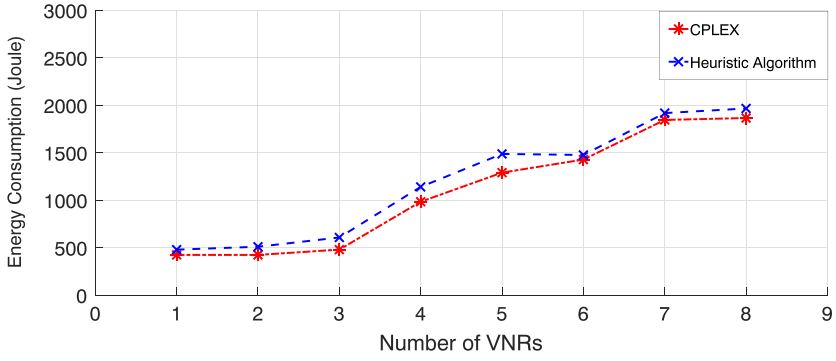

FIGURE 4 Energy consumption with respect to the number of virtual network requests (VNRs)

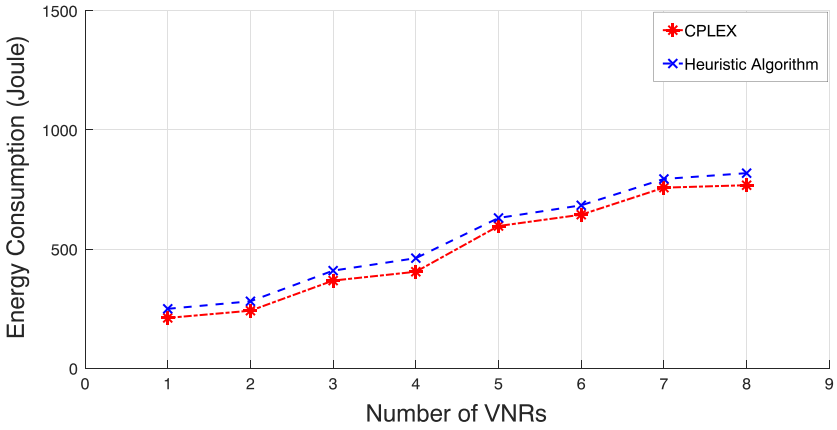

FIGURE 5 Energy consumption with respect to the number of virtual network requests (VNRs) for GEANT

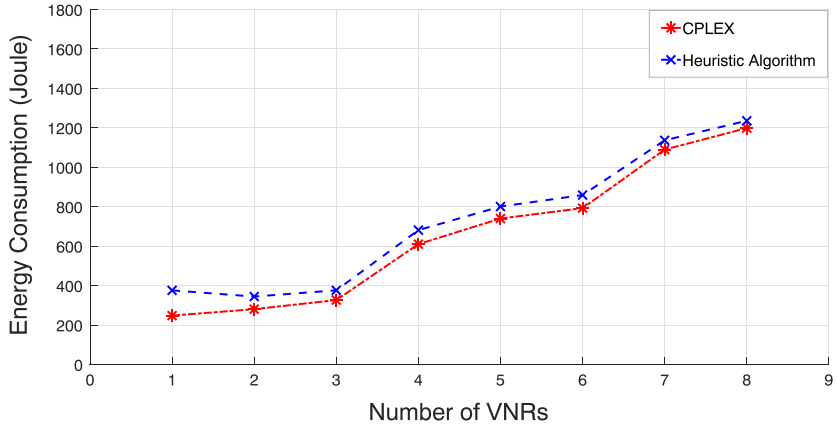

FIGURE 6 Energy consumption with respect to the number of virtual network requests (VNRs) for NOBEL-EU

performance to the CPLEX solutions, thereby demonstrating its satisfactory performance. Figures 5 and 6 demonstrate the performance on real topologies, in particular on GEANT and NOBEL-EU, respectively, taken from SNDLib.[38] GEANT has 22 nodes while NOBEL-EU has 28 nodes. We have partitioned GEANT topology into two equally sized domains and NOBEL-EU topology into four equally sized domains. Similar to the case with random topologies, the simulation results demonstrate that as the number of VNRs increases, the energy consumed by both federated and CPLEX scenarios increases as well. Besides, the performance of our heuristic algorithm is close to the performance of CPLEX solutions.

