Novel Location-Constrained Virtual Network Embedding (LC-VNE) Algorithms Towards Integrated Node and Link Mapping

Long Gong, *Student Member, IEEE*, Huihui Jiang, Yixiang Wang, and Zuqing Zhu, *Senior Member, IEEE*

***Abstract*— This paper tries to solve the location-constrained virtual network embedding (LC-VNE) problem efficiently. We first investigate the complexity of LC-VNE, and by leveraging the graph bisection problem, we provide the first formal proof of the $\mathcal{NP}$-completeness and inapproximability result of LC-VNE. Then, we propose two novel LC-VNE algorithms based on a compatibility graph (CG) to achieve integrated node and link mapping. In particular, in the CG, each node represents a candidate substrate path for a virtual link, and each link indicates the compatible relation between its two endnodes. Our theoretical analysis proves that the maximal clique in the CG is also the maximum one when the substrate network has sufficient resources. With CG, we reduce LC-VNE to the minimum-cost maximum clique problem, which inspires us to propose two efficient LC-VNE heuristics. Extensive numerical simulations demonstrate that compared with the existing ones, our proposed LC-VNE algorithms have significantly reduced time complexity and can provide smaller gaps to the optimal solutions, lower blocking probabilities, and higher time-average revenue as well.**

***Index Terms*— Network virtualization, maximum clique, location-constrained virtual network embedding (LC-VNE), compatibility graph (CG), inapproximability.**

I. Introduction

SINCE its inception, Internet has been growing rapidly with more network elements and end-users, larger volume of traffic, and more various applications. Due to the huge scale and coexistence of numerous service providers, Internet infrastructure has recently been brought into "ossification" [1], which means that it becomes more and more difficult to support new networking mechanisms and applications timely. Especially, recent advances on Big Data applications [2], [3] made this trend more explicit. It is known that network virtualization allows heterogeneous virtual networks to coexist in the same substrate/physical network and share the resources (*e.g.*, CPU, storage, and bandwidth) efficiently [4]. Therefore, it has been considered as a promising solution to the Internet ossification, and has attracted intensive attentions from both academia and industry [4]–[7]. With network virtualization, the role of Internet service providers (ISPs) evolves into two entities: infrastructure provider (InP) and service provider (SP). InPs own substrate/physical networks (*i.e.*, network infrastructure), while SPs lease substrate resources from InPs to build virtual networks for end-users.

One of the vital challenges in network virtualization is how to allocate resources in a substrate network to virtual network requests (VNRs) efficiently, which is also known as virtual network embedding (VNE) [8]. Specifically, VNE finds each virtual node (VN) a unique substrate node (SN) to meet its computing resource requirement and selects a set of substrate paths to satisfy the bandwidth requirement of each virtual link (VL). The former procedure is the so-called *node mapping*, while the latter is *link mapping*. VNE is an $\mathcal{NP}$-hard problem [9]. Previously, to solve it for different scenarios, researchers have developed several integer linear programming (ILP) and mixed ILP (MILP) models and a few heuristics [10]–[23].

Meanwhile, it is known that many emerging network virtualization applications may impose location constraints on VNs, similar to other location-aware applications [24]–[27]. For instance, a service provider of teleconference needs to apply the location constraints to VNs such that the user-perceived latency is short enough for guaranteed quality-of-experience (QoE). Nevertheless, the existing algorithms for generic VNE are not suitable for solving this type of location-constrained VNE (LC-VNE) problems [14]. Specifically, the introduction of location restrictions dramatically increases the difficulty on both the complexity analysis and algorithm design, especially when we want to achieve integrated node and link mapping. By formulating LC-VNE as an MILP, the authors of [14] proposed a relaxation-based method. Later, a column generation based approach was developed in [15]. However, both of these two approaches suffered from relatively high time complexity. The study in [28] also considered LC-VNE, but instead of using the generic network model in [14], it incorporated a specific node mapping model. To the best of our knowledge, there has been no comprehensive complexity analysis on LC-VNE, which makes how to solve LC-VNE efficiently a rather difficult problem.

In this work, we aim at designing efficient LC-VNE algorithms based on the generic network model in [14].

Manuscript received April 11, 2015; revised August 3, 2015; accepted February 21, 2016; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor P.-J. Wan. This work was supported in part by the NCET Project under Grant NCET-11-0884, the National Natural Science Foundation of China under Project 61371117, the Fundamental Research Funds for the Central Universities under Grant WK2100060010, the Natural Science Research Project for Universities in Anhui Province under Grant KJ2014ZD38, and the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant XDA06011202.

The authors are with the School of Information Science and Technology, University of Science and Technology of China, Hefei 230027, China (e-mail: gonglong@mail.ustc.edu.cn; hhjiang@mail.ustc.edu.cn; yx153054@mail.ustc.edu.cn; zqzhu@ieee.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2016.2533625

We first analyze the complexity of LC-VNE. Specifically, by leveraging graph bisection [29], we prove that the decision version of LC-VNE is $\mathcal{NP}$-complete and also provide the in-approximatibility result of it. These findings make us give up the efforts on looking for polynomial-time approximation proposals and switch to efficient heuristics. Hence, we propose two novel efficient LC-VNE algorithms for achieving integrated node and link mapping based on compatibility graph (CG). First of all, we develop the approach to build a CG, in which each node represents one candidate substrate path for a VL, while each link indicates the compatible relation between its two end-nodes. Secondly, our theoretical analysis proves that the CG has a very good property that the maximal clique in it is also the maximum one, when the substrate network satisfies certain conditions, which will be specified in *Theorem* 4 in Subsection V-B2. Finally, with CG, we reduce LC-VNE to the minimum-cost maximum clique problem [29] and propose two heuristics, namely, greedy LC-VNE based on CG (G-CG), and load-balance enhanced LC-VNE based on CG (LBE-CG). With extensive numerical simulations, we demonstrate that the proposed CG-based LC-VNE algorithms outperform existing ones by achieving smaller gaps to the optimal solutions and providing lower blocking probability and higher time-average revenue. Moreover, they also yield lower time complexity and supply better fairness among VNRs that have different sizes.

In summary, the contributions of this work are as follows.

- We provide the first formal proof of the $\mathcal{NP}$-completeness of LC-VNE and give its in-approximatibility result.
- We develop two novel LC-VNE algorithms based on CG to facilitate integrated node and link mapping, and prove the good property of CG through theoretical analysis.
- We conduct extensive simulations to demonstrate that the proposed algorithms achieve better performance and lower time complexity than the existing ones.

The rest of this paper is organized as follows. Section II summarizes the related work. In Section III, we provide the network models and problem description. Section IV analyzes the complexity of LC-VNE and gives its in-approximability result. Section V describes the CG-based LC-VNE algorithms, and their performances are evaluated in Section VI. Finally, Section VII summarizes the paper.

II. Related Work

As a major challenge of network virtualization, VNE has recently attracted intensive interests. For a complete review of the previous work on VNE, one is encouraged to read the two surveys in [8] and [30]. In [10], by defining the local resource of a node as its computing capacity multiplying the total bandwidth capacity of all its incident links, Yu *et al.* proposed a two-phase VNE algorithm that included greedy-based node mapping and link mapping using K-shortest paths. Later studies found that the performance of VNE can be improved by considering the topology information (*i.e.*, global resource) in node mapping [11]–[13], [21]. With the consideration of global resource in the whole substrate network, the work in [11], [13], and [21] adopted PageRank [31] inspired approaches for node mapping, while Wang *et al.* leveraged the metric of centrality [32] to achieve the similar goal [12]. However, these investigations did not consider LC-VNE, which is very important for many emerging network virtualization applications.

To solve LC-VNE, Chowdhury *et al.* developed an augmented graph based approach, used it to model LC-VNE as a multi-commodity flow (MCF) problem, and formulated an MILP for coordinated node and link mapping [14]. They also proposed heuristics based on techniques such as linear programming (LP) relaxation together with deterministic or random rounding to overcome the intractability of the MILP. In [17], Papagianni *et al.* extended this MILP model to consider multi-dimensional resource requirements on both VNs and VLs and addressed quality-of-service (QoS) issues. Nevertheless, it is known that LP is relatively time-consuming, even though it can be solved in polynomial time. Moreover, LP relaxation together with rounding may not always provide good solutions and can even lead to infeasible ones under certain circumstances [33]. In [15], Hu *et al.* formulated a path-based MILP model and utilized column generation to solve LC-VNE. However, time complexity is still an issue for this proposal. Note that, high time-efficiency of network control and management is essential for online operations [34]. Recently, focusing on LC-VNE in optical network virtualization, Zhao *et al.* proposed an ILP model and two heuristic algorithms in which they allowed multiple VNs to be embedded onto the same SN [28].

In this work, we develop two novel LC-VNE algorithms based on compatibility graph (CG). As we will show later, the proposed algorithms achieve integrated node and link mapping efficiently with significantly reduced time complexity.

