Congestion-aware and Energy-aware Virtual Network Embedding

Minh Pham, *Student Member, IEEE*, Doan B. Hoang, *Member, IEEE*, Zenon Chaczko, *Member, IEEE*

***Abstract*—Network virtualization is an inherent component of future internet architectures. Network resources are virtualized from the underlying substrate and elastically provisioned and offered to customers on-demand. Optimal allocation of network resources in terms of utilization, quality of service, and energy consumption has been a challenge. Existing solutions consider congestion control in a single-objective virtual network embedding (VNE) problem. This paper defines a multiple-objective VNE problem called the congestion-aware, energy-aware VNE (CEVNE). The aim is to seek a solution that saves cost, saves energy and avoids network congestion simultaneously. CEVNE's modelling techniques and solution approaches apply both the weighting method and the constraint method to search for pareto-optimal solutions that produce the best compromised solutions for all three objectives. Solving VNE problem is, however, NP-hard. A heuristic solution is proposed involving a two-stage coordinated CEVNE. The node-mapping algorithm searches for the sub-optimal solutions for three objectives. The link mapping process is an SDN-based heuristic algorithm that deploys a path service and a resource monitoring application on an SDN controller. The solution is realized using SDN, Segment Routing, and open network operating system platform (ONOS) technologies. The energy minimization is implemented with a registry that keeps track of active nodes and sets inactive nodes to sleep mode. The evaluation results showed that the multiple-objective CEVNE approach is feasible and achieves its goals of optimizing the resource allocation, improving the runtime, saving the energy consumption and controlling the network congestion.**

***Index Terms*—Virtual network embedding, Network virtualization, Software-defined networks, Segment Routing, Energy saving, Congestion control, Path services, Multiple-objective optimization.**

I. Introduction

Network virtualization is an effective technology for provisioning of network resources in future networks. Originally it was designed to evaluate new protocols and services and used in various research test beds like G-Lab [1] or 4WARD [2], and it is becoming an integral part of cloud/data centers and telecommunication infrastructures [3]. In a network virtualization environment, multiple virtual networks (VNs) coexist on the same underlying substrate network (SN) by virtue of virtualization of the substrate resources (nodes and links). A VN is composed of virtual nodes and virtual links that form a virtual topology. Optimal mapping of virtual networks onto the substrate network is the main challenge in resource allocation in network virtualization, and it is called the virtual network-embedding (VNE) problem.

Most of the State-of-the-Art VNE algorithms aim at solving the VNE problem by optimizing a single objective such as minimizing the overall cost, maximizing the overall revenue, or optimizing network quality of service performance. A recent study of optimization models focused on a single congestion mitigation objective in virtual networks [4]. Several research efforts have been made to solve multiple-objective VNE problems. In 2017, Zhang, et al. [5] put forward a solution to a VNE problem that aims to satisfy two objectives concurrently: minimization of the energy consumption and maximization of the revenue in a cloud data center. Houidi, et al. [6] investigated a three-objective VNE problem that minimizes the energy consumption, maximizes the availability, and the load balance of virtual machines. Clearly, optimizing a single objective is not satisfactory as the solution for one (say, congestion) may have negative impacts on other desirable outcomes such as energy consumption, revenue, etc., for the network and service providers. In fact, single-objective VNE leads to frequent ad-hoc adjustments and reconfiguration, which is resource-expensive, and interferes with the current activities on the virtual networks [3]. Realistic solutions require the optimization of multiple objectives simultaneously.

In this paper, we propose a new VNE formulation, called the congestion-aware and energy-aware VNE (CEVNE), to solve the virtual network embedding problem with multiple objectives: minimizing network traffic flow costs, preventing network congestion and minimizing the overall energy consumption for cloud network data-centers' operation. CEVNE modeling techniques apply the weighting method with the positive weights and the constraint method with the binding constraint on three objectives to create a program that generates the pareto-optimal solutions for involved objectives. Solving the VNE problem is NP-hard as it relates to the multi-way separator problem [7] [8]; a heuristic solution is proposed involving a two-stage coordinated CEVNE, a node-mapping process and a link mapping process. The node-mapping process will solve the CEVNE problem using the gnu linear programming kit (GLPK) solver on the augmented substrate network with meta nodes and meta links, and the link-mapping process will be realized as a composite application on an SDN controller with an SDN substrate network deploying Segment

M. Pham is with the Faculty of Engineering and IT, University of Technology Sydney, NSW 2007, Australia
(e-mail: minh.pham@student.uts.edu.au)

Doan B. Hoang is with the Faculty of Engineering and IT, University of Technology Sydney, NSW 2007, Australia, (e-mail: doan.hoang@uts.edu.au)

Zenon Chaczko is with the Faculty of Engineering and IT, University of Technology Sydney, NSW 2007, Australia
(e-mail: Zenon.Chaczko@uts.edu.au)

Routing (SR). The CEVNE's feasibility depends on the SDN's logical centralization. The energy saving is implemented by keeping track of active nodes in the VNR embedding and setting inactive nodes to sleep mode. The congestion avoidance spreads the network traffic on different paths to minimize the maximum link usage.

The test results show that CEVNE delivers improved performance relative to a single-objective VNE in terms of congestion control, cost saving and energy saving over the substrate network; and its congestion control objective is beneficial to the VNE in challenging, near-congestion networks. Potentially, CEVNE offers a realistic and cost-effective approach in data centers. The main contributions of the paper are summarised as follows:

1. Mathematically, we formulate a multi-objective virtual network embedding problem, the CEVNE, with three objectives: resources saving, energy saving, and congestion avoidance.
2. We present the modelling techniques and the solution approaches that use both the weighting method and the constraint method to search for the pareto-optimal solutions that produce the best compromised solutions for all three objectives [9].
3. We propose a multi-objective heuristic algorithm for the node mapping process and an SDN-based heuristic algorithm for the virtual link embedding process.
4. We develop an SDN application over the ONOS controller to implement the CEVNE link mapping process.
5. We evaluate the CEVNE node mapping thoroughly in networks of different topologies, sizes, and workloads. We evaluate the CEVNE link mapping process on an SDN substrate network configured as a leaf-spine fabric running SR protocol.

The structure of the paper is as follows. Section 2 presents related work to the CEVNE problem. Section 3 describes the CEVNE problem formulation with multiple objectives, constraints, and SDN, SR technologies. Section 4 describes CEVNE heuristic solutions with the node mapping, link mapping, and integrated algorithms. Section 5 presents the evaluation scenarios, the benefits of CEVNE's congestion control in scarce-resource, near-congestion networks, and the CEVNE node mapping performance results. Section 6 describes the realization and evaluation of the CEVNE link mapping process. Section 7 presents the discussion and future work, and section 8 concludes the paper.

II. Related works

On the VNE problem, Zhu and Ammar [10], Chowdhury, et al. [11], Yu, et al. [12], and Cheng, et al. [13] have made profound contributions to its solutions.

Zhu and Ammar [10] investigated the challenge in network virtualization, which is assigning the substrate resources to the virtual networks (VNs) efficiently and on-demand. The authors proposed algorithms to embed VNs without reconfiguration (VNA-I) and with reconfiguration (VNA-II).

Chowdhury, et al. [11] proposed ViNEYARD, a set of online virtual network embedding algorithms for embedding nodes and links. ViNEYARD algorithms consist of the deterministic virtual network algorithm (D-ViNE), randomized VN algorithm (R-ViNE), D-ViNE load balancing (LB) algorithm, R-ViNE-LB algorithm and window-based (Win) VN embedding algorithm. The authors introduced the concept of the augmented substrate graph in their problem formulation.

Yu, et al. [12] investigated the design of a substrate network to simplify the VN embedding process. The proposed approaches allow the substrate network to split a virtual link into multiple substrate paths and exploit the path migration to re-optimize the resource allocation.

Cheng, et al. [13] proposed a model using the Markov random walk (MRW) to rank a node based on its resource and topological attributes. The node embedding algorithm is designed to map the virtual nodes on substrate nodes based on the ranking. The link embedding is implemented on shortest paths, using the breadth-first search method with back-tracking.

Fischer, et al. [3] surveyed VNE algorithms and classified them using different criteria. They were classified as centralized or distributed using the embedding algorithm. They were classified based on objectives such as the optimal quality-of-service in terms of bandwidth, delay, optimal infrastructure costs, survivable VNEs. Some VNEs have their performance improved via the coordination of node and link mapping processes; these are called coordinated VNEs. In contrast, uncoordinated VNEs have the node mapping and link mapping implemented in the un-coordinated way.

Fischer, et al. [14] proposed an energy-efficient virtual network embedding based on minimizing the number of active substrate nodes. They consider a node is active if one or more virtual nodes are mapped to it. The authors set up the link weight for each link based on whether it is connected to an inactive node and choose paths with minimum weight. They showed that their algorithm was comparable with other algorithms, but used significantly less active nodes. Özbek, et al. [15] proposed an energy-aware routing algorithm based on the same concept of active nodes in the SDN networks.

Elias, et al. [4] proposed an optimization model to mitigate the congestion in the virtual network functions (VNF) in the cloud environment. The congestion control functions include minimizing the total congestion cost in the link usage and minimizing the congestion ratio. The performance evaluation shows that the proposal is efficient in controlling congestion in the virtual networks. Recently, the authors have extended their work to the distributed cloud environments [16]. However, it is a single-objective optimization problem.