We then evaluate the impact of link capacity on energy consumption for both our heuristic algorithm and CPLEX results. The total number of substrate nodes is 30, and the bandwidth requirement of virtual links is uniformly distributed between 1 and 10 Mbps. Figure 7 shows the results of this experiment. We perform this experiment for the values of link capacities where both federated and CPLEX scenarios find feasible solutions. Each capacity value in Figure 7 corresponds to a scenario where all link capacities are equal to that particular value. Each point in Figure 7 is the average energy consumption of 50 randomly generated instances of SNs. The resulting behavior is due to the fact that when link capacities increase, our heuristic algorithm is more likely to integrate VNRs to a smaller set of active links and power off the remaining links. We also observe that the solutions found by our proposed heuristic algorithm are close to those produced by the CPLEX as link capacities increase, demonstrating its satisfactory performance.

We then investigate the impact of the link capacity on the ratio of feasible solutions. The simulation setup is the same as the previous experiment except the range of the examined link capacities. Figure 8 demonstrates the ratio of feasible solutions of our heuristic algorithm and CPLEX solutions for varying link capacity (Mbps). As the link capacities increase, the ratio of feasible solutions increases since higher link capacity facilitates the accommodation of substrate paths satisfying the bandwidth demands of the virtual links. Once the link capacities reach a point where all instances are able to generate feasible solutions, further increases in link capacity naturally does not impact the ratio of feasible solutions.