TABLE I
Notations

Notation	Description
v^s	Substrate node (SN)
e^s	Substrate link (SL)
$c^s_{v^s}$	Computing capacity of SN v^s
$l^s_{v^s}$	Location of SN v^s
$b^s_{e^s}$	Bandwidth capacity of SL e^s
p^s	Substrate path
$P^s_{v^s}$	Set of loopless paths that use SN v^s as an end-node
$P^s_{v^s_1,v^s_2}$	Set of loopless paths connecting SNs v^s_1 and v^s_2
K	Maximum number of substrate paths between each SN-pair
v^r	Virtual node (VN)
e^r	Virtual link (VL)
$l^r_{v^r}$	Preferred location of VN v^r
$\Phi^s_{v^r}$	Candidate SNs for VN v^r
$P^s_{e^r}$	Candidate substrate paths for VL e^r
$b^{e^r}_{p^s}$	Bandwidth allocated to accommodate VL e^r on path p^s

III. Network Models and Problem Description

In this section, we describe the network models and define the LC-VNE problem. Here, Table I summarizes the main notations used in this paper.

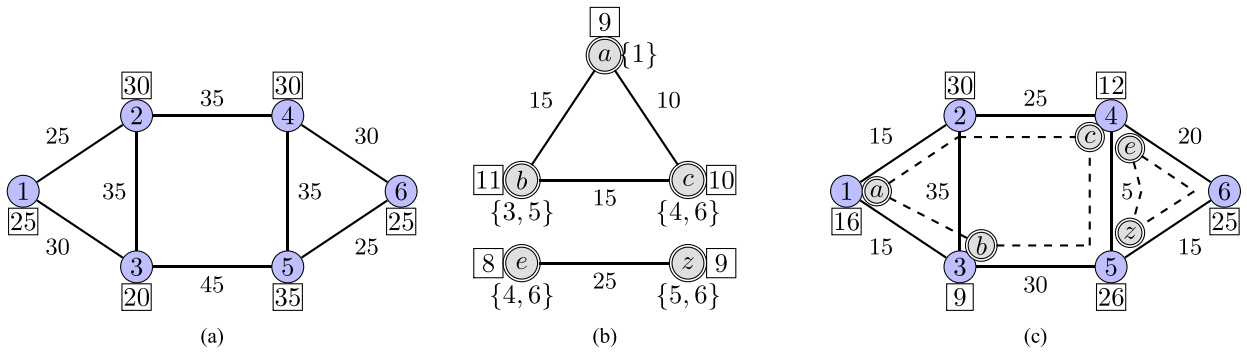


Fig. 1. An example of LC-VNE. (a) Substrate network. (b) VNRs. (c) LC-VNE solutions.

A. Network Models

1) Substrate Network: We model the substrate network as an undirected graph, denoted as $G^s(V^s, E^s)$, where V^s and E^s are the sets of substrate nodes (SNs) and substrate links (SLs), respectively. Each SN $v^s \in V^s$ has a computing capacity $c^s_{v^s}$, while the bandwidth capacity of each SL is $b^s_{e^s}$. Each SN v^s is also associated with a location $l^s_{v^s}$. The set of loopless paths in $G^s(V^s, E^s)$ is denoted as P^s, and the set of paths that start/end at node v^s is $P^s_{v^s}$. For two SNs $v^s_1, v^s_2 \in V^s$, we pre-calculate K shortest paths between them and denote the path set as $P^s_{v^s_1, v^s_2}$. Fig. 1(a) shows an example of substrate network. For simplicity, the location of each SN is omitted. The numbers around the SNs and on the SLs are their computing and bandwidth capacities, respectively.

2) Virtual Network: The virtual network request (VNR) is also modeled as an undirected graph, $G^r(V^r, E^r)$. The computing requirement of each virtual node (VN) $v^r \in V^r$ is $c^r_{v^r}$, while the bandwidth requirement of each virtual link (VL) $e^r \in E^r$ is denoted as $b^r_{e^r}$. For LC-VNE, we assume that each VN $v^r \in V^r$ has a preferred location, denoted as $l^r_{v^r}$. Based on $l^r_{v^r}$, v^r can only be mapped onto a set of candidate SN(s), $\Phi^s_{v^r} \subseteq V^s$, which is defined based on the preferred location $l^r_{v^r}$ and a radius ρ, *i.e.*, $\Phi^s_{v^r} = \{u^s \in V^s : ||l^s_{u^s} - l^r_{v^r}|| \leq \rho\}$. Here, $||l^s_{u^s} - l^r_{v^r}||$ is the distance between the two locations. Each VL $e^r \in E^r$ is also associated with a candidate substrate path set, $P^s_{e^r}$, which includes the substrate paths between the candidate SNs of the two end-nodes of e^r, *i.e.*, $P^s_{e^r} = \bigcup_{v^s_1 \in \Phi^s_{e^r_+}} \bigcup_{v^s_2 \in \Phi^s_{e^r_-}} P^s_{v^s_1, v^s_2}$, where e^r_+/e^r_- denotes the two end-nodes of e^r. Fig. 1(b) shows two VNRs. The numbers in the rectangles around the VNs denote their computing requirements, the numbers in the braces around the VNs indicate their candidate SNs, and the numbers on the VLs are their bandwidth requirements.

B. Problem Description of LC-VNE

Generic VNE provisions proper resources from the substrate network to support VNRs [10], while LC-VNE is a special scenario of it. In node mapping, we select a unique SN for each VN to satisfy its computing and location requirements, while in link mapping, we establish one or more substrate paths for each VL to meet its bandwidth requirement. Fig. 1(c) shows the LC-VNE results for embedding the VNRs in Fig. 1(b) onto the substrate network in Fig. 1(a). More specifically, the node mapping for the two VNRs are $\{a \to 1, b \to 3, c \to 4\}$ and $\{e \to 4, z \to 5\}$, respectively, while the link mapping relations are $\{(a,b) \to \{1-3\}, (a,c) \to \{1-2-4\}, (b,c) \to \{3-5-4\}\}$ and $\{(e,z) \to \{4-5, 4-6-5\}\}$. The objective of LC-VNE is to accommodate as many VNRs as possible, or to maximize the revenue of the InP, which is formally defined in Eqs. (11)-(12) in Subsection VI-A.

IV. Complexity of LC-VNE

In this section, we analyze the complexity of LC-VNE. Firstly, we transform a well-known $\mathcal{NP}$-complete problem, *i.e.*, the graph bisection problem (decision version) [29, p. 210, Problem ND17], into our LC-VNE problem, and prove that the decision version of LC-VNE is $\mathcal{NP}$-complete.[1] Then, we investigate the hardness of approximating LC-VNE, and prove that approximating LC-VNE within a factor of $m^{1-\epsilon}$ is also $\mathcal{NP}$-hard, for some $\epsilon > 0$, where m is the size of the substrate network, *i.e.*, the number of SNs. Before conducting the proof, we give the formal definitions of the problems.

Definition 1 [LC-VNE (Decision Version)]: Given the substrate network $G^s(V^s, E^s)$ with computing and bandwidth capacity $c^s_{v^s}$ and $b^s_{e^s}$ on each SN v^s and SL e^s, respectively, a location $l^s_{v^s}$ associated with each SN v^s, a VNR $G^r(V^r, E^r)$, $|V^r| \leq |V^s|$ with computing and bandwidth requirement $c^r_{v^r}$ and $b^r_{e^r}$ on each VN v^r and VL e^r, respectively, the location constraint $\Phi^s_{v^r} \subseteq V^s$ for each VN v^r, and a positive number Q, the problem is to ask whether there is an LC-VNE solution whose total resource consumption is no more than Q, *i.e.*,

$$\mathcal{R} = \sum_{v^r \in V^r} c^r_{v^r} + \sum_{e^r \in E^r} \sum_{p^s \in f_L(e^r)} |p^s| \cdot b^{e^r}_{p^s} \leq Q, \qquad (1)$$

where $f_L(\cdot)$ is the link mapping relation, $|p^s|$ denotes the hop-count of substrate path p^s, and $b^{e^r}_{p^s}$ is the bandwidth that is allocated on p^s for VL e^r. An LC-VNE solution has two related mappings: $f_N : V^r \to V^s$ and $f_L : E^r \to P^s$, and by allocating bandwidth $b^{e^r}_{p^s}$ on each embedded path p^s for VL e^r, it satisfies the following constraints.

[1] Although the $\mathcal{NP}$-hardness of VNE was proved in [9], to the best of our knowledge, there has been no formal proof of the $\mathcal{NP}$-hardness of LC-VNE.

- *Location Constraint:* each VN can only be embedded onto the SN that is in its candidate SN set, *i.e.,*

$$v^s \in \Phi^s_{v^r}, \quad \text{if } f_N(v^r) = v^s, \quad \forall v^r \in V^r. \tag{2}$$

Here, the location constraint may come from the functionality, security, or availability requirements of VNs [30].

- *One-to-One Node Mapping Constraint:* each VN in the same VNR can only be mapped onto a single SN and any two different VNs in a single VNR cannot be mapped onto the same SN, *i.e.*,

$$f_N(v^r_1) = f_N(v^r_2), \quad \textbf{if and only if } v^r_1 = v^r_2. \tag{3}$$

Note that one-to-many node mapping is not resource-efficient, because if we split a VN and embed it onto multiple SNs, additional bandwidth resources need to be allocated on the substrate paths among the SNs to support the VN's internal communication. On the other hand, many-to-one node mapping makes a VNR more vulnerable to substrate network failures, *e.g.*, a single SN failure can bring down multiple VNs [20].