With the advance of software defined technology, researchers have renewed their interest in the VNE problem for the software defined networks. Haghani, et al. [17] proposed the VNE algorithms for the SDN substrate networks, aiming to maximize business profiles and minimize delays in the embedded virtual networks (VN). The authors decomposed the problem into two stages. The first stage focuses on maximizing the profit. The result of the first stage is used in the second stage that focuses on minimizing the delay in the embedded VNs. However, they did not consider the ternary content addressable memory (TCAM) problem of OpenFlow (OF) switches.

Li, et al. [18] proposed a self-adaptive VNE (SA-VNE) in the SDN substrate networks, focusing on three types of virtual network requests (VNR): high bandwidth, low latency, and both high bandwidth and low latency. An SDN-based VNE mapping framework is proposed to centralize the allocation of

the network resources. The evaluation showed that the SA-VNE could make full use of the resources, improved the revenue, and handled multi-demands effectively. However, the proposal did not take advantage of SDN's centralized network knowledge for handling the congestion.

Bays, et al. [19] investigated VNE in the SDN networks. To manage the TCAM memory's usage efficiently, each VNR is required to specify a usage profile for its requested VN (the network policies, traffic patterns). Based on this profile, resources (CPU, TCAM memory, bandwidth) are allocated accordingly. The evaluation showed that the scheme is satisfactory in embedding VNs into SDN networks.

Blenk and Kellerer [20] investigated the SDN virtualization processes that allow multiple virtual SDN networks (vSDNs) to be constructed and share the same underlying infrastructure. A virtualization layer was designed to manage the vSDNs using the network hypervisors. The proposed design, however, did not consider the traffic engineering aspects in these vSDNs.

Dahir, et al. [21] proposed an energy efficient VNE solution in multi-domain SDNs. The optimization problem is formulated as a mixed-integer linear program (MILP). A heuristic algorithm was designed to solve the problem in polynomial time. The heuristic exploited the hierarchy of the SDN controllers to allocate the cross-domain resources to the VNRs. The solution, however, did not consider the network congestion issue.

Rout, et al. [22] proposed an energy-aware routing algorithm in an SDN network using the SDN controller. Depending on the incoming traffic, the controller determines a proper link rate for each link. As a considerable amount of energy can be saved when the links operate at low rates, the overall network's energy consumption was much reduced. However, the problem was not formulated as an optimization problem.

To the best of our knowledge, there are no multi-objective VNE proposals that focus on congestion avoidance, energy saving and cost saving which are implemented in an SDN substrate network running Segment routing.

III. CEVNE Problem Formulation

TABLE I presents all notations used in the whole section III except sub-section III.B.4.

	TABLE I: NOTATIONS & EXPLANATION
$G_S(N_S, E_S)$	The substrate network modelled as an undirected graph G_S with N_S substrate nodes and E_S substrate links
$G_V(N_V, E_V)$	The virtual network request modelled as an undirected graph G_V of N_V virtual nodes and E_V virtual links
n_s	a substrate node, $n_s \in N_S$
e_s	a substrate link, $e_s \in E_S$
$c(n_s)$	the CPU capacity of $n_s \in N_S$
$bw(e_s)$	the bandwidth capacity of $e_s \in E_S$
n_v	a virtual node, $n_v \in N_V$
e_v	a virtual link, $e_v \in E_V$
$c(n_v)$	the CPU demand of virtual node $n_v \in N_V$
$bw(e_v)$	the bandwidth demand of $e_v \in E_V$
$R_N(n_s)$	the CPU residual capacity of $n_s \in N_S$ after n_s embeds virtual nodes of several VNRs
P_S	all substrate paths in G_S
$p_s(s,t)$	all substrate paths between nodes $s, t \in N_S, p_s(st) \in P_S$
$p_s(e_s)$	all substrate paths passing the substrate link $e_s \in E_S$; $p_s(e_s) \in P_S$
$M: G_V \to G_S$	the embedding process of the VNR onto the substrate network
$M(M_N, M_L)$	the embedding process consists of the node mapping M_N and link mapping M_L
$R_{rev}(G_v)$	The revenue received when the VNR G_V is embedded.
$R_E(e_s)$	The residual bandwidth of substrate link e_s after e_s embeds the virtual links in several VNRs.
$R_E(p_s(e_s))$	the bandwidth residual of substrate path p_s is equal to the minimal residual bandwidth of substrate links in the path $p_s(e_s)$: $R_E(p_s(e_s)) = \min_{e_s \in p_s} R_E(e_s)$
$bw(e_v, e_s)$	the bandwidth of $e_s \in E_S$ allocated to $e_v \in E_V$
i	the index of virtual links, $1 \le i \le \lvert E_V \rvert$
$bw(\mathrm{e}_{\mathrm{v}}^{\mathrm{i}})$	the bandwidth demand of the ith flow on $e_v \in E_V$
$\mathrm{f}_{\mathrm{uv}}^{\mathrm{i}}$	the flow of virtual link ith on the substrate link (u, v)
s^i, s_i	The ith virtual flow's source node
$t^i,\ t_i$	The ith virtual flow's destination node
$c_w(m)$	the CPU resource of substrate node $m \in N_S$ allocated to virtual node $w \in E_V$
r	the network congestion ratio
R	the max congestion ratio, which is the upper bound of r
x_{uv}	a binary variable is equal to 1 if $\sum_i (f_{un}^i + f_{nu}^i) > 0$, otherwise 0
y_n^{uv}	A binary variable denoting an active node; is equal to 1 if $\sum_i (f_{un}^i + f_{nu}^i) > 0\ OR \sum_i (f_{vn}^i + f_{nv}^i) > 0$, otherwise 0
$\mu(n_v)$	The meta node of virtual node $n_v \in N_V$ has unlimited CPU resources
δ	a very small positive number to prevent dividing by 0
$\Omega(\mathrm{n_v})$	the cluster of the virtual node $n_v \in N_V$
N_{S1}	the set of substrate nodes and meta nodes
E_{S1}	the set of substrate links and meta links

A. The virtual network embedding problem

1) Substrate networks and Virtual networks

A substrate network (SN) is modelled as an undirected graph $G_S = (N_S, E_S)$. Each substrate node $n_s \in N_S$ has a CPU capacity $c(n_s)$; each substrate link $e_s \in E_S$ has a bandwidth capacity $bw(e_s)$.

Users send a virtual network request (VNR) to a service provider to acquire a virtual network (VN). A VN is modelled as $G_V = (N_V, E_V)$. Each virtual node $n_v \in N_V$ has a CPU demand $c(n_v)$; each virtual link $e_v \in E_V$ has a bandwidth demand $bw(e_v)$.

2) VNE process

The VNE is the mapping of a VNR onto the SN:

$$M: G_V \to G_S \tag{1}$$

There are two mapping processes, the node mapping M_N and the link mapping M_L: $M = (M_N, M_L)$. Within a VNR, each virtual node n_v can be mapped to only one substrate node n_s as the virtual link e_v connecting the virtual nodes has the bandwidth demand $bw(e_v)$. A substrate node can embed multiple virtual nodes belonging to multiple VNRs if it satisfies the CPU demands. The condition to embed successfully a virtual node n_v on a substrate node n_s is as follows.

$$R_N(n_s) \ge c(n_v) \tag{2}$$

Similarly, a virtual link e_v is embedded successfully onto the substrate path $p_s(e_s)$ based on the condition:

$$R_E\big(p_s(e_s)\big) \ge bw(e_v) \tag{3}$$

Similar to the previous work [12], the revenue of a VNR after it has been embedded is defined as follows.

$$R_{rev}(G_v) = \sum_{n_v \in N_v} c(n_v) + \sum_{e_v \in E_v} bw(e_v) \quad (4)$$

B. Overview CEVNE multiple objectives

CEVNE is an online, two-stage coordinated VNE that focuses on saving cost, saving energy and avoiding congestion objectives. As an online VNE, CEVNE embeds VNRs in real-time when they arrive using its node and link embedding processes. Each objective is presented as follows.

1) Cost saving objective:

CEVNE aims for a least cost solution in terms of SN resources that are used in the embedding VNRs. CEVNE applies the load-balancing resource consumption that is proposed by Chowdhury, et al. [11] and is expressed as follows.

$$\min \Big(\sum_{(u,v) \in E_S} \frac{\sum_i f_{uv}^i}{R_E(u,v) + \delta} + \sum_{w \in N_S} \sum_{m \in N_V} \frac{c_w(m)}{R_N(w) + \delta} \Big) \quad (5)$$

In (5), the first term denotes the virtual link demand's load balancing over the substrate links' residual resources, the second term denotes the virtual node demand's load balancing over substrate nodes' residual resources. The cost saving objective focuses on minimizing the resource cost (5). Mijumbi, et al. [23] showed that the load balancing approach can reduce the fragmentation in the substrate networks. Minimizing the resource cost is equivalent to maximizing the number of VNRs embedded onto the same SN, hence, the revenue is increased. Although ViNE-LB algorithms support splittable path mappings, CEVNE supports un-splittable path mappings in our SDN-based link mapping algorithm, and the splittable one is for future work.