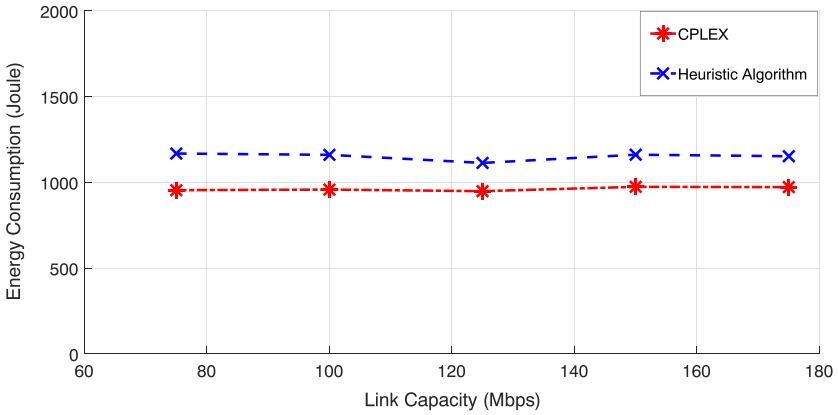

FIGURE 7 Energy consumption with respect to link capacities

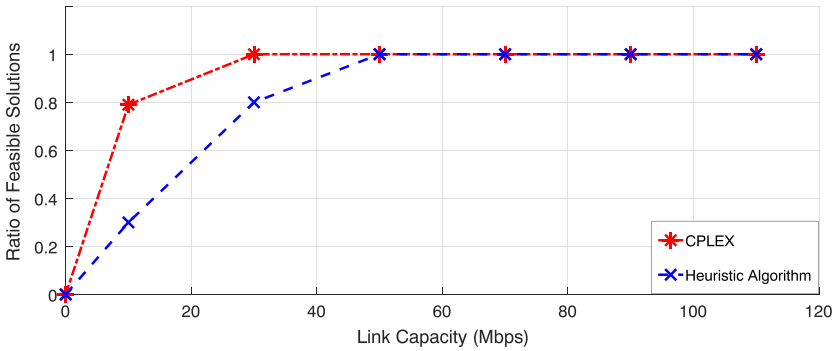

FIGURE 8 Ratio of feasible solutions for varying link capacities

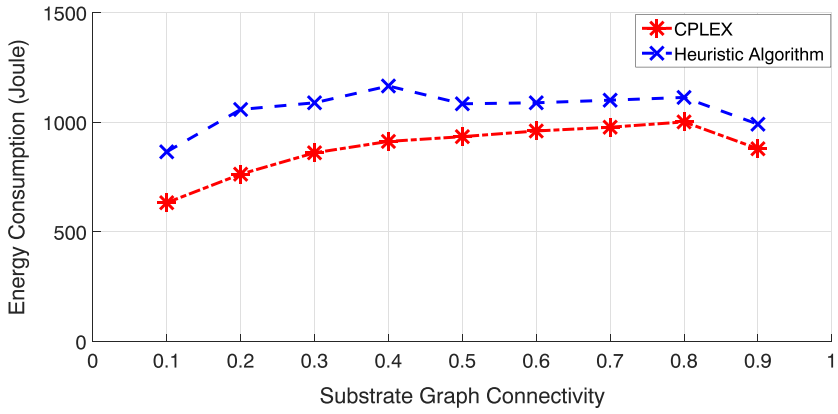

FIGURE 9 Energy consumption with respect to the substrate network (SN) connectivity

Moreover, we observe that the feasibility performance of our heuristic algorithm is satisfactory since the link capacity value that achieves 100% feasible solutions is close to the corresponding value of the solutions generated by CPLEX.

In Figure 9, we investigate the impact of SN connectivity on energy consumption. The SN connectivity refers to the probability that there exists an edge between any pair of substrate nodes. Naturally, SN density increases as SN connectivity increases. In our experiments, SN connectivity ranges from 0.1 to 0.9. We consider a network with 50 randomly generated SN topologies. We set the bandwidth capacity of substrate links randomly between 100 and 150 Mbps. The bandwidth requirement of virtual links is uniformly randomly distributed between 1 and 10 Mbps. Figure 9 shows the results of this experiment. As the SN connectivity increases, the energy consumed by both federated and CPLEX scenarios increases as well. We recognize that the solutions found by our proposed heuristic algorithm are close to those produced by the CPLEX for varying SN connectivity.

7 | CONCLUSION

In this work, we have focused on energy efficient VNE in federated SDN. We have formulated an optimization problem as an ILP that minimizes the energy consumption of the SN links while adhering to the bandwidth and CPU requirements of the VNRs. We have then proposed a semi-centralized heuristic algorithm which is tailored for federated SDN networks. To the best of our knowledge, our proposed heuristic algorithm is the first VNE heuristic in the literature for federated SDN networks. We have made a comparative numerical evaluation of our heuristic algorithm and ILP formulation by using CPLEX optimization software. Our simulation results demonstrate that our proposed heuristic algorithm yields satisfactory solutions in terms of total energy consumption in comparison with the CPLEX solutions.

As a future study, we plan to investigate the computational complexity of our problem in its particular cases and thus provide a combinatorial analysis. We also plan to evaluate the performance of our algorithm in other network topologies and formulate different ILP problems with additional constraints by taking the issues of security, survivability, and energy fairness into account. We also plan to design approximation algorithms, which have theoretically provable performance guarantee, to address energy efficient VNE in a federated SDN.

ACKNOWLEDGMENT

This work is supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under grant no. 114E245.