- *Link Mapping Constraint*[2]*:* each VL should be mapped onto the path(s) connecting the SNs that its two end-nodes are embedded onto, *i.e.*,

$$f_L(e^r) \subseteq P^s_{f_N(e^r_+), f_N(e^r_-)}, \quad \forall e^r \in E^r, \tag{4}$$

where e^r_+ and e^r_- are the two end-nodes of VL e^r.

- *Computing Capacity Constraint:* the computing capacity of an embedded SN should satisfy the computing requirement of the corresponding VN, *i.e.*,

$$c^r_{v^r} \le c^s_{v^s}, \quad \text{if } f_N(v^r) = v^s, \quad \forall v^r \in V^r. \tag{5}$$

- *Bandwidth Capacity Constraint:* the allocated bandwidth on each SL should not exceed its bandwidth capacity, while satisfying the bandwidth requirements of VLs, *i.e.*,

$$b^s_{e^s} \ge \sum_{e^r \in E^r} \sum_{p^s \in f_L(e^r)} b^{e^r}_{p^s} \cdot I^{e^s}_{p^s}, \quad \forall e^s \in E^s, \tag{6a}$$

$$b^r_{e^r} = \sum_{p^s \in f_L(e^r)} b^{e^r}_{p^s}, \quad \forall e^r \in E^r, \tag{6b}$$

where $b^{e^r}_{p^s}$ is the amount of bandwidth that is allocated to accommodate VL e^r on path p^s, and $I^{e^s}_{p^s}$ is a known parameter, which indicates whether p^s traverses e^s or not, *i.e.*, $I^{e^s}_{p^s} = 1$ if p^s traverses e^s, otherwise, $I^{e^s}_{p^s} = 0$.

Definition 2 [Graph Bisection Problem (Decision Version)]: Given a graph $G(V, E)$ and an integer $K \in (0, |E|]$, the problem is to ask whether there exist two subsets V_1 and V_2 of V, satisfying that $V = V_1 \cup V_2$, $V_1 \cap V_2 = \emptyset$, $|V_1| = \lceil \frac{|V|}{2} \rceil$, $|V_2| = \lfloor \frac{|V|}{2} \rfloor$ and the number of links between them is no more than K, *i.e.*, $|\{(u, v) \in E : u \in V_1, v \in V_2\}| \le K$.

Theorem 1: The LC-VNE problem is $\mathcal{NP}$-complete.

[2]Note that the link mapping scheme described here is generic and allows path splitting, *i.e.*, one VL can be embedded onto one or more substrate paths. For a more specific scheme that only uses single-path mapping [21], $f_L(e^r)$ should only include one substrate path.

Proof: It is easy to verify that LC-VNE belongs to $\mathcal{NP}$ because given an LC-VNE solution, we can test whether it is the desired one[3] by checking Eqs. (1)-(6) in polynomial time.

We then transform the graph bisection problem into LC-VNE. Let $\{G(V, E), K\}$ be an **arbitrary** instance of graph bisection. We can build an instance of the LC-VNE problem, which includes a substrate network $G^s(V^s, E^s)$, a VNR $G^r(V^r, E^r)$ and a positive number Q. We will prove that a desired LC-VNE solution exists **if and only if** there exists a desired graph bisection for $\{G(V, E), K\}$.

We first design the polynomial-time transformation, in which VNR $G^r(V^r, E^r)$ consists of two main components:

1) **Component One** has the same structure as the instance of graph bisection $\{G(V, E), K\}$, *i.e.*,

$$V^r_1 = V, \quad E^r_1 = E.$$

Since they are adopted directly from G, we refer to the VNs as $\mathcal{O}$-nodes (*i.e.*, original nodes) and denote them as $V^r_1 = \{o^r_1, o^r_2, \ldots, o^r_{|V|}\}$. For each VN o^r_i, the corresponding node in G is denoted as o_i. Similarly, we call VLs in E^r_1 as $\mathcal{O}$-links (*i.e.*, original links).

2) **Component Two** is a set of $|V|$ VNs, which are named as $\mathcal{X}$-nodes (*i.e.*, extra nodes) and denoted as $\{x^r_i\}$, *i.e.*,

$$V^r_2 = \cup^{|V|}_{i=1}\{x^r_i\}, \quad E^r_2 = \emptyset.$$

There is a link between every pair of $\mathcal{O}$-node and $\mathcal{X}$-node, which is named as an $\mathcal{XO}$-link. The set of $\mathcal{XO}$-links are denoted as follows,

$$E^r_{1,2} = \{(o^r_i, x^r_j) : o^r_i \in V^r_1 \text{ and } x^r_j \in V^r_2\},$$

Each VN or VL in the VNR has a unit requirement on computing or bandwidth resource.

The substrate network has three components, which can be understood as two islands connected by a bridge.

1) **Components One and Two** are the two islands, which are referred to as $\mathcal{V}$-island and $\mathcal{U}$-island, respectively. Actually, they are two complete sub-graphs, each of which has $|V|$ SNs,

- $\mathcal{V}$-island,

$$V^s_1 = \bigcup_{i=1}^{|V|}\{v^s_i\}, \quad E^s_1 = \{(v^s_i, v^s_j) \in V^s_1 \times V^s_1 : i \ne j\}.$$

- $\mathcal{U}$-island,

$$V^s_2 = \bigcup_{i=1}^{|V|}\{u^s_i\}, \quad E^s_2 = \{(u^s_i, u^s_j) \in V^s_2 \times V^s_2 : i \ne j\},$$

where $X \times Y$ returns the Cartesian product [35] of sets X and Y, *i.e.*, $X \times Y = \{(x, y) : x \in X \text{ and } y \in Y\}$.

2) **Component Three** is the bridge that represents the connections between $\mathcal{V}$-island and $\mathcal{U}$-island. We call it as $\mathcal{VU}$-bridge. The ingress- and egress-ends of $\mathcal{VU}$-bridge are denoted as κ^s_1 and κ^s_2. All the SNs in $\mathcal{V}$- and $\mathcal{U}$-islands are connected to the ingress- and egress-ends, respectively. There are two paths in $\mathcal{VU}$-bridge, *i.e.*, one goes

[3]Note that, by saying a "desired" LC-VNE solution, we mean that the LC-VNE solution satisfies Eq. (1).

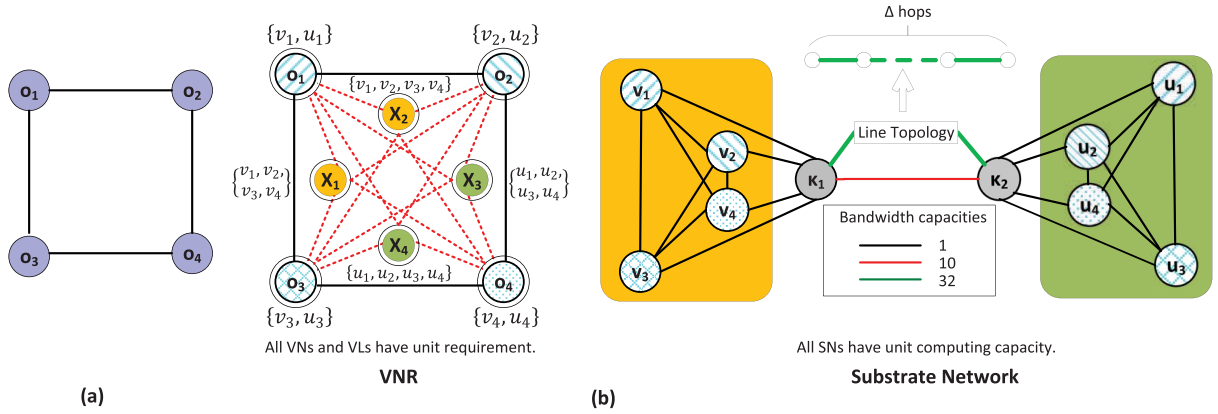


Fig. 2. Example on the transformation from (a) an instance of minimum graph bisection ($K = 2$) to (b) the instance of LC-VNE ($Q = 48$).

through the link connecting κ_1^s and κ_2^s with a capacity of $K + \left(\left\lfloor \frac{|V|}{2} \right\rfloor\right)^2 + \left(\left\lceil \frac{|V|}{2} \right\rceil\right)^2$, and the other traverses a line topology with Δ hops, on which the capacity of each SL is $2|V|^2$. The mathematical descriptions of all the SNs and SLs in $\mathcal{VU}$-bridge are as follows.

$$\begin{aligned} V_3^s &= \bigcup_{i=1}^{\Delta+1} \{\iota_i^s\} \cup \{\kappa_1^s, \kappa_2^s\}, \\ E_{3^1}^s &= \{(v_i^s, \kappa_1^s) : v_i^s \in V_1^s\} \cup \{(u_i^s, \kappa_2^s) : u_i^s \in V_2^s\}, \\ E_{3^2}^s &= \{(\kappa_1^s, \iota_1^s), (\iota_{\Delta+1}^s, \kappa_2^s)\} \cup \bigcup_{i=1}^{\Delta} \{(\iota_i^s, \iota_{i+1}^s)\}, \\ E_{3^3}^s &= \{(\kappa_1^s, \kappa_2^s)\}, \end{aligned}$$

where $\iota_i^s, i = 1, 2, \ldots, \Delta + 1$ are the SNs on the Δ-hop line topology, where $\Delta > 1$ is a constant number.