2) Energy saving objective

CEVNE focuses on saving energy consumption in the VNE process. This objective is implemented by minimizing the number of active substrate nodes that embed virtual nodes in VNRs. [14], [24] showed that the routers consume the major part of the energy in the data center. Inactive nodes will operate in the sleep mode or in the low power idle (LPI) mode [25] based on IEEE 802.3az. A router's power consumption is modeled as a base with no traffic load, and a dynamic component that is dependent on its active interfaces (ports) and traffic loads. CEVNE's energy saving objective is expressed using y_n^{uv} as follows.

$$\min \sum_{uv \in E_S} \sum_{n_s \in N_S} y_{n_s}^{uv} \quad (6)$$

3) Congestion avoidance objective

Network congestions imply the conditions that the amount of traffic injected into the network approaches or exceeds the capacity limits of the network handling resources. They affect severely the performance of the substrate networks and the embedded virtual networks in terms of throughput, response time, and services completion. Different methods are investigated to control the network congestion. The congestion-ratio minimization approach is preferred over the congestion handling based on the link cost [26] as CEVNE handles traffic on a flow basis. CEVNE aims to avoid the network congestion by minimizing *r*, the congestion ratio [27], [28].

$$r = \max_{uv \in E_S} \Big(\sum_i \frac{f_{uv}^i + f_{vu}^i}{R_E(uv)} \Big) \quad (7)$$

The congestion avoidance objective is to minimize the congestion ratio:

$$\min \ r \quad (8)$$

subject to:

$$\sum_i (f_{uv}^i + f_{vu}^i) \ \leq r * R_E(uv) \ \ \forall (uv \in E_S) \quad (9)$$

$$f_{uv}^i - f_{vu}^i = \begin{cases} bw(e^i) & if \ u = s^i \\ 0 \ if & u \neq s^i, u \neq t^i \\ -bw(e^i) & if \ u = t^i \end{cases} \quad (10)$$

The constraint (9) is derived from (7). In (9), the bound of link resources is updated to apply the congestion ratio r times the link capacity. (10) is the flow conservation: for an intermediate node (not source or destination) the total flow into the node is equal to the total flow out of that node.

4) The maximum congestion ratio R

TABLE II presents notations used only in this sub-section

	TABLE II: NOTATIONS & EXPLANATION
$G(V,E)$	directed network with V nodes and E links
$Q \subseteq V$	the set of edge nodes in G
c_{ij}	the link capacity of the substrate link $(i,j) \in E$
y_{ij}	the total load on the substrate link $(i,j) \in E$
t_{pq}	the traffic demand from node p to node q; $p,q \in Q$
r	the congestion ratio, which is calculated as the max link utilization of all links $(i,j) \in E$: $r = \max_{(ij \in E)} \left\{ \frac{y_{ij}}{c_{ij}} \right\}$
T	the traffic demand matrix, $T = \{t_{pq}\}$, $\forall p,q \in Q$
α_p	the total of out-going traffic at the edge node p for all t_{pq}
β_q	the total of in-coming traffic at the edge node q for all t_{pq}

In the direct network $G(V,E)$, the optimal routing problem seeks to optimize the routing of the traffic between edge nodes $(p,q) \in Q$ that maximizes the link utilization but avoids network congestion. The network demand is given by the traffic demand matrix T. The total traffic of incoming and outgoing at the edge node $p,q \in Q$ is constrained by α_p and β_q respectively

$$\sum_q t_{pq} \leq \alpha_p, \ p \in Q; \ \sum_p t_{pq} \leq \beta_q, q \in Q$$

The optimal congestion ratio, if it exists, will be used as the benchmark to set the value for R, which denotes the maximum congestion ratio in our VNE problem. We use the hose demand model as the traffic demand in VNE is almost unknown except that the total of all virtual network traffic is limited by the substrate link capacities.

C. CEVNE problem formulation

The notations are defined at the beginning of section III.

1) The augmented substrate network

CEVNE uses the augmented substrate network for the coordination of node and link mapping that is proposed by Chowdhury, et al. [11]. Each virtual node n_v is used to create a cluster $\Omega(n_v)$ of the substrate nodes that satisfy its CPU demand. A meta node m is created for each virtual node n_v. Meta links are created between the meta node m and all substrate nodes n_s in the cluster.

In the augmented substrate network, the total number of nodes N_{S1} includes the meta nodes m and the substrate nodes N_S.

$$N_{S1} = N_S \cup \{m = \mu(n_v) | n_v \in N_V\}$$

The total number of links E_{S1} includes the meta links and substrate links E_S.

$$E_{S1} = E_S \cup \{(\mu(n_v), n_s), n_s \in \Omega(\mathrm{n_v})\}$$

2) The problem statement

It is given the substrate network (SN) with a set of N_S substrate nodes and E_S of substrate links, and the set of substrate paths P_S

$$G_S = \{N_S, E_S, P_S\}$$

Each substrate node $n_s \in N_S$ has a CPU resource. Each substrate link $(u, v) \in E_S$ has a bandwidth resource.

A given virtual network request (VNR) demands N_V virtual nodes and E_V virtual links:

$$G_V = \{N_V, E_V\}$$

Each virtual node $n_v \in N_V$ has a CPU demand. Each virtual link $(u, v) \in E_V$ has a bandwidth demand.

It is given that the VNE process consists of two separate processes: the node embedding and link embedding processes. An augmented substrate network is built from the SN with N_{S1} nodes, and E_{S1} links.

3) CEVNE problem formulation

The CEVNE programming model applies the weighting methods on the cost saving and energy saving objectives and the constraint method on the congestion avoidance objective (Cohon [9] and Williams [29]). The cost saving and the energy saving objectives are both related to the embedded substrate nodes; the weighting method is applied to create a combined program for CEVNE. Each of the objective terms is normalized to have the same ratio unit and has the base weight applied. These two objectives form the CEVNE objective as follows.

Objective function:

$$\begin{aligned}\text{minimize } (& \sum_{\mathrm{uv}\in \mathrm{E_S}} \left(\frac{1}{\mathrm{R_E(u,v)} + \delta}\right) \sum_{\mathrm{i}} \mathrm{f_{uv}^i} \\ & + \sum_{\mathrm{w}\in \mathrm{N_S}} \left(\frac{1}{\mathrm{R_N(w)} + \delta}\right) \sum_{\mathrm{m}\in \mathrm{N_{S1}}\setminus \mathrm{N_S}} \mathrm{x_{mw} c(m)} \\ & + \sum_{\mathrm{uv}\in \mathrm{E_s}} \left(\frac{1}{\mathrm{card(N_S)}}\right) \sum_{\mathrm{n_s}\in \mathrm{N_S}} \mathrm{y_{n_s}^{uv}}) \end{aligned} \quad (11)$$

In (11), the first two terms include the costs of the virtual links and virtual nodes that are specified as the ratios to load balance. The model selects the links and nodes with the maximum residuals of bandwidth, and CPU resources. The last term tries to limit the ratio of active nodes on the total number of substrate nodes to save energy; the terms have been specified in (5) (6).

For the congestion avoidance objective, to integrate with the combined model and to form the overall multi-objective formulation, the constraint method is applied. As a constraint, the objective does not appear in the expression of the objective function (11). The congestion avoidance objective is transformed into a constraint by setting an upper limit R on the congestion ratio r. This is a binding constraint if the solution exists. Therefore, the CEVNE programming model will generate the noninferior solutions [9] if they exist.

Subject to:

$$\sum_i (f_{uv}^i + f_{vu}^i) \le r * R_E(uv) \quad \forall (uv) \in E_S \quad (12)$$

$$0 \le r \le R \quad (13)$$

$$0 \le R \le 1 \quad (14)$$

In constraint (12), the bound of the link resources is updated to apply the congestion ratio r times the link capacity [27]. Constraint (13) denotes the limit of variable r that is R as presented in III.B.4. Constraint (14) is the limit of R.

Other constraints:

$$R_N(w) \ge x_{mw}\, c(m), \quad \forall m \in N_{S1}\setminus N_S, \forall w \in N_S \quad (15)$$

$$\sum_{w\in N_{S_1}} f_{uw}^i - \sum_{w\in N_{S_1}} f_{wu}^i = 0 \ \forall (i,u) \in N_{S_1}\setminus\{s_i, t_i\} \quad (16)$$

$$\sum_{w\in N_{S1}} f_{s_i w}^i - \sum_{w\in N_{S1}} f_{w s_i}^i = bw(e_v^i) \quad \forall i \quad (17)$$

$$\sum_{w\in N_{S1}} f_{t_i w}^i - \sum_{w\in N_{S1}} f_{w t_i}^i = -bw(e_v^i) \quad \forall i \quad (18)$$

$$\sum_{w\in \Omega(m)} x_{mw} = 1 \quad \forall m \in N_{S1}\setminus N_S \quad (19)$$

$$\sum_{m\in N_{S1}\setminus N_S} x_{mw} \le 1 \quad \forall w \in N_S \quad (20)$$

$$f_{uv}^i \ge 0 \quad \forall u, v \in N_{S1} \quad (21)$$

$$x_{uv} \in \{0,1\} \ \forall u, v \in N_{S1} \quad (22)$$

$$y_{n_s}^{uv} \in \{0,1\} \ \forall u, v \in N_S \quad (23)$$

Constraint (15) ensures the substrate node's CPU residual resource must satisfy the virtual node's CPU demand for the virtual node embedding [11].

Constraints (16), (17) and (18) are about the flow conservation. Constraint (16) ensures that for any internal nodes u, w, which are not the source node s_i or the destination node t_i, the net flow at these nodes is 0. Constraint (17) ensures that the total flow at the source node s_i is equal to the required bandwidth of the virtual link. Constraint (18) ensures that the total flow at the destination node t_i is equal to the required bandwidth of the virtual link in the negative sign.