ORCID

Didem Gözüpek https://orcid.org/0000-0001-8450-1897

REFERENCES

1. Houidi I, Louati W, Ameur WB, Zeghlache D. Virtual network provisioning across multiple substrate networks. *Comput Netw*. 2011;55(4):1011-1023. Special Issue on Architectures and Protocols for the Future Internet.
2. Chowdhury M, Rahman MR, Boutaba R. Vineyard: virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Trans Netw*. February 2012;20(1):206-219.
3. Yu M, Yi Y, Rexford J, Chiang M. Rethinking virtual network embedding: substrate support for path splitting and migration. *SIGCOMM Comput Commun Rev*. March 2008;38(2):17-29.
4. Zhu Y, Ammar M. Algorithms for assigning substrate network resources to virtual network components. In: Proceedings IEEE Infocom 2006. 25th IEEE International Conference on Computer Communications; 2006; Barcelona, Spain. 1-12.
5. Ghazar T, Samaan N. Hierarchical approach for efficient virtual network embedding based on exact subgraph matching. In: 2011 IEEE Global Telecommunications Conference - Globecom 2011; 2011; Kathmandu, Nepal:1-6.
6. Fischer A, Botero JF, Beck MT, Meer H, Hesselbach X. Virtual network embedding: a survey. *IEEE Commun Surv Tutorials*. 2013;15(4):1888-1906.
7. Botero JF, Hesselbach X, Duelli M, Schlosser D, Fischer A, Meer H. Energy efficient virtual network embedding. *IEEE Commun Lett*. 2012May;16(5):756-759.
8. Chen X, Li C, Jiang Y. Optimization model and algorithm for energy efficient virtual node embedding. *IEEE Commun Lett*. 2015;19(8):1327-1330.
9. Su S, Zhang Z, Cheng X, Wang Y, Luo Y, Wang J. Energy-aware virtual network embedding through consolidation. In: 2012 Proceedings IEEE Infocom Workshops; 2012; Orlando, FL, USA. 127-132.
10. Guan X, Wan X, Choi BY, Song S. Ant colony optimization based energy efficient virtual network embedding. In: 2015 IEEE 4th International Conference on Cloud Networking (CloudNet); 2015; Niagara Falls, ON, Canada. 273-278.
11. Nunes BAA, Mendonca M, Nguyen XN, Obraczka K, Turletti T. A survey of software-defined networking: past, present, and future of programmable networks. *IEEE Commun Surv Tutorials*. 2014;16(3):1617-1634.
12. Hong S, Jue JP, Park P, Yoon H, Ryu H, Hong S. Survivable virtual topology design in multi-domain optical networks. *IEEE/OSA J Opt Commun Networking*. 2016;8(6):408-416.
13. Hong S, Jue JP, Zhang Q, et al. Virtual optical network embedding in multi-domain optical networks. In: 2014 IEEE Global Communications Conference; 2014Dec; Austin, TX, USA. 2042-2047.
14. Gao C, Cankaya HC, Jue JP. Survivable inter-domain routing based on topology aggregation with intra-domain disjointness information in multi-domain optical networks. *IEEE/OSA J Opt Commun Networking*. 2014;6(7):619-628.
15. Zhong Q, Wang Y, Meng L, Xiao A, Zhang H. A max-flow/min-cut theory based multi-domain virtual network splitting mechanism. In: 2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS); 2015Aug; Busan, South Korea. 392-395.