All the SNs in the substrate network have a unit capacity of computing resource, and all the SLs (except for those being explicitly described) also have a unit capacity of bandwidth.

The location constraints are set as follows,

$$\begin{aligned} \Phi_{o_i^r}^s &= \{v_i^s, u_i^s\}, \quad i = 1, 2, \ldots, |V|, \\ \Phi_{x_i^r}^s &= \begin{cases} V_1^s, & i \le \dfrac{|V|}{2}, \\ V_2^s, & \text{otherwise}, \end{cases} \end{aligned}$$

where v_i^s and u_i^s are the i-th SNs in $\mathcal{V}$-island and $\mathcal{U}$-island, respectively.

The positive number Q is set as

$$Q = 2|V| + |V|^2 + |E| + 2K + 2\left(\left\lceil \frac{|V|}{2} \right\rceil^2 + \left\lfloor \frac{|V|}{2} \right\rfloor^2\right). \tag{7}$$

It is easy to verify that the construction above can be finished in polynomial time.

Fig. 2 gives an example of such a transformation with $|V| = 4$, $|E| = 4$, and $K = 2$. For the VNR, all the VNs and VLs have a unit requirement, and their location constraints are shown by their colors and/or patterns (*e.g.*, $\Phi_{o_1}^s = \{v_1, u_1\}$ and $\Phi_{x_1}^s = \bigcup_{i=1}^{4} \{v_i\}$). For the substrate network, the orange and green regions correspond to $\mathcal{V}$-island and $\mathcal{U}$-island, respectively. All the SNs have a unit computing capacity and all the black SLs have a unit bandwidth capacity, while the bandwidth capacities of the red and green SLs are 10 and 32, respectively. Here, for simplicity, we omit the superscripts of all the VNs and SNs.

Then, we prove that the instance of LC-VNE has a desired solution **if and only if** there is a desired graph bisection of V, *i.e.*, V_1 and V_2, satisfying $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \emptyset$, $|V_1| = \lceil \frac{|V|}{2} \rceil$, $|V_2| = \lfloor \frac{|V|}{2} \rfloor$ and $|\{(u, v) \in E : u \in V_1, v \in V_2\}| \le K$.

Suppose that the two disjoint subsets V_1 and V_2 are the desired bisection of $G(V, E)$. We denote $\widetilde{K} = |(u, v) \in E : u \in V_1, v \in V_2|$. Then, we have $\widetilde{K} \le K$. Firstly, we obtain the node mapping as follows.

- *Node Mapping of $\mathcal{O}$-Nodes*,

$$f_N(o_j^r) = \begin{cases} v_j^s, & \text{if } o_j \in V_1, \\ u_j^s, & \text{otherwise}, \end{cases} \quad j = 1, 2, \ldots, |V|.$$

- *Node Mapping of $\mathcal{X}$-Nodes*,

1) If $i \le \frac{|V|}{2}$,

$$f_N(x_i^r) = v_{\pi_i}^s, \quad v_{\pi_i}^s \in V_1^s \setminus \{f_N(v^r) : v^r \in V_1^r\}.$$

2) If $i > \frac{|V|}{2}$, we let $k = i - \lfloor \frac{|V|}{2} \rfloor$ and have

$$f_N(x_i^r) = u_{\pi_k}^s, \quad u_{\pi_k}^s \in V_2^s \setminus \{f_N(v^r) : v^r \in V_1^r\}.$$

Here, π_i denotes the index of the i-th remaining SN in an island (*i.e.*, the SN that has not accommodated any $\mathcal{O}$-nodes in $\mathcal{V}$-island or $\mathcal{U}$-island), and $X \setminus Y = \{z : z \in X \text{ and } z \notin Y\}$.

The link mapping is relatively obvious when we have obtained the node mapping, which is as follows,

1) **Case One**: Both end-nodes of a VL (v_1^r, v_2^r) are mapped onto SNs in the same island, and the link mapping is

$$f_L((v_1^r, v_2^r)) = \{f_N(v_1^r) - f_N(v_2^r)\}.$$

The number of the VLs in this case is $2\lceil \frac{|V|}{2} \rceil \lfloor \frac{|V|}{2} \rfloor + |E| - \widetilde{K}$, and hence, the consumed bandwidth resources are $2\lceil \frac{|V|}{2} \rceil \lfloor \frac{|V|}{2} \rfloor + |E| - \widetilde{K}$ units.

2) **Case Two**: The end-nodes of a VL (v_1^r, v_2^r) are mapped onto SNs in different islands. In this case, $\mathcal{VU}$-bridge will be crossed to embed the VL, *w.l.o.g.*, we assume that $f_N(v_1^r) \in V_1^s$ is in $\mathcal{V}$-island and $f_N(v_2^r) \in V_2^s$ locates in $\mathcal{U}$-island. Then, the link mapping is

$$f_L((v_1^r, v_2^r)) = \{f_N(v_1^r) - \kappa_1^s - \kappa_2^s - f_N(v_2^r)\}.$$

The number of the VLs in this case is $\left(\lceil \frac{|V|}{2} \rceil\right)^2 + \left(\lfloor \frac{|V|}{2} \rfloor\right)^2 + \widetilde{K}$, and the consumed bandwidth resources are $3\left(\left(\lceil \frac{|V|}{2} \rceil\right)^2 + \left(\lfloor \frac{|V|}{2} \rfloor\right)^2 + \widetilde{K}\right)$ units.

Finally, the total resource consumption of the LC-VNE solution mentioned above is

$$\widetilde{Q} = 2|V| + |V|^2 + |E| + 2\widetilde{K} + 2\left(\left\lceil \frac{|V|}{2} \right\rceil^2 + \left\lfloor \frac{|V|}{2} \right\rfloor^2\right).$$

As $\widetilde{K} \leq K$, we have $\widetilde{Q} \leq Q$, which means that the LC-VNE solution is a desired one.

For the example in Fig. 2, if we assume that the desired graph bisection for the instance in Fig. 2(a) is $V_1 = \{o_1, o_3\}$ and $V_2 = \{o_2, o_4\}$, the corresponding LC-VNE solution is as follows.

1) *Node Mapping*:

$$\begin{aligned}\{&o_1 \to v_1, o_2 \to u_2, o_3 \to v_3, o_4 \to u_4,\\ &x_1 \to v_2, x_2 \to v_4, x_3 \to u_1, x_4 \to u_3\}.\end{aligned}$$

2) *Link Mapping*:

$$\begin{aligned}\{&(o_1, o_2) \to \{v_1 - \kappa_1 - \kappa_2 - u_2\},\\ &(o_3, o_4) \to \{v_3 - \kappa_1 - \kappa_2 - u_4\},\\ &(o_1, x_3) \to \{v_1 - \kappa_1 - \kappa_2 - u_1\},\\ &(o_1, x_4) \to \{v_1 - \kappa_1 - \kappa_2 - u_3\},\\ &(o_3, x_3) \to \{v_3 - \kappa_1 - \kappa_2 - u_1\},\\ &(o_3, x_4) \to \{v_3 - \kappa_1 - \kappa_2 - u_3\},\\ &(x_1, o_2) \to \{v_2 - \kappa_1 - \kappa_2 - u_2\},\\ &(x_1, o_4) \to \{v_2 - \kappa_1 - \kappa_2 - u_4\},\\ &(x_2, o_2) \to \{v_4 - \kappa_1 - \kappa_2 - u_2\},\\ &(x_2, o_4) \to \{v_4 - \kappa_1 - \kappa_2 - u_4\},\\ &(o_1, x_1) \to \{v_1 - v_2\}, (o_1, x_2) \to \{v_1 - v_4\},\\ &(o_3, x_1) \to \{v_3 - v_2\}, (o_3, x_2) \to \{v_3 - v_4\},\\ &(o_2, x_3) \to \{u_2 - u_1\}, (o_2, x_4) \to \{u_2 - u_3\},\\ &(o_4, x_3) \to \{u_4 - u_1\}, (o_4, x_4) \to \{u_4 - u_3\},\\ &(o_1, o_3) \to \{v_1 - v_3\}, (o_2, o_4) \to \{u_2 - u_4\}\}.\end{aligned}$$

Conversely, assuming that f_N and f_L are for the desired LC-VNE solution and $\widetilde{Q}$ denotes the total resource consumption, we can get the following graph bisection, V_1 and V_2,

$$\begin{cases} o_j \in V_1, & \text{if } f_N(o_j^r) = v_j^s.\\ o_j \in V_2, & \text{if } f_N(o_j^r) = u_j^s.\end{cases}$$

As $\widetilde{Q} \leq Q$, there will be at most $K + \left(\lceil \frac{|V|}{2} \rceil\right)^2 + \left(\lfloor \frac{|V|}{2} \rfloor\right)^2$ VLs that are in **Case Two**. Among these VLs, there are $\left(\lceil \frac{|V|}{2} \rceil\right)^2 + \left(\lfloor \frac{|V|}{2} \rfloor\right)^2$ ones that are $\mathcal{XO}$-links. Hence, there would be at most K $\mathcal{O}$-links, and we have $|\{(u, v) \in E : u \in V_1, v \in V_2\}| \leq K$ and get the desired graph bisection of G.