Constraint (19) and (20) ensure that in the augmented substrate network, only one substrate node w is selected for the meta node m [11].

Constraint (21) is the limit of the flow f_{uv}^i. Constraints (22) and (23) are the integer constraints of the variable x_{uv} and variable $y_{n_s}^{uv}$ accordingly.

Solving the MIP is computationally intractable. Hence, the MIP is relaxed to a LP so it can be solved in polynomial time. The MIP relaxation is carried out by changing the integer constraints into the linear constraints. Clearly the solutions to the relaxed problem, if they exist, are only sub-optimal solutions. The integer constraints (22) and (23) have been relaxed into the linear constraints as in (24) and (25).

$$0 \le x_{uv} \le 1 \quad \forall u, v \in N_{S1} \quad (24)$$

$$0 \le y_{n_s}^{uv} \le 1 \ \forall u, v \in N_S \quad (25)$$

Solving the relaxed LP will give sub-optimal solutions if they exist. A heuristic algorithm is required to find the approximation for the integer variables of the original MIP.

D. The technologies that enable CEVNE

1) Software-defined networks

The CEVNE investigates the VNE problem in SDN networks. The SDN architecture includes three layers: the data layer consists of Openflow (OF) network devices, the control layer includes SDN controllers, and the application layer consists of applications built on top of the controller. Recently SDN is applied to traffic engineering (TE) because of its ability to steer the network traffic at both coarse-grained and fine-grained levels [30].

CEVNE exploits SDN's TE solutions in addressing the congestion control objective. Based on the SDN's programmability, the virtual link embedding process is realized as a composite service on top of the SDN controller. For routing, OF switches store flow rules in their TCAM. As TE routing algorithms would result in a large number of flow rules in SDN switches, a huge amount of expensive TCAM memory is required. Segment routing protocol is deployed to reduce the memory requirement.

2) Segment routing

Segment routing (SR) is a source routing protocol that allows the packets to be forwarded with minimal retaining of the network states. In SR, a source node specifies the traversal path for a packet in its header using a list of labels or segment identifiers (SID). In SR, routers along the path do not need to store routing rules in the routing tables, they only need to process the labels using simple pop, push or continue operations. SR allows alternative paths between the two nodes. In an SDN network, SR replaces the flow rules in the TCAM memory with the list of SIDs in the packet's header, leading to a reduction in TCAM storage costs. SR also speeds up the process of changing routes or adding new routes to the network [30]. CEVNE exploits the flexibility, agility, programmability of SDN and SR in addressing the congestion control issue in our virtual link mapping solution.

IV. CEVNE Algorithms

A. CEVNE node mapping algorithm

The CEVNE node mapping algorithm (CEVNE NoM) solves the relaxed LP to find the sub-optimal solutions (SOPS) in polynomial time if they exist. The relaxed LP is solved with the GLPK in the Alevin framework. In the SOPS, the variables may take on linear values as their integer constraints have been relaxed. The D-ViNE rounding algorithm [11] is used to convert back these linear values to integer values, approximating the integer variables of the original MIP problem. As the CEVNE has been formulated based on the rules to generate noninferior solutions, the SOPS and their approximations reflect the trade-offs between three objectives.

The input to the Algorithm 1 is the CEVNE mathematical model that combines the three objectives, then is relaxed to a LP; the augmented substrate network as presented in section III.C.1, and the value of R (line 2 - 4). Each virtual node is used to create a cluster of substrate nodes based on CPU demands (line 9 - 11) [11]. GLPK is invoked to solve the CEVNE relaxed formulation (line 12). GLPK returns the sub-optimal solutions, if they exist, with values of (linear and relaxed) variables, and the sub-optimal values of each objective. The approximation for the integer variable x is calculated using the rounding function (line 14), which is in D-ViNE algorithm [11]. In the rounding function, the virtual node's cluster is used to find the substrate node with the highest weight. The weight is calculated as a product of the value of x and the total flow passing the meta link in both directions. The CEVNE NoM returns the embedded substrate nodes of virtual nodes in the VNR.

B. CEVNE link mapping algorithm

The CEVNE link-mapping algorithm (CEVNE LiM) embeds the virtual links on top of the substrate paths that satisfy the demands and the three objectives. CEVNE LiM utilizes the path services in the SDN controller, and the monitor application to actively select the best path for each virtual link in the VNR.

The inputs of CEVNE link embedding algorithm are the virtual links in the VNR, the embedded substrate nodes (line 2 –3). At first, the path service is invoked to retrieve the shortest paths between nodes (u, v) (line 8). The select path function is called to return the first non-congested path among the shortest paths (line 11), the CEVNE LiM returns with a success status. Otherwise, the path service is invoked with a different weight to retrieve another set of paths between nodes (u, v) (line 13-14). They are sorted by the path length. The select path function is called to return the first non-congested path (line 15) among the second set of paths, the CEVNE LiM returns with a success status (line 17). Otherwise, an alert is sent to the operator, and the function exits with a failure status (line 18 - 19).

This process is repeated for every virtual link in the VNR. If the embedding has failed on a virtual link, the VNR is rejected. The select path function reflects the monitor application checking for congestion on a substrate path (line 22 – 28).

C. CEVNE mapping algorithm

The CEVNE algorithm includes the CEVNE NoM algorithm and CEVNE LiM algorithm as presented in Algorithm 3. A registry is used to keep track of the active nodes of each VNR. Inactive nodes are filtered using the SDN topology service, and are put in sleep mode. First, CEVNE NoM is invoked (line 2). If it is failed, the VNR is rejected (line 3). Otherwise, CEVNE LiM is invoked (line 4); if it is failed, the VNR is rejected (line 5). Otherwise, after each VNR is embedded successfully, the registry keeps track of the active nodes of both the NoM and LiM processes (line 6). The CEVNE algorithm operates in a virtualization platform that will keep track of VNRs that have been processed. The residual resources of substrate nodes are calculated (line 7 - 9). So are the residual resources of substrate links (line 10 - 12). Finally, inactive nodes are set to the sleep mode (line 13).

V. CEVNE Node mapping and Evaluation Results

In this section, we focus on the evaluation of the CEVNE NoM process. Firstly, the evaluation setup is presented, secondly, near-congestion scenarios are presented, in which resources are scarce and CEVNE's congestion control is beneficial to the VNR embedding, and lastly, the simulation and test scenarios used to evaluate the CEVNE NoM process is presented.

A. CEVNE evaluation setup

To set up the evaluation for CEVNE, the Alevin framework [31] is used as a simulation tool to generate network topologies for both SNs and VNs, and generate network resources,

***Algorithm 1**: CEVNE node mapping algorithm*

Input
  CEVNE objective, is relaxed
  The augmented substrate network
  The value of R
Output
  The embedded substrate nodes

Begin
  For virtual node n_v in N_V
    Create a cluster for virtual node n_v
  End for
  Solve the CEVNE formulation using GLPK
  // the rounding function
  Applying only the rounding function in D-ViNE
  Return the embedded substrate nodes
End

***Algorithm 2:** CEVNE link mapping algorithm*

Input
  Virtual link $(i, j) \in E_V$, the bandwidth demands
  Embedded nodes $(u, v) \in E_S$, the bandwidth capacities
Output
  Embedded substrate paths $p_{u,v} \in P_S$ for virtual link (i, j)

Begin
  Retrieve shortest path between (u, v) assign to $P_{u,v}$;
  path = select_path($P_{u,v}$);
  If path is not empty
    Return path;
  End if
  Retrieve other paths between (u, v), sort them by
  length, assign to $P_{u,v}$;
  path = select_path($P_{u,v}$);
  If path is not empty
    Return path;
  Else alert "all paths are congested'
    Exit with failure
  End if
End
Function select_path ($P_{u,v}$)
  For each $p_{u,v} \in P_{u,v}$
    If $p_{u,v}$ is not congested
      Return $p_{u,v}$
    End if
  End for
End function

***Algorithm 3:** CEVNE algorithm $(G_V = N_V, E_V), (G_S = N_S, E_S)$*

Begin
  invoke CEVNE node mapping algorithm
  if VNR is rejected, exit
  invoke CEVNE link mapping algorithm
  if VNR is rejected, exit
  register active nodes into the active-node registry
  for substrate node $n_s \in N_S$
    $R_N(n_s) = cpu(n_s) - cpu(n_v)$
  end for
  for substrate link $l \in E_S$
    $R_E(e_s) = bw(e_s) - bw(e_v)$
  end for
  set inactive substrate nodes to sleep mode
End

demands for these network elements.

1) Topology generation

Waxman method is used to create SN and VN topology. First, it creates nodes and their coordination in the topology graph. Whether a network link exists between two random nodes is determined by their distance, the values of α and β in the probability P that is calculated as follows [32].

$$P(u, v) = \alpha * \exp\left(-\frac{d}{\beta * d_{\max}}\right) \quad (26)$$

$P(u, v)$ is the probability that a connection of two nodes u, v is established, d is the Euclidian distance between u, v; the $d_{\max}$ is the max Euclidian distance between any two nodes. All SN and VN are directional.