16. Chabarek J, Sommers J, Barford P, Estan C, Tsiang D, Wright S. Power awareness in network design and routing. In: IEEE Infocom 2008 - The 27th Conference on Computer Communications; 2008; Phoenix, AZ, USA.
17. Hou W, Guo L, Zheng Z, Zhang X. Green grooming in a more realistic and multidomain optical network. *J Opt Commun Netw*. 2013;5(8):858-869.
18. Hou W, Guo L, Gong X, Sun Z. Multi-domain integrated grooming algorithm for green IP over WDM network. *Comput Commun*. 2013;36(3):342-350.
19. Farooq Butt N, Chowdhury M, Boutaba R. *Topology-Awareness and Reoptimization Mechanism for Virtual Network Embedding*. Berlin, Heidelberg: Springer Berlin Heidelberg; 2010: 27–39.
20. Zhang S, Qian Z, Wu J, Lu S, Epstein L. Virtual network embedding with opportunistic resource sharing. *IEEE Trans Parallel Distrib Syst*. 2014;25(3):816-827.
21. Beşiktaş C, Gözüpek D, Ulaş A, Lokman E. Secure virtual network embedding with flexible bandwidth-based revenue maximization. *Comput Netw*. 2017;121:89-99.
22. Li R, Wu Q, Tan Y, Zhang J. On the optimal approach of survivable virtual network embedding in virtualized SDN. *IEICE Trans Inf Syst*. 2018;101(3):698-708.
23. Pyoung CK, Baek SJ. Joint load balancing and energy saving algorithm for virtual network embedding in infrastructure providers. *Comput Commun*. 2018;121:1-18.
24. Chochlidakis G, Friderikos V. Mobility aware virtual network embedding. *IEEE Trans Mob Comput*. 2017;16(5):1343-1356.
25. Leivadeas A, Papagianni C, Papavassiliou S. Efficient resource mapping framework over networked clouds via iterated local search-based request partitioning. *IEEE Trans Parallel Distrib Syst*. 2013;24(6):1077-1086.
26. Feng M, Liao J, Qing S, Li T, Wang J. COVE: co-operative virtual network embedding for network virtualization. *J Netw Syst Manag*. 2018;26(1):79-107.
27. Wanis B, Samaan N, Karmouch A. Efficient modeling and demand allocation for differentiated cloud virtual-network as-a service offerings. *IEEE Trans Cloud Comput*. 2016;4(4):376-391.
28. Beck MT, Fischer A, Botero JF, Linnhoff-Popien C, Meer H. Distributed and scalable embedding of virtual networks. *J Netw Comput Appl*. 2015;56:124-136.
29. Samuel F, Chowdhury M, Boutaba R. Polyvine: policy-based virtual network embedding across multiple domains. *J Internet Serv Appl*. 2013;4(1):6.
30. McKeown N, Anderson T, Balakrishnan H, et al. Openflow: enabling innovation in campus networks. *SIGCOMM Comput Commun Rev*. 2008;38(2):69-74.
31. Zhou B, Gao W, Zhao S, et al. Virtual network mapping for multi-domain data plane in software-defined networks. In: IEEE International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace Electronic Systems (VITAE); 2014:1-5.
32. Zaman F, Jarray A, Karmouch A. QoS aware virtual network embedding in SDN-based metro optical network. In: International conference on network-based information systems. Cham, Switzerland: Springer; 2017:408-419.
33. Tegueu AS, Abdellatif S, Villemur T, Berthou P. A dynamic resource defragmentation scheme for virtualized SDN-enabled substrate networks. In: 2017 IEEE 6th International Conference on Cloud Networking (CloudNet). Prague, Czech Republic: IEEE; 2017:1-6.
34. Osgouei AG, Koohanestani AK, Saidi H, Fanian A. Online assignment of non-SDN virtual network nodes to a physical SDN. *Comput Netw*. 2017;129:105-116.
35. Yen JY. Finding the *K* shortest loopless paths in a network. *Manag Sci*. 1971;17(11):712-716.
36. Martins ErnestoQV, Pascoal MartaMB. A new implementation of Yen's ranking loopless paths algorithm. *Q J Belg Fr Ital Oper Res Soc*. 2003;1(2):121-133.
37. IBM ILOG CPLEX; 2018.
38. Orlowski S, Pióro M, Tomaszewski A, Wessäly R. SNDlib 1.0—survivable network design library. 2007. https://doi.org/sndlib.zib.de, extendedversionacceptedinNetworks,2009.

How to cite this article: Dahir MH, Alizadeh H, Gözüpek D. Energy efficient virtual network embedding for federated software-defined networks. *Int J Commun Syst*. 2019;e3912. https://doi.org/10.1002/dac.3912