Eventually, we prove that the LC-VNE problem is $\mathcal{NP}$-complete. ■

Next, we focus on analyzing the hardness of approximating LC-VNE. We first introduce the following lemma based on [36, p. 373, Lemma 10.2] and [37, Th. 10.1].

Lemma 1: Given a minimization problem Π and an $\mathcal{NP}$-complete problem Λ, if there exists a polynomial-time transformation g from Λ to Π, such that,

$$\begin{aligned}\lambda \in \Lambda &\Rightarrow OPT(g(\lambda)) \leq \alpha,\\ \lambda \notin \Lambda &\Rightarrow OPT(g(\lambda)) > \beta,\end{aligned}$$

where $\lambda \in \Lambda$ means that instance λ is a "YES" instance for problem Λ, and $OPT(g(\lambda))$ denotes the value of the optimal solution of $g(\lambda)$, then there is no $\frac{\beta}{\alpha}$-approximation algorithm[4] for optimization problem Π.

Based on *Lemma 1*, we get the following hardness result of approximating LC-VNE.

Theorem 2: There exists some $\epsilon > 0$ such that approximating LC-VNE problem within a factor of $m^{1-\epsilon}$ is $\mathcal{NP}$-hard, where m is the size of the substrate network, *i.e.*, the number of SNs.

Proof: We first construct a transformation $\widehat{g}$ from graph bisection (decision version, denoted as GBD) to LC-VNE (optimization version), which is similar to the one we designed in *Theorem 1*. The only difference is that, we set $\Delta = poly(|V|) \cdot Q$ in $\widehat{g}$, where Q is defined by Eq. (7), and $poly(|V|)$ denotes a polynomial function of $|V|$, whose highest power is no less than 1. It is obvious that $\widehat{g}$ is still a polynomial-time one. Based on the analysis in *Theorem 1*, we can prove that for any instance λ of graph bisection, we have

$$\begin{aligned}\lambda \in GBD &\Rightarrow OPT(\widehat{g}(\lambda)) \leq Q,\\ \lambda \notin GBD &\Rightarrow OPT(\widehat{g}(\lambda)) > \Delta.\end{aligned}$$

As $\frac{\Delta}{2|V|+|V|^2+|E|+2K+2\left(\left\lceil \frac{|V|}{2} \right\rceil^2 + \left\lfloor \frac{|V|}{2} \right\rfloor^2\right)} = poly(|V|)$ and $poly(|V|) \approx |V^s|^{1-\frac{2}{k+2}}$, where $k \geq 1$ is the highest power of $poly(|V|)$, we can rewrite the reduction gap as $|V^s|^{1-\epsilon} = m^{1-\epsilon}$, for some $\epsilon > 0$. Therefore, it is $\mathcal{NP}$-hard to approximate LC-VNE within $m^{1-\epsilon}$, for some $\epsilon > 0$. ■

The in-approximability result of LC-VNE above suggests that our investigation on LC-VNE should switch from finding approximation algorithms to designing efficient heuristics. We design two efficient LC-VNE heuristics to minimize the total resource consumption in Eq. (1) in the following sections.

V. LC-VNE Framework Based on CG

In this section, we leverage compatibility graph (CG) to design an LC-VNE framework that achieves integrated node and link mapping. *Algorithm* 1 shows the overall procedure

[4]An α-approximation algorithm is a polynomial-time algorithm that can always get a solution whose value is at most α times of the optimal for a minimization problem or at least $\frac{1}{\alpha}$ of the optimal for a maximization problem.

Algorithm 1 LC-VNE Framework Based on CG

```
1  obtain the incoming VNR G^r;
2  get current status of the substrate network G^s;
3  preprocess VNR G^r with Algorithm 2;
4  construct CG for the LC-VNE with Algorithm 3;
5  if CG is successfully constructed then
6  |  get the maximum clique in CG with Algorithm 4;
7  |  if the maximum clique is found then
8  |  |  mark VNR G^r as accepted;
9  |  else
10 |  |  mark VNR G^r as blocked;
11 |  end
12 else
13 |  mark VNR G^r as blocked;
14 end
15 wait for the next VNR;
```

of the framework, which has two key steps: 1) constructing the CG (*Line* 4), and 2) finding the minimum-cost maximum clique (*Line* 6). Before them, we need to preprocess the VNR with *Algorithm* 2 (*Line* 3) to generalize the framework to support the cases in which the candidate SN sets for VNs are not disjoint. The details of *Algorithms* 2-4 are explained in the following subsections.

A. Preprocessing

Note that the algorithms discussed in Subsections V-B and V-C work on the assumption that the candidate SN sets for any two VNs in the VNR are disjoint, *i.e.*, $\Phi^s_{v^r_1} \cap \Phi^s_{v^r_2} = \emptyset, \{v^r_1, v^r_2 : v^r_1 \neq v^r_2\}$. However, in practical LC-VNE, this may not always be the case. Fortunately, we can apply preprocessing on VNRs to generalize our framework and make the candidate SN sets that have non-empty intersections disjoint.

The preprocessing works as follows. In *Lines* 2-8, we remove all SNs that do not have enough computing capacities. Then, for the VNR, we first get the number of the VNs that an SN is a candidate for, and remove certain SNs from the candidate SN sets of some VNs, according to the set sizes. Specifically, for SN v^s that is a candidate for two or more VNs, we reserve it as the candidate for the VN whose candidate SN set has the smallest size, and remove it from the candidate SN sets of the rest VNs. *Algorithm* 2 shows the detailed procedure, which ensures that an arbitrary VNR would become the one whose VNs have disjoint candidate SN sets. Its time complexity is $O(|V^r| \cdot |V^s|^2)$. Note that even though the preprocessing does reduce the solution space of LC-VNE, it works well in practice, as we will show later in Section VI.

B. CG for LC-VNE

1) Construction of CG: To solve LC-VNE, we construct a CG, in which each node represents a candidate substrate path for a VL $e^r \in E^r$, and if two substrate paths are compatible with each other, we insert a link to connect their corresponding nodes in the CG. Here, by saying two substrate paths are "compatible", we mean that they can carry two VLs in the

Algorithm 2 Preprocessing

input : Original candidate SN sets $\{\Phi^s_{v^r}\}$, SN set V^s, and VN set V^r
output: New candidate SN sets $\{\Phi^s_{v^r}\}$

```
1  Ψ_{v^s} ← ∅, ∀v^s ∈ V^s;
2  foreach v^r ∈ V^r do
3  |  foreach v^s ∈ Φ^s_{v^r} do
4  |  |  if c^r_{v^r} > c^s_{v^s} then
5  |  |  |  remove v^s from Φ^s_{v^r};
6  |  |  end
7  |  end
8  end
9  foreach v^r ∈ V^r do
10 |  foreach v^s ∈ Φ^s_{v^r} do
11 |  |  Ψ_{v^s} ← Ψ_{v^s} ∪ {v^r};
12 |  end
13 end
14 foreach v^s ∈ V^s do
15 |  if |Ψ_{v^s}| > 1 then
16 |  |  v^r_m = arg min_{v^r ∈ Ψ_{v^s}} |Φ^s_{v^r}|;
17 |  |  Ψ_{v^s} ← Ψ_{v^s} \ {v^r_m};
18 |  |  foreach v^r ∈ Ψ_{v^s} do
19 |  |  |  Φ^s_{v^r} ← Φ^s_{v^r} \ {v^s};
20 |  |  end
21 |  end
22 end
```

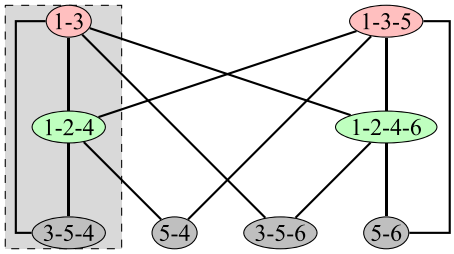


Fig. 3. The compatibility graph (CG) for the upper VNR in Fig. 1(b). The nodes in red represent the candidate substrate paths for VL (a, b), the green ones are for VL (a, c), and the gray ones for VL (b, c). The shadow region shows a maximum clique corresponded to the LC-VNE result shown in Fig. 1(c).

VNR simultaneously. Or in other words, the substrate paths should satisfy: 1) they are the candidate paths for two adjacent VLs and have one SN as the common end-node, or 2) they are the candidate paths for two VLs that are not adjacent.