2) Resource and demand generation

Alevin framework generates resources and demands randomly within a specified range of the minimum and maximum values. The load ρ is calculated based on the max resource and max demand [32] as follows.

$$D_{\max} = \rho * R_{\max} * \left(\frac{V}{k * V_k}\right) \quad (27)$$

$D_{\max}$ is the max demand value, $R_{\max}$ is the max resource value, V is the number of substrate nodes in the SN, k is the number of VNs in a VNR, which is specified as the layers in the range [4-6], and V_k is the number of virtual nodes in each VN; it is assumed that V_k is the same in all VNs in a VNR. To examine the effects of the load on the CEVNE NoM performance, different values of ρ (from light to heavy load) are used in different test cases.

3) The congestion ratio parameter

The evaluation networks are varied randomly to ensure the fairness and completeness of the evaluation, so R is calculated using a test network provided in [33]. It is a dense network (6 nodes and 12 links) with α_p and β_p are set to the value: $\sum_q t_{pq} = \alpha_p; \sum_p t_{pq} = \beta_q$; the traffic demand matrix (on the left) and the traffic capacity matrix (on the right) are as follows.

0	35	35	35	35	35		0	100	100	100	0	100
35	0	35	35	35	35		100	0	100	100	100	100
35	35	0	35	35	35		100	100	0	0	100	0
35	35	35	0	35	35		100	100	0	0	100	100
35	35	35	35	0	35		0	100	100	100	0	100
35	35	35	35	35	0		0	100	0	100	100	0

The result of the optimal routing ratio in the above traffic and capacity matrices is 0.875; it is used as a benchmark to estimate the value of R such that $0 \leq R \leq 1$. The selected value R is close to its upper bound so it does not affect the acceptance ratio. We select R=0.945 so it does not violate the bandwidth capacity of the edge; as such, our test results are non-biased. As R is the optimal value of the congestion ratio, we ensure it is never the case that $R > 1$.

4) The simulation environment and scenario generators

Alevin framework is used to generate test scenarios. Each test scenario includes the SN parameters, VN parameters, resource and demand parameters, and the node and link mapping algorithms. Each test scenario is designed as one experiment in Alevin framework with 9 different test cases (three seed values and three layer values).

TABLE III
THREE TOPOLOGIES OF THE 50-NODE NETWORK

	Substrate network		Virtual networks			
50-node	α, β	Links	α, β	Nodes	Layers	Links
Simple (S)	0.5 0.5	1000-1100	0.5 0.5	9	4-6	30-40
Moderate (M)	0.75 0.75	1300-1400	0.75 0.75	9	4-6	50-60
Complex (L)	1.0 0.5	1500-1600	1.0 0.5	9	4-6	65-75

TABLE IV
THREE TOPOLOGIES OF THE 25-NODE NETWORK

	Substrate network		Virtual networks			
25-node	α, β	Links	α, β	Nodes	Layers	Links
Simple (S)	0.5 0.5	245-275	0.5 0.5	5	4-6	5-9
Moderate (M)	0.75 0.75	320-340	0.75 0.75	5	4-6	10-14
Complex (L)	1.0 0.5	420-450	1.0 0.5	5	4-6	16-20

TABLE V
RESOURCES AND DEMANDS

	Substrate network Resources		Virtual networks Demands	
	CPU	BW	CPU	BW
	min/max	min/max	min/max	min/max
Simple (S)	10 / 20	60 / 70	1 / 5	**5 / 10**
Moderate (M)	10 / 20	60 / 70	1 / 5	**10 / 15**
Overloaded (L)	10 / 20	60 / 70	1 / 5	**15 / 20**

For the evaluation, two SNs are set up, a large 50 nodes, and a small 25 nodes. Three types of topologies are designed for each SN based on values of α and β: simple (S), moderate (M) and complex (L); they are applied for large and small SNs. Both SNs and VNs use the same values of α and β. Table III and TABLE IV present the parameters of the 50-node network and the 25-node network and their VNs.

Three types of load are designed based on virtual links' bandwidth demands: light load (S), medium load (M) and overloaded (L). The rest of the parameters, CPU resources and CPU demands are the same for all load types. Table V presents each type of load based on resources and demands; the unit of CPU resource is GHz, the unit of bandwidth resource is Gbps.

B. Challenging and near-congestion experiments

1) The design of challenging scenarios

The network congestion is caused by node congestion, link congestion and both node congestion and link congestion. Node congestion occurs where the node demand is close to the node capacity. Similarly, link congestion occurs when the link demand is close to the link capacity. In VNE, network congestion is caused by incoming VNRs with virtual node demands, virtual link demands and virtual network topologies. Therefore, the challenging and near congestion experiments are designed so that a node congestion can occur:

total CPU demand=total CPU resource

a link congestion can occur:

total bandwidth demand = total bandwidth resource

and in both node and link congestion with the VN topology having many links, the SN topology less links. Hence,

TABLE VI
NEAR CONGESTION EXPERIMENT SETUP

Substrate network: 25 nodes, $\alpha = 0.5, \beta = 0.3$ →160-170 links; β =0.6 → 270-280 links, directional
Virtual networks: 7 nodes, 5 layers, $\alpha = .75$, β in [0.70-0.85] → [28-34] links/VN, directional
Substrate node: CPU resource in [10-20] GHz
Virtual node: CPU demand in [5-20] GHz
Substrate link: bandwidth resource in [60-70] Gbps
Virtual link: bandwidth demand in [20-35] Gbps
Mapping algorithms:
Node mapping: CEVNE NoM algorithm; **link mapping**: k-shortest path
Node mapping: D-ViNE-LB algorithm; **link mapping**: k-shortest path

challenging and near-congestion experiments are designed with VNRs which are very difficult to be embedded successfully onto the SN.

2) Approximate resource scarcity (ARS) concept

ARS is introduced as a quantitative measurement to show how close is a VN demand compared to a SN resource [34]. ARS consists of node scarcity and link scarcity as follows.

For a mapping $(M, (N_i)_{i=1,..,n})$ with each VN: $N_i = (V_i, E_i, w)$: V_i are virtual nodes, E_i are virtual links, w is the demand; and SN: $M = (U, F, w)$, U are substrate nodes, F are substrate links, and w is the resource. ARS of an experiment is defined by a vector (x, y) that are the ratios of total demands over total resources: x is node scarcity, y is link scarcity; (x,y) is presented as the coordination in the scarcity map [34]:

$$x = \sum_{i=1}^{n} \frac{w(V_i)}{w(U)} \qquad y = \sum_{i=1}^{n} \frac{w(E_i)}{w(F)} \tag{28}$$

3) Network parameters in challenging scenarios

We transfer the ARS into parameters in the Alevin framework to set up near-congestion experiments for both algorithms CEVNE and D-ViNE-LB [11]. In TABLE VI, challenging and near-congestion experiments are set up on 25-node networks, the VN topologies are set up with $\alpha = 0.75$, β in [0.70 - 0.85]. The algorithms are CEVNE NoM and k-shortest path, and D-ViNE-LB and k-shortest path. Two sub-categories are identified in near-congestion scenarios:

(i) Most-scarce resource (MSR) category with a simple-topology SN $\alpha = 0.5$ and $\beta = 0.3$; a greedy-topology VN $\alpha = 0.75$; $\beta = 0.70, 0.75$;

(ii) Less-scarce resource (LSR) with average-complexity SN $\alpha = 0.5$ and $\beta = 0.6$, greedy and complex-topology VN $\alpha = 0.75$ and $\beta = 0.80; 0.85$. Although VNs are more complex than the MSR, the SN topology is also more complex: we must increase β from 0.3 to 0.6 in the SN, otherwise, it is too hard to complete successfully any VNRs.

4) Evaluation results of challenging scenarios

Experiments in the MSR and LSR categories are tested repeatedly in the pre-defined time window, and only successful ones, which have acceptance ratio > 0, are recorded; their ARSs are shown in the ARS map [34] for both algorithms, CEVNE and D-ViNE-LB.

In the test results, CEVNE successful embeddings in each category are always double to three times D-ViNE-LB successful embeddings. The number of CEVNE successful embeddings in the MSR category is about 60% of the total number of CEVNE successful embeddings in LSR category. This holds true for D-ViNE-LB as well although its number of

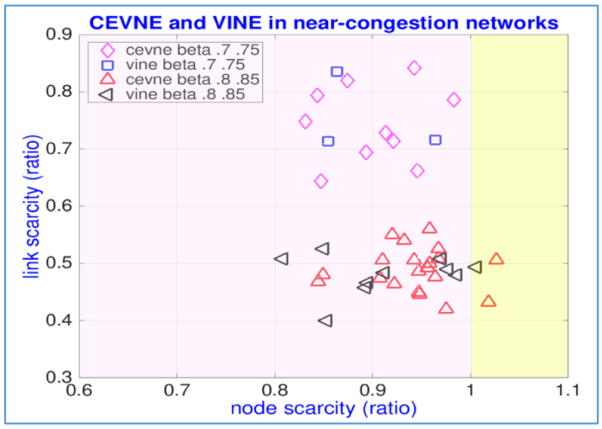

Fig. 1: Node and link scarcity of near-congestion experiments [34]

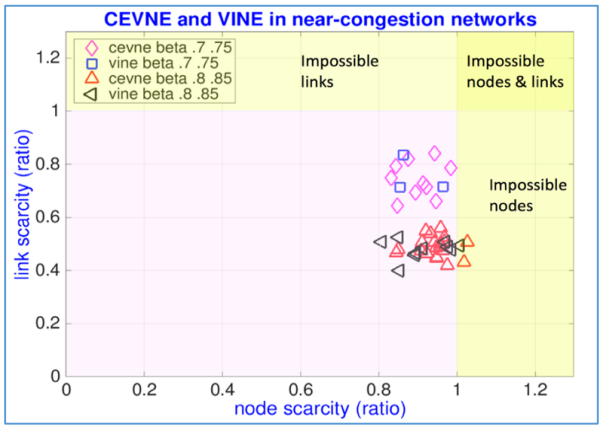

Fig. 2: Displaying the experiment results in ARS map [34]

successful embeddings is far less than CEVNE. The results also show the effects on VNE of the network topology and the random allocation of resource and demands that can only be measured on a case-by-case basis.

The results of experiments for two categories and their algorithms are shown in Fig. 1 with each experiment's ARS vector (x,y) calculated as in (27). Fig. 2 displays the results in the whole ARS map for a global view.

(i)The MSR embeddings (including blue and magenta symbols) have their node scarcity in the range 0.8 to 1, and the link scarcity in the range 0.3 to 0.45. SN is directional so the actual link scarcity is from 0.6 to 0.9. This is very close with the "hard" area in [34].

(ii) The LSR embeddings (including red and black symbols) have their node scarcity in the range 0.8 to 1.05 and link scarcity in the range 0.2 to 0.28; as the SN is directional, it is actually from 0.4 to 0.56, which is matched with the survey that D-ViNE-LB link scarcity is 0.4-0.5 in [34].

The result showed CEVNE NoM algorithm outperforms D-ViNE-LB algorithm in nearly-congestion, scarce-resource networks.

C. *The node mapping evaluation results*

CEVNE NoM algorithm is evaluated against D-ViNE-LB algorithm by running experiments, recording and analysing results in metrics. As both are node-mapping algorithms, a common link-mapping algorithm is used for the evaluation purposes. The Eppstein algorithm [35] for k-shortest path is selected as it matches with the testing scenario of 50-node or less SNs.

Results of the evaluation are collected in metrics that are related to the CEVNE objectives as follows.

(i) the runtime and cost/ revenue metrics are used to justify the cost saving objective

(ii) the average active-node stress metric is used to justify the congestion avoidance objective

(iii) the running nodes metric is used to justify the energy saving objective; and acceptance ratio is used for overall performance.

Each metric is collected for both 50-node and 25-node SNs, three types of loads S, M and L. The result of a test scenario is presented in a boxplot. In the boxplot, the central mark is the median, two top edges are the 25th and 75th percentiles, outliers are displayed separately. Each boxplot is named by its topology (the first letter) and load (the second letter). For example, LL specifies the large topology (50 node) and overloaded.

1) Runtime

The runtime results in Fig. 3 show that CEVNE NoM converged faster than D-ViNE-LB algorithm: CEVNE's total runtime is improved by 50% compared to D-ViNE-LB's. This is explained using the Simplex method: CEVNE introduces more explicit constraints, so the search space is more confined and hence it found solutions quicker. The 50-node network results are shown in high demands scenarios (moderate and overloaded workload).

2) Average active node stress

The average active node stress is calculated by dividing the total number of embedded virtual nodes by the number of active substrate nodes. CEVNE objective function puts a limit on the bandwidth usage to avoid congestion: virtual flows will only use R value multiplied by the max bandwidth; this implies that virtual node mappings are arranged tightly for virtual flows to meet the constraints on the limit bandwidth resources. Therefore, CEVNE average active node stress will be higher as in Fig. 4 that shows the results of 50-node network with high demands.

3) Energy consumption

The running nodes metric is used to evaluate the energy consumption of CEVNE against D-ViNE-LB: the algorithm with less running nodes will consume less energy. The test results in Fig. 5 show that the number of running nodes in CEVNE is always less than in D-ViNE-LB. In 50-node network, CEVNE always has 10 running nodes less than D-ViNE-LB; this means CEVNE consumes 20% less energy. Fig. 5 shows the results in 50-node networks with high demand.

4) Acceptance ratio

The acceptance ratio is the ratio of virtual networks that have been embedded successfully on the substrate network. The acceptance ratios of both algorithms are very similar in both 50-node and 25-node networks.

5) Total costs

The total cost is calculated based on the substrate resources that are used for the VNE; it does not include the energy cost. The total costs of both algorithms are very similar in both 50-node and 25-node networks. Through the analysis of the above metrics: runtime, average active node stress, running nodes, total cost and acceptance ratio, CEVNE NoM prevails over D-ViNE-LB in all three objectives: congestion-aware,

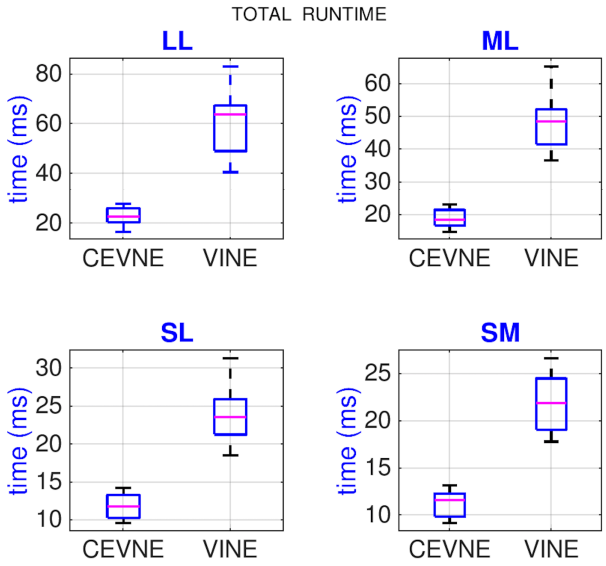

Fig. 3: Total runtime of 50-node network

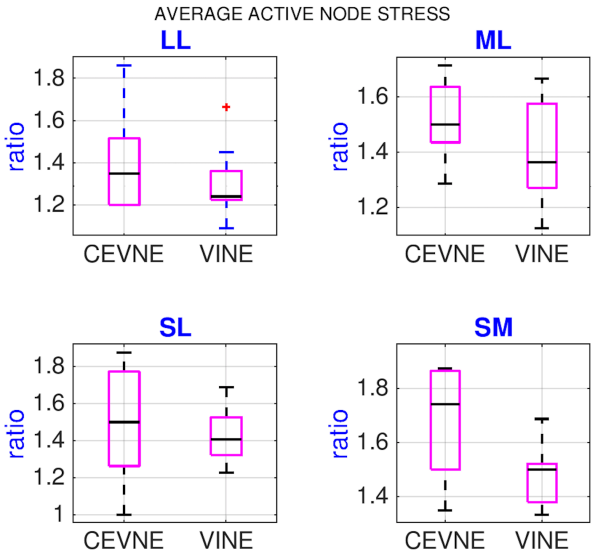

Fig. 4: Average active node stress of 50-node network

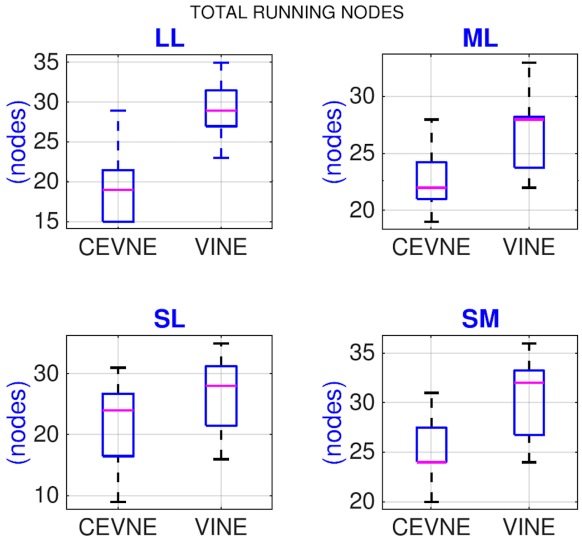

Fig. 5: Running nodes in 50-node network

energy-aware and cost saving. The next section describes the evaluation of CEVNE LiM.

VI. CEVNE LINK MAPPING AND EVALUATION RESULTS

We realize the CEVNE LiM algorithm as a composite SDN application using micro-services technology on an SDN substrate network, deploy it on mininet network using open virtual switches. CEVNE LiM has the central view and central control of the network from the SDN controller; it can steer the network flows for traffic engineering purposes and it can select the substrate paths that meet its three objectives. Mininet is used as the test environment because it is designed to test SDN networks.

A. The Realization of CEVNE Link Mapping Algorithm

CEVNE LiM is realised as an SDN application called CEVNE link mapping application (CEVNE LiM App) running on ONOS controller deploying Segment Routing (SR) application. CEVNE LiM App is a composite SDN application in a three-tier architecture [36]. In the business tier, CEVNE LiM App creates its new link-mapping service and composes ONOS services, SR application. CEVNE LiM App applies built-in micro-service patterns in ONOS controller: self-registration, service discovery, and service lookup to register new services, and to discover existing services.

The SDN substrate network is configured into a leaf-spine fabric that also runs SR protocol [37]. In ONOS controller, the SR application provides such configuration function. Each switch has its segment ID. The SR application provides the routing service for the leaf-spine fabric via its equal-cost multi-path (ECMP) shortest path graph (SPG) [38]. The SR application's ECMP SPG also plays the role of the Path services.