Fig. 3 shows the CG for the upper VNR in Fig. 1(b). Here, for simplicity, we only consider the shortest substrate path for each SN pair in Fig. 1(a). Note that our CG construction can also consider more candidate substrate paths in addition to the shortest ones. Each row of the CG represents the candidate substrate paths of a particular VL in the VNR. For instance, since VNs a and b can be mapped onto SNs $\{1\}$ and $\{3, 5\}$, respectively, VL (a, b) can be mapped onto substrate paths $\{1-3\}$ or $\{1-3-5\}$, as shown in the first row of the CG. Then, the corresponding nodes in the CG are connected according to the compatibility relation mentioned above. In Fig. 3, the CG's nodes for substrate paths $\{1-3\}$ and

Algorithm 3 CG Construction

input : Substrate network G^s and VNR G^r
output: CG $G^c(V^c, E^c)$ and construction status F

1 $V^c \leftarrow \varnothing$, $E^c \leftarrow \varnothing$;
2 **foreach** $e^r \in E^r$ **do**
3 **foreach** $p \in P^s_{e^r}$ **do**
4 **if** $\min_{e^s \in p} b^s_{e^s} \geq b^r_{e^r}$ **then**
5 insert a node in V^c to represent p;
6 **else**
7 remove p from $P^s_{e^r}$;
8 **end**
9 **end**
10 **if** $P^s_{e^r} = \varnothing$ **then**
11 return(F = FAILURE);
12 **end**
13 **end**
14 **foreach** $e^r_1 \in E^r$ **do**
15 **foreach** $e^r_2 \in E^r, e^r_2 \neq e^r_1$ **do**
16 **if** e^r_1 *and* e^r_2 *are adjacent through VN* v^r **then**
17 **foreach** $v^s \in \Phi^s_{v^r}$ **do**
18 $E^c \leftarrow E^c \cup ((P^s_{e^r_1} \cap P^s_{v^s}) \times (P^s_{e^r_2} \cap P^s_{v^s}))$;
19 **end**
20 **else**
21 $E^c \leftarrow E^c \cup (P^s_{e^r_1} \times P^s_{e^r_2})$;
22 **end**
23 **end**
24 **end**
25 return(F = SUCCESS);

$\{1-2-4\}$ are connected, since the link mappings $(a,b) \rightarrow \{1-3\}$ and $(a,c) \rightarrow \{1-2-4\}$ are compatible, *i.e.*, they can coexist in an LC-VNE solution. On the other hand, as the link mappings $(a,b) \rightarrow \{1-3\}$ and $(b,c) \rightarrow \{5-4\}$ cannot coexist, their nodes in the CG are not connected. The shadow region shows a maximum clique, *i.e.*, $\{\{1-3\}, \{1-2-4\}, \{3-5-4\}\}$, which corresponds to the LC-VNE solution in Fig. 1(c).

Algorithm 3 shows the detailed procedure of CG construction. *Line 1* initializes the node set V^c and link set E^c. *Lines 2-13* describe the construction of V^c, which is the set of the candidate substrate paths that have enough bandwidth to serve the corresponding VLs. Note that, in this work, we only consider single-path based link mapping [21]. Thus, in *Lines* 3-9, we remove the candidate substrate paths whose bandwidth capacities are insufficient for the corresponding VLs. The CG construction can also be extended to consider a more generic scheme in which multi-path based link mapping is allowed, which will be our future work.

Theorem 3: The time complexity of *Algorithm 3* is $O\left(K^2 \cdot |V^s|^3 \cdot (|V^r|^2 + |V^s|)\right)$, where K is the maximum number of candidate substrate paths between any SN-pair.

Proof: For the for-loop covering *Lines 2-13*, since in the worse case, each substrate path would be traversed once, its time complexity is $O(K \cdot |V^s|^3)$.

For the for-loop covering *Lines 14-24*, as the time complexity for calculating the intersection of two sets with m and n elements ($m \leq n$) is $O\left(m \cdot (1 + \log\left(\frac{n}{m}\right))\right)$ [38], the set intersection in *Line 18* would use $O(\max\{|P^s_{e^r_1}|, |P^s_{e^r_2}|, |P^s_{v^s}|\})$ operations,[5] and the Cartesian product in *Line 18* needs at most $|P^s_{v^s}|^2 \leq K^2 \cdot |V^s|^2$ operations. As $|\Phi^s_{v^r}| \leq |V^s|$, we have $\max\{|P^s_{e^r_1}|, |P^s_{e^r_2}|\} < K \cdot |V^s|^2$. Thus, each iteration in the inner for-loop that covers *Lines 17-19* has the complexity of $O(K^2 \cdot |V^s|^2)$. Furthermore, since the candidate SN sets for any two VNs in the VNR are disjoint, *Line 18* would be executed for at most $|V^r|^2 \cdot |V^s|$ times. Hence, the total number of operations executed for *Line 18* is bounded by $O(K^2 \cdot |V^r|^2 \cdot |V^s|^3)$. Meanwhile, the maximum number of operations executed for *Line 21* would be less than $\sum_{e^r_1 \in E^r} \sum_{e^r_2 \in E^r : e^r_1 \neq e^r_2} |P^s_{e^r_1}| \cdot |P^s_{e^r_2}|$. Since the following relation holds

$$\sum_{e^r_1 \in E^r} \sum_{e^r_2 \in E^r : e^r_1 \neq e^r_2} |P^s_{e^r_1}| \cdot |P^s_{e^r_2}| < \left(\sum_{e^r \in E^r} |P^s_{e^r}|\right)^2 < K^2 \cdot |V^s|^4.$$

Therefore, the time complexity of *Algorithm 3* is $O\left(K^2 \cdot |V^s|^3 \cdot (|V^r|^2 + |V^s|)\right)$. ■

2) Property of CG: We establish and prove the following property of the CG constructed by *Algorithm* 3, since our CG based LC-VNE algorithms rely on the maximum clique of CG. In general, maximum clique problem in an arbitrary graph is $\mathcal{NP}$-hard to approximate within a factor of $n^{1-\epsilon}$, for any $\epsilon > 0$, where n is the size of the graph [39]. However, finding a maximal clique is much easier and can be finished in linear time, the following good property of CG ensures that our CG-based LC-VNE algorithms do not have complexity issues.

Theorem 4: Given the substrate network $G^s(V^s, E^s)$ and VNR $G^r(V^r, E^r)$, if the substrate network satisfies that, for each VL, there is at least one substrate path to connect any pair of its candidate SNs with enough bandwidth, *i.e.*, $\{p^s : p^s \in P^s_{v^s_1, v^s_2}, \min_{e^s \in p^s} b^s_{e^s} \geq b^r_{(v^r_1, v^r_2)}\} \neq \emptyset$, where $v^s_i \in \Phi^s_{v^r_i}$, $v^r_i \in V^r, i = 1, 2$, $v^r_1 \neq v^r_2$, then in the CG, any maximal clique is also the maximum clique.[6] The size of the maximum clique is equal to the number of VLs in the VNR, *i.e.*, $|E^r|$.

Proof: We first prove that the size of the maximum clique in the CG cannot exceed $|E^r|$ by contradiction. Suppose that the former is larger than the later. Then, according to the Pigeonhole Principle [40], the maximum clique includes at least two nodes that represent candidate substrate paths of the same VL. However, *Algorithm* 3 has ensured that there would be no link between two such nodes in the CG, and this results in contradiction. Therefore, the size of the maximum clique cannot exceed $|E^r|$.

Also by using contradiction, we can prove that the size of any maximal clique in the CG cannot be less than $|E^r|$. Here, for the sake of convenience, we say a VL is covered, if one of its candidate substrate paths is included in a maximal clique. Then, according to the Pigeonhole Principle, if the size of a maximal clique is less than $|E^r|$, there will be at

[5]Here, we use the relaxation that $m \cdot (1 + \log\frac{n}{m}) \leq (\frac{2^{1-\frac{1}{\ln 2}}}{\ln 2}) \cdot n \approx 1.0615n$, for any $n \geq m > 0$.

[6]A clique is a complete subgraph in a graph, a maximal clique is the clique that we cannot add any other node into it and still make it a clique, and the maximum clique is the clique that includes the most nodes.

least one VL that is not covered, which means that none of its candidate substrate paths is compatible with those in the maximal clique. We denote this maximal clique as MC, and use L and UL to represent the covered and uncovered VLs, respectively.

Let e^r_{ul} and e^r_l be any element in UL and L, respectively. For any e^r_{ul}, denote $k = |\bigcup_{e^r_l \in L} e^r_l \wedge e^r_{u1}|$. Here, we use operation $e^r_l \wedge e^r_{u1}$ to obtain the common end-node(s) of e^r_l and e^r_{ul}. Apparently, we have $k > 0$ because $k = 0$ indicates that e^r_{ul} has no common end-node with any VL in L. Hence, according to *Algorithm* 3, in the CG, all the nodes that represent the candidate substrate paths of e^r_{ul} are connected with all the nodes in MC. This, however, is contradicted with the fact that MC is the maximal.

Therefore, we are left with two cases to analyze:

- $k = 1$: We consider those VLs in L, which are adjacent to e^r_{ul}, and denote them as L_0. Since we have $k = 1$, all the VLs in L_0 should have the same common end-node with e^r_{ul}, which can be denoted as v^r_0. We assume that v^r_0 is mapped onto SN v^s_0, *i.e.*, $f_N(v^r_0) = v^s_0$. Then, according to *Algorithm* 3, all the nodes in the CG for the substrate paths in $P^s_{e^r_{ul}} \cap P^s_{v^s_0}$ are connected with all the nodes in MC. This, however, is also contradicted with the maximal of MC.
- $k = 2$: Similar to the case of $k = 1$, we just need to consider the VLs in L, which are adjacent to e^r_{ul}. For the two common end-nodes, we denote them as v^r_1 and v^r_2, and the corresponding SNs are v^s_1 and v^s_2, respectively. Obviously, in the CG, all the nodes for the substrate paths in $P^s_{v^s_1, v^s_2}$ are connected with all the nodes in MC. Again, this is contradicted with the maximal of MC.

Hence, we prove that the size of any maximal clique in the CG cannot be less than $|E^r|$, which in turn verifies that in the CG, any maximal clique is also the maximum clique. ■

C. Heuristic MCMC Algorithms

With CG, we can transform the LC-VNE problem into the minimum-cost maximum clique (MCMC) problem. More specifically, in order to solve LC-VNE, we need to find the maximum clique to cover all the VLs in the VNR with the minimum total consumed resources (calculated with Eq. (1)), while satisfying the constraints in Eqs. (2)-(6). Even though finding the maximum clique in an arbitrary graph is $\mathcal{NP}$-hard [29], our CG-based LC-VNE algorithms would not have complexity issues, since we have proved that in the CG, a maximal clique is also the maximum clique as long as there are sufficient bandwidth resources in the substrate network.

We design a heuristic to find the near-optimal MCMC in the CG, and *Algorithm* 4 shows the procedure. The time complexity of *Algorithm 4* is $O(K^2 \cdot |E^r| \cdot |V^s|^2)$. In *Line* 2, P^N is the set of all the substrate paths whose nodes in the CG are the potential members of the maximum clique M. Here, we use function $f_{SP}(\cdot)$ to obtain the substrate path that a node in the CG represents, while $f^{-1}_{SP}(\cdot)$ is the inverse function. *Line* 2 first initializes P^N as all the substrate paths. In *Line* 4, we obtain the substrate paths that have enough bandwidth capacities to carry VL e^r, and store them in $\widehat{P}^s_{e^r}$. The operation in *Line* 8 then selects the desired substrate path p^s_m for e^r from $\widehat{P}^s_{e^r}$ based on the paths' h-parameters.

Algorithm 4 Heuristic MCMC Algorithms

input : CG G^c, substrate network G^s and VNR G^r
output: Maximum clique M, status F

1 $M \leftarrow \varnothing$;
2 $P^N \leftarrow f_{SP}(V^c)$;
3 **for** $e^r \in E^r$ *in non-increasing order of* $b^r_{e^r}$ **do**
4 $\quad \widehat{P}^s_{e^r} \leftarrow \{p^s : p^s \in P^s_{e^r} \cap P^N, (\min_{e^s \in p^s} b^s_{e^s}) \geq b^r_{e^r}\}$;
5 $\quad$ **if** $\widehat{P}^s_{e^r} = \varnothing$ **then**
6 $\quad\quad$ return(F = FAILURE);
7 $\quad$ **end**
8 $\quad$ select p^s_m from $\widehat{P}^s_{e^r}$ with the minimum h-parameter defined by Eq. (8) or Eq. (9);
9 $\quad M \leftarrow M \cup \{f^{-1}_{SP}(p^s_m)\}$;
10 $\quad P^N \leftarrow P^N \cap \{p^s : (f^{-1}_{SP}(p^s), f^{-1}_{SP}(p^s_m)) \in E^c\}$;
11 $\quad$ update bandwidth resources;
12 **end**
13 update computing resources;
14 return(F = SUCCESS);

We develop two ways to calculate the h-parameter of a substrate path and will discuss the details later. *Line* 9 includes the node for p^s_m in M, and *Line* 10 updates P^N by finding the intersection of the current P^N and the set of substrate paths whose nodes in the CG are connected with node $f^{-1}_{SP}(p^s_m)$. The operations in *Lines* 3-12 are repeated until M covers all the VLs in VNR or we cannot add any other node into M.

With the two different ways to calculate the h-parameter in *Algorithm* 4, we obtain two LC-VNE heuristics, *i.e.*, the greedy algorithm based on CG (G-CG) and load-balance enhanced algorithm based on CG (LBE-CG).[7]

1) Greedy LC-VNE Based on CG (G-CG): In G-CG, we greedily select the shortest substrate path from $\widehat{P}^s_{e^r}$, *i.e.*, the one that has the smallest number of hops. Then, we have

$$h^{(g)} = |p^s|, \quad \forall p^s \in \widehat{P}^s_{e^r}. \tag{8}$$

The motivation for defining h-parameter as the path's hop-count is straightforward, since the shortest substrate path consumes the least bandwidth resources in link mapping.

2) Load-Balance Enhanced LC-VNE Based on CG (LBE-CG): LBE-CG takes load-balancing into consideration, and h-parameter is defined as the hop-count of a substrate path divided by its available bandwidth capacity,

$$h^{(lb)} = \frac{|p^s|}{\delta + \min_{e^s \in p^s} b^s_{e^s}}, \quad \forall p^s \in \widehat{P}^s_{e^r}. \tag{9}$$

where δ is a small positive number to avoid zero denominator.

[7]Here, G-CG or LBE-CG includes all the algorithms in the LC-VNE framework, *i.e.*, operating as *Algorithm* 1.

TABLE II
SIMULATION PARAMETERS

Substrate network	Number of SNs	50
	Number of SLs	172
	SNs' initialized computing capacity	50 - 100 units
	SLs' initialized bandwidth capacity	50 - 100 units
	K, number of candidate substrate paths for a VL	5
VNRs	Number of VNs	2 - 10
	VNs' connectivity rate	0.5
	VNs' computing requirement	0 - 20 units
	VLs' bandwidth requirement	0 - 20 units
	Radius	$\{20, 22, 25, 28, 30\}$
	Traffic model	Poisson model
	λ, VNRs' average arrival rate	0.2 - 6 per 100 time-units
	$\frac{1}{\mu}$, VNRs' average holding time	1000 time-units

D. Time Complexity of CG Based LC-VNE

Considering all the sub-procedures (*i.e.*, *Algorithms* 2-4), we get the time complexity of the CG based LC-VNE algorithms (*i.e.*, G-CG and LBE-CG) as $O\left(K^2 \cdot |V^s|^3 \cdot (|V^r|^2 + |V^s|)\right)$.

VI. PERFORMANCE EVALUATION

In this section, we evaluate the performance of the proposed LC-VNE framework. Similar to our previous work in [21], the substrate network and all the VNRs are randomly generated with the GT-ITM tool [41]. The substrate network G^s has 50 SNs and 172 SLs, and all the SNs are located within a 100×100 grid. The number of VNs in each VNR is randomly selected from 2 to 10, and the probability that any two VNs are connected (*i.e.*, the VNs' connectivity rate) is set as 0.5. For each VN v^r, its preferred location $l^r_{v^r}$ is randomly located in the 100×100 grid, and the candidate SNs are those who are located within the circle that is centered at $l^r_{v^r}$ and has a radius of ρ, whose default value is 20. For each SN-pair in the substrate network, we pre-calculate K shortest paths with the Yen's algorithm [42]. The VNRs are generated according to a Poisson process, which has an average arrival rate of λ VNRs per time-unit and the average holding time of each VNR is $\frac{1}{\mu}$ time-units. The traffic load of VNRs is quantified as $\frac{\lambda}{\mu}$ in Erlangs. Table II summarizes the simulation parameters.

A. Performance Metrics

The objective of LC-VNE is to accommodate as many VNRs as possible and to maximize the revenue of InP. Hence, we use VNR blocking probability and time-average revenue as the performance metrics.

- *VNR Blocking Probability*: It is defined as the ratio of blocked to total arrived VNRs, *i.e.*,

$$p_b = \lim_{T \to \infty} \frac{|\Omega_b(T)|}{|\Omega_b(T)| + |\Omega_a(T)|}, \tag{10}$$

  where $\Omega_b(T)$ and $\Omega_a(T)$ denote the sets of blocked and accepted VNRs during $[0, T]$, respectively.

- *Time-Average Revenue*: It is defined as the total revenue of accepted VNRs averaged over time, *i.e.*,

$$\bar{\mathcal{R}} = \lim_{T \to \infty} \frac{\sum_{G^r \in \Omega_a(T)} \mathcal{R}_{G^r}}{T}, \tag{11}$$

  where $\mathcal{R}_{G^r}$ is the revenue generated by VNR G^r, as

$$\mathcal{R}_{G^r} = \Big(\sum_{v^r \in V^r} c^r_{v^r} + \sum_{e^r \in E^r} b^r_{e^r} \Big) \cdot \tau^r. \tag{12}$$

  Here, τ^r is the holding time of G^r.