The main function of CEVNE LiM App is to implement the path selection strategy to select substrate paths satisfying three objectives: congestion aware, energy aware and cost saving, to embed virtual links. CEVNE LiM App saves costs via using the ECMP SPG to get the shortest paths. CEVNE LiM App uses the In-band network telemetry (INT) [39], or the Openflow statistic functions to monitor network traffic to select the least traffic path among resulting paths. It saves energy via setting one of the spines into inactive mode. It controls network congestion via spreading the virtual link mappings onto all different spines in the fabric.

B. The Link Mapping Evaluation Results

The performance of the CEVNE LiM application in ONOS controller is compared with the k-shortest path algorithm in the Alevin framework. Different link-mapping test scenarios are designed to evaluate CEVNE LiM App. The test results are collected in two separate environments: mininet and Alevin framework. The CEVNE LiM application testbed is a leaf-spine fabric with four spine switches, four leaf switches and eight hosts. These are running in mininet environment using open vswitch. The result of k-shortest path algorithm is calculated in the Alevin framework. The performances of CEVNE LiM and the k-shortest path algorithm are evaluated in terms of their

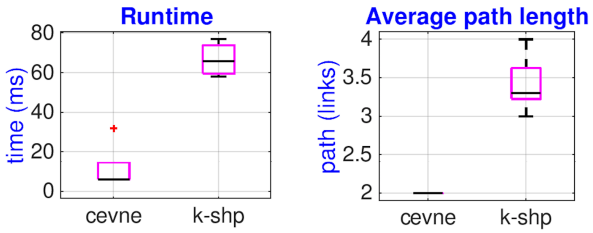

Fig. 6: The result of runtime metric and average path length metric

runtime, average path length, average node stress, average link stress, energy consumption and acceptance ratio. Each result is analysed as follows.

1) Runtime

In CEVNE LiM App, the runtime is measured in the application itself. The runtime results of both algorithms in Fig. 6 (left) show that CEVNE LiM converged faster than the k-shortest path algorithm. This is explained by the path selection algorithm of the CEVNE LiM App, and ECMP SPGs providing in-memory routing rules for the fabric.

2) Average path length

In CEVNE LiM App, the average path length is measured using the ONOS UI based on the created intent to count the number of links involving in the path. The average path length results in Fig. 6 (right) shows that CEVNE LiM approach always has shorter substrate path lengths compared to the k-shortest path algorithm. This is the result of CEVNE LiM App using the ECMP SPGs to search for the shortest paths between two end-points. It also means that CEVNE LiM uses fewer active nodes along its paths, so it consumes less energy.

3) Average node stress and average link stress

The average node stress is calculated by dividing the total number of embedded virtual nodes by the total number of substrate nodes. The average link stress is calculated by dividing the total number of embedded virtual links by the total number of substrate links. The average node stress and average link stress results in Fig. 7 (upper) show that the CEVNE LiM algorithm achieves the congestion awareness objective by spreading the traffic over different nodes and links because it has higher node stress and higher link stress compared to the k-shortest path approach.

4) Energy consumption

In CEVNE LiM application, the energy consumption is calculated based on the active nodes in the link mapping process. In each substrate path, the total number of hop counts is the number of active nodes. As CEVNE LiM has a shorter average path length than the k-shortest path algorithm, CEVNE LiM has a fewer number of active nodes along the paths, resulting in less power consumption. The energy consumption results in Fig. 7 (lower) show that CEVNE LiM algorithm prevails over the k-shortest path in energy saving.

5) Acceptance ratio

In CEVNE LiM application, the acceptance ratio is calculated by dividing the number of successful VNRs by the total number of VNRs. The acceptance ratios of both algorithms are always at the 100% in all test scenarios. Both algorithms' results of the runtime, the average path length, the average node stress, the average link stress and the energy consumption show that CEVNE LiM algorithm on ONOS controller running SR application achieves all three objectives: cost saving, energy

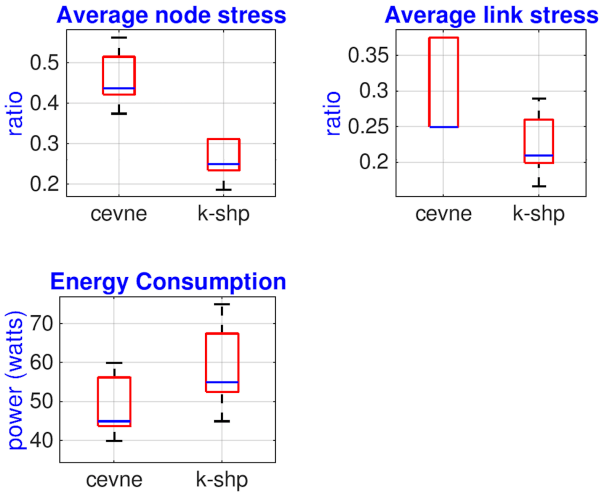

Fig. 7: The result of average node stress, average link stress and energy consumption metric

saving and congestion awareness, and prevails over the k-shortest path algorithm in these three objectives.

VII. Discussion and Future Works

A. Discussion

On the convergence of algorithms. The CEVNE mathematical model introduces multiple variables for multiple objectives. In the Simplex method, the extreme points are at the intersection of the objective space and its constraints. The CEVNE converges quicker because it introduces more constraints than D-ViNE-LB so the search space is narrower. The quick convergence of the formula results in the shorter runtime of CEVNE compared to D-ViNE-LB.

On CEVNE as a dynamic VNE.

CEVNE is a dynamic VNE that considers the changes in virtual networks and substrate networks during the VN lifetime. It reapplies node and link mapping algorithms for the reconfiguration process [3].

On scarce-resource near-congestion networks

The congestion control of CEVNE NoM brings significant benefits to the scarce-resource and near-congestion networks as described in section V.B. CEVNE LiM algorithm is beneficial for near-congestion networks because it takes advantage of SDN central control, central view and programmability to ignore congested paths and select the paths that are still available for traffic forwarding that is described in section VI.

On the costs of the link mapping process. The cost of the link mapping process is computed based on the cost of the SDN network, its core services, the monitoring application, the SR application and the path services. The cost of SDN network is comprised of the cost of TCAM in SDN switches. The exploiting of SR in the SDN network has reduced this cost significantly. The cost of SR includes the setup of the segment ID and the adjacency ID for switches and routers, the added function to process the segment ID, and the program to generate the list of labels in the packets' headers. This cost of SR is compensated if SR is built on top of an MPLS-enabled network, which is the case in most provider networks. The list of segment IDs can increase the packet header's size quickly. To overcome this issue, a maximum number of labels is set for the segment

list. This does not affect the routing of the packet because only the main label switched routers (LSR) along the path are put in the segment list, and ECMP decides the routes between these LSRs.

B. Future works

The integration of SDN and SR on the SDN network will benefit the network resilience objective. In the current CEVNE,

- The CEVNE LiM App will handle the splittable path mappings using the services provided by the controller
- The CEVNE LiM App can be extended to handle resilience if there are node or link failures along the original substrate path, and the quality of service (QoS) assurance via the composability of the services on top of the SDN controller.

VIII. CONCLUSION

The CEVNE is proposed with the purpose of minimizing network node and link costs, minimizing network congestion, and minimizing the total energy usage. Its substrate network is an SDN network equipped with SR, monitoring, and the SDN controller. It applies the optimal routing and the hose traffic demand model to set the upper limit for the congestion ratio. The CEVNE is a coordinated node and link-mapping algorithm. The CEVNE programming model applies the weighting methods on the cost saving and energy saving objectives with the base weights; and the constraint method on the congestion avoidance objective (Cohon [9] and Williams [29]) with the binding constraint so it will generate pareto-optimal solutions for all objectives. Its mathematical model is a MIP program. Solving MIP is computationally intractable, hence, it is relaxed to a LP and solved by the GLPK for sub-optimal solutions. The CEVNE node-mapping algorithm applies the rounding function in D-ViNE-LB to find the approximation values of integer variables. The CEVNE LiM is realized using SDN central control and programmability and SR. CEVNE LiM is a composite application that applies the path service and the monitoring application. The evaluation on near-congestion/scarce-resource networks shows that the CEVNE's congestion control objective is beneficial: its successful embeddings are double to triple D-ViNE-LB's successful embeddings. The overall evaluation of both the node mapping and the link mapping processes show that the CEVNE approach succeeds in delivering concurrently the optimal-cost, the congestion-aware and the energy-aware objectives. The evaluation shows that CEVNE NoM improves some performance metric by 50% in a 50-node network compared to D-ViNE-LB. It reduces the total energy consumption and the number of active nodes. The technologies that enable CEVNE play a very important role in fulfilling the traffic engineering task in the CEVNE LiM algorithm. This allows CEVNE to be dynamically extended to schedule virtual resources from one time window to the next. CEVNE LiM can also be extended to consider the splittable path mappings, the resilience objective, and the QoS objective in SDN networks that facilitate the application composition, the programmability, and the centralized control. Overall, CEVNE offers a realistic and cost-effective approach in data centers.

REFERENCES

[1] D. Schwerdel, D. Günther, R. Henjes, B. Reuther, and P. Müller, "German-lab experimental facility," presented at the Future Internet Symposium, Berlin, Heidelberg, 2010.

[2] J. Carapinha and J. Jiménez, "Network virtualization: a view from the bottom," presented at the Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures, 2009.

[3] A. Fischer, J. F. Botero, M. T. Beck, and H. De Meer, "Virtual network embedding: A survey," *IEEE Communications Surveys & Tutorials,* vol. 15, no. 4, p. 19, 2013.

[4] J. Elias, F. Martignon, S. Paris, and J. Wang, "Optimization models for congestion mitigation in virtual networks," presented at the IEEE 22nd International Conference on Network Protocols, 2014.

[5] Z. Zhang, S. Su, Y. Lin, X. Cheng, K. Shuang, and P. Xu, "Adaptive multi-objective artificial immune system based virtual network embedding," *Journal of Network and Computer Applications,* vol. 53, p. 16, 2015.

[6] I. Houidi, W. Louati, and D. Zeghlache, "Exact multi-objective virtual network embedding in cloud environments," *The Computer Journal,* vol. 58, no. 3, p. 13, 2015.

[7] E. Amaldi, S. Coniglio, A. M. Koster, and M. Tieves, "On the computational complexity of the virtual network embedding problem," *Electronic Notes in Discrete Mathematics,* vol. 52, p. 8, 2016.

[8] M. Rost and S. Schmid, "Charting the complexity landscape of virtual network embeddings," presented at the 2018 IFIP Networking Conference (IFIP Networking) and Workshops, 2018.

[9] J. L. Cohon, *MultiObjective Programming and planning* (Mathemtics in Science and Engineering). Courier Corporation, 1978.

[10] Y. Zhu and M. H. Ammar, "Algorithms for Assigning Substrate Network Resources to Virtual Network Components," *INFOCOM,* vol. 1200, no. 2006, p. 12, 2006.

[11] M. Chowdhury, M. R. Rahman, and R. Boutaba, "Vineyard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Transactions on Networking (TON),* vol. 20, no. 1, p. 14, 2012.

[12] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *ACM SIGCOMM Computer Communication Review,* vol. 38, no. 2, p. 13, 2008.

[13] X. Cheng *et al.*, "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Computer Communication Review,* vol. 41, no. 2, p. 10, 2011.

[14] A. Fischer, M. T. Beck, and H. De Meer, "An approach to energy-efficient virtual network embeddings," in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, 2013, p. 6.

[15] B. Özbek, Y. Aydoğmuş, A. Ulaş, B. Gorkemli, and K. Ulusoy, "Energy aware routing and traffic management for software defined networks," presented at the 2016 IEEE NetSoft Conference and Workshops (NetSoft), Seoul, Korea, 2016.

[16] J. Elias, F. Martignon, S. Paris, and J. Wang, "Efficient orchestration mechanisms for congestion mitigation in NFV: Models and algorithms," *IEEE Transactions on Services Computing,* vol. 10, no. 4, p. 13, 2017.

[17] M. Haghani, B. Bakhshi, and A. Capone, "Multi-objective embedding of software-defined virtual networks," *Computer communications,* vol. 129, no. 2018, p. 11, 2018.

[18] Z. Li, Z. Lu, S. Deng, and X. Gao, "A Self-Adaptive Virtual Network Embedding Algorithm Based on Software-Defined Networks," *IEEE Transactions on Network and Service Management,* vol. 16, no. 1, p. 12, 2018.

[19] L. R. Bays, L. P. Gaspary, R. Ahmed, and R. Boutaba, "Virtual Network Embedding in Software-Defined Networks," presented at the NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium, 2016.

[20] A. Blenk and W. Kellerer, "Towards Virtualization of Software-Defined Networks: A Journey in Three Acts," presented at the 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2019.

[21] M. Dahir, H. Alizadeh, and D. Gözüpek, "Energy efficient virtual network embedding for federated software-defined networks," *International Journal of Communication Systems,* vol. 32, no. 6, p. 17, 2019.

[22] S. Rout, S. S. Patra, and B. Sahoo, "Saving energy and improving performance in SDN using rate adaptation technique," *International Journal of Engineering & Technology,* vol. 7, no. 2.6, p. 6, 2018.

[23] R. Mijumbi, J. Serrat, J. Rubio-Loyola, N. Bouten, F. De Turck, and S. Latre, "Dynamic resource management in SDN-based virtualized

networks," in *10th international conference on network and service management (CNSM) and workshop*, 2014, p. 6.
[24] E. Rodriguez, G. P. Alkmim, N. L. da Fonseca, and D. M. Batista, "Energy-Aware Mapping and Live Migration of Virtual Networks," *IEEE Systems Journal,* vol. 11, no. 2, p. 12, 2015.
[25] P. Reviriego *et al.*, "An Energy Consumption Model for Energy Efficient Ethernet Switches," presented at the 2012 International Conference on High Performance Computing & Simulation (HPCS), 2012.
[26] B. Fortz and M. Thorup, "Internet Traffic Engineering by optimizing OSPF weights," in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, 2000, vol. 2, p. 10, IEEE.
[27] D. Medhi, *Network routing: algorithms, protocols, and architectures (edition 2)*, second ed. Morgan Kaufmann., 2017.
[28] T. O. M. Y. M. J. Ng and D. Hoang, "Joint optimization of capacity and flow assignment in a packet-switched communications network," *IEEE Transactions on Communications,* vol. 35, no. 2, p. 8, 1987.
[29] H. P. Williams, *Model building in mathematical programming*. John Wiley & Sons., 2013.
[30] M. C. Lee and J. P. Sheu, "An efficient routing algorithm based on segment routing in software-defined networking," *Computer Networks,* no. 103, p. 12, 2016.
[31] M. T. Beck, C. Linnhoff-Popien, A. Fischer, F. Kokot, and H. de Meer, "A simulation framework for virtual network embedding algorithms," presented at the 2014 16th international telecommunications network strategy and planning symposium (Networks) 2014.
[32] X. Hesselbach, J. R. Amazonas, S. Villanueva, and J. F. Botero, "Coordinated node and link mapping VNE using a new paths algebra strategy," *Journal of Network and Computer Applications,* vol. 69, p. 13, 2016.
[33] E. Oki, *Linear programming and algorithms for communication networks: a practical guide to network design, control, and management.* CRC Press., 2012
[34] A. Fischer, "AN EVALUATION METHODOLOGY FOR VIRTUAL NETWORK EMBEDDING," PhD, University Passau, Germany, 2016.
[35] D. Eppstein, "Finding the k shortest paths," *SIAM Journal on computing,* vol. 28, no. 2, p. 22, 1998.
[36] M. Pham and D. B. Hoang, "SDN applications-The intent-based Northbound Interface realisation for extended applications," presented at the 2016 IEEE NetSoft Conference and Workshops (NetSoft), Seoul, Korea, 2016.
[37] S. Das. (2019). *Leaf-spine fabric configuration*. Available: https://github.com/opennetworkinglab/routing
[38] ONOS. (2019, 6 October). *Segment Routing*. Available: https://github.com/opennetworkinglab/onos/tree/master/apps/segmentrouting
[39] C. Kim *et al.* (2016, July 5th). *Inband network telemetry (INT) specification*. Available: http://p4.org/wp-content/uploads/fixed/INT/INT-current-spec.pdf

Minh Pham received a Masters of Applied Science, IT at RMIT University in Melbourne, Australia in 1997, MBA with major in IT at UTS, Sydney, Australia in 2007. She has more than 15 year-experience in Australia in software development in different sectors: finance, government, and tele-communication. She worked as Senior Software Developer, Senior Technical Specialist, and Consultant in IT industries. She is currently working on her PhD in the Faculty of Engineering and IT at UTS Sydney. Her current research interests include Software-defined Networking (SDN), optimization, network slicing in 5G networks, and Cloud Computing.

Doan B. Hoang is a Professor in the School of Electrical and Data Engineering, Faculty of Engineering and Information Technology, University of Technology Sydney (UTS). He leads the Virtualized Infrastructures and Cyber Security (VICS) research group and is a Research Associate of the Open Networking Foundation (ONF). His current research interests include: Software-defined infrastructures (Cloud, SDN and NFV) and services, Security capability maturity models and metrics for Cloud, Trust assessment modelling, and IoTs for Assistive Healthcare. Professor Hoang has published over 200 research papers. He was the Co-director of UTS iNEXT research center (2007-2017) and the Head of the School of Computing and Communications (2011-2015). Before UTS, he was with Basser Department of Computer Science, University of Sydney. He held various visiting positions: Visiting Professorships at the University of California, Berkeley; Nortel Networks Technology Centre in Santa Clara, the University of Waterloo, Carlos III University of Madrid, and Nanyang Technological University, Lund University, POSTECH University, South Korea

Zenon Chaczko is a Senior Lecturer and the Director of ICT Programs at Faculty of Engineering and Information Technology, the University of Technology, Sydney (UTS). He is the core member of GBDTC - Global Big Data Technologies Canter. Dr Chaczko's research interests include Artificial Intelligence, Software Systems Engineering, Cloud and Ambient Computing (Wireless Sensor/Actuators Networks, Internet of Things (IoT), Body Area Networks), Augmented Reality and Complex Software Systems (Laparoscopic Simulation, Early Detection of Environmental/Medical Anomalies, Morphotronics). Dr Zenon Chaczko received his Bachelor in Computer Science (Honours, 1st class) and his PhD in Computer and Software Engineering. Dr Chaczko has been actively conducting research in his fields and has published numerous (~250) research papers. Before UTS, he was full-time working for over 20 years in the software and system engineering industry. He held various visiting positions: Visiting Professorships at the University of Arizona, Tucson, USA; IPN in Mexico City, Mexico; the University of Las Palmas, Canary Islands, Spain; Wroclaw University of Technology, Poland; and University of Applied Science and Engineering, Hagenberg/Linz, Austria. He was a keynote speaker at multiple conferences.