B. Benchmark Algorithms

We use the algorithms proposed in [14] and their modified versions as the benchmarks. They are denoted as DViNE, DViNE-LB, DViNE-KSP, and DViNE-LB-KSP, respectively. Among them, those ending with "KSP" stand for the modified versions with *K*-shortest path routing, while the others are the original ones proposed in [14]. The modified versions are used for the reason that the original ones did not apply any limitation on the number of candidate substrate paths for each VL, which can be impractical for large substrate networks. All the benchmark algorithms use LP relaxation to solve the MCF-based MILP, with/without considering load-balance. After that, they perform deterministic rounding on the solution to get a feasible node mapping, and then conduct link mapping by solving the MCF or by using the *K*-shortest-path routing to select the shortest substrate path with enough bandwidth capacity for a VL. Actually, we have also implemented four counterparts with random rounding. But as they provide similar results, we omit their results due to the page limit of the paper. Table III summarizes all the benchmarks.

C. Simulation Results

1) Impact of Preprocessing: We first investigate the impact of the preprocessing on our algorithms. Initially, we make the substrate network have certain resource utilizations on the SNs and SLs, with the distributions in Fig. 4.

TABLE III
BENCHMARK ALGORITHMS

Algorithms	Node mapping	Link mapping
DViNE	Solving LP relaxation of the MCF-based MILP without considering load balance, and then using deterministic rounding approach to get node mapping results	Solving MCF
DViNE-KSP		Using the K-shortest-path routing
DViNE-LB	Solving LP relaxation of the MCF-based MILP considering load balance, and then using deterministic rounding approach to get node mapping results	Solving MCF with consideration of load-balance
DViNE-LB-KSP		Using K-shortest-path routing

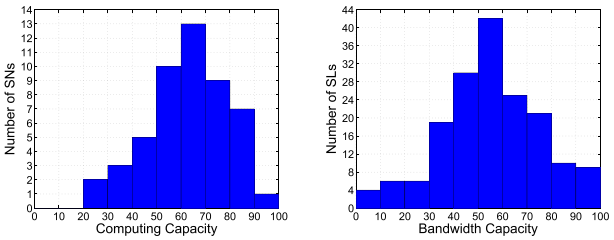


Fig. 4. Computing and bandwidth capacity distributions used in the simulations for evaluating preprocessing.

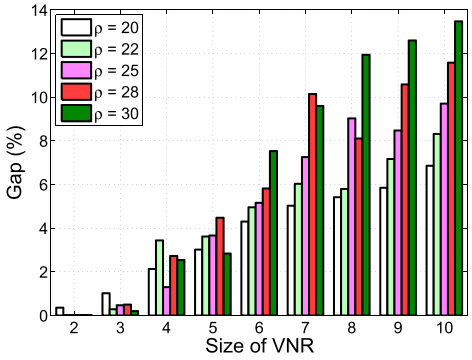


Fig. 5. Optimization gaps for preprocessing.

Then, we generate VNRs randomly. Specifically, the number of VNs ranges from 2 to 10, VN's connectivity rate is fixed as 0.5, and the radius $\rho \in [20, 30]$. For each VNR, we first use the MILP proposed in [14] to get its optimal solution and the associated total resource consumption $\mathcal{R}^*$ (calculated with Eq. (1)), and then solve the MILP again with the preprocessing in *Algorithm* 2 for another LC-VNE solution with the total resource consumption $\mathcal{R}^*_{pre}$. The optimization gap can be obtained as

$$\Delta = \frac{\mathcal{R}^*_{pre} - \mathcal{R}^*}{\mathcal{R}^*}. \tag{13}$$

Fig. 5 shows the results on optimization gap. Each data point in the figure is averaged over 50 VNRs. We can see that since for two different VNs in a VNR, the probability that their candidate SN sets overlap with each other will increase with the size of VNR and the radius, the gap also becomes larger in the simulations. However, even when the size of VNR is 10 and the radius is 30, the optimization gap is still less than 14%, which is relatively small compared to the results from the heuristics, which will be shown in Subsection VI-C2.

2) Solution Quality Analysis: We then investigate the solution quality of our proposed algorithms. Fig. 6 shows the distributions of the computing and bandwidth capacities in

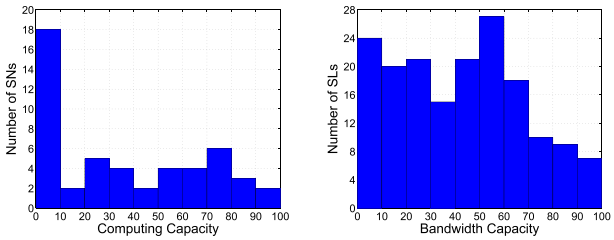


Fig. 6. Computing and bandwidth capacity distributions used in the simulations for evaluating solution quality.

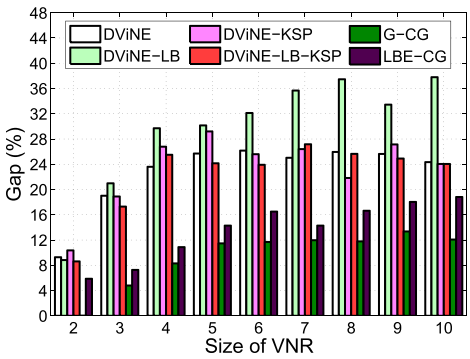
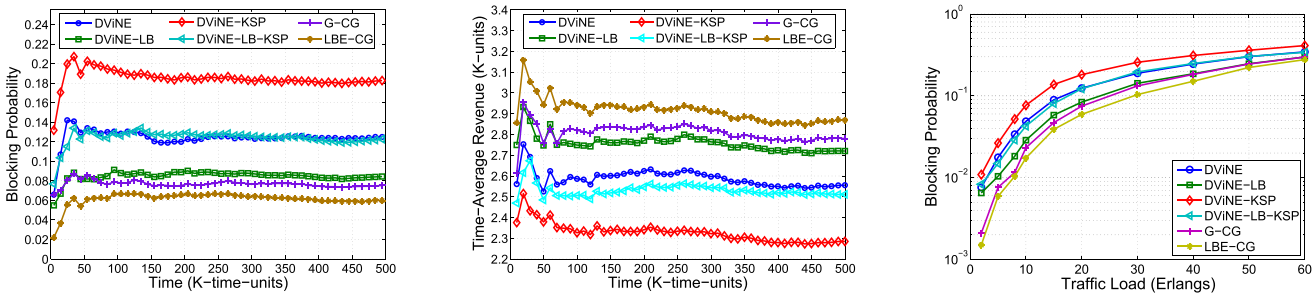


Fig. 7. Optimization gaps from different algorithms.

TABLE IV
NUMBER OF VNRs FOUND SOLUTIONS

Size of VNR	2	3	4	5	6	7	8	9	10
MILP	50	50	50	50	50	49	47	48	50
DViNE	50	50	50	50	49	49	47	46	46
DViNE-LB	50	50	50	50	50	49	47	47	46
DViNE-KSP	48	45	45	34	26	24	18	14	9
DViNE-LB-KSP	50	49	48	43	41	37	31	31	22
G-CG	50	50	49	48	46	43	34	37	42
LBE-CG	50	50	50	49	48	46	39	44	44

the substrate network. In the simulations, we fix the radius as $\rho = 20$. For each VNR, we first use the MILP to get the optimal LC-VNE solution, and then try to use each heuristic algorithm to get another solution. Similar to that in Subsection VI-C1, we calculate the optimization gap and average the results from 50 VNRs for each data point. Fig. 7 shows the gaps from different heuristics, and Table IV provides the number of VNRs for which the heuristics can get a solution. The results in Fig. 7 indicate that our CG-based heuristics always provide the smallest optimization gaps. When the VNR size is 10, their gaps are still less than 19%. We also notice that when other conditions are the same, the gaps from the algorithms without considering load-balance are smaller than those with, except for DViNE-KSP and



(a) (b) (c)

Fig. 8. Simulation results. (a) Blocking probability. (b) Time-average revenue. (c) Blocking probability vs. traffic loads.

DViNE-LB-KSP. This is because in general, load-balance can let the algorithm select an LC-VNE solution with higher resource consumption for making the residual resources more balanced. But since DViNE-LB-KSP only considers load-balance in node mapping with the MCF-based load-balance model [14], it might not consume more resources in the link mapping stage.

It is interesting to notice that the numbers of VNRs for which DViNE and DViNE-LB can get a solution are more than those from our CG-based algorithms. This observation is reasonable since they use MCF in link mapping (*i.e.*, allowing multi-path mapping), while our heuristics are both based on single-path link mapping.

3) Baseline Performance Comparisons: We then perform dynamic LC-VNE simulations with the scenario that VNRs can come and leave on-the-fly. The performance of the heuristics are first evaluated under a fixed traffic load as 20 Erlangs. We simulate each algorithm for $500,000$ time-units and monitor its performance in a long run. Fig. 8(a) compares the results on blocking probability. We obverse that each algorithm that considers load-balance achieves lower blocking probability than its counterpart that does not. Since our CG-based LC-VNE algorithms can achieve integrated node and link mapping, they provide lower blocking probabilities than the four benchmarks. It is worth noting that, our CG-based algorithms (with single-path mapping) not only provide lower blocking probabilities than the benchmarks that use single-path mapping (*i.e.*, DViNE-KSP and DViNE-LB-KSP), but also outperform DViNE and DViNE-LB, which allow multi-path mapping. Specifically, when simulations enter the stationary stage, G-CG and LBE-CG reduce the blocking probability by 39.68%, 10.74%, 58.86%, 38.64%, and 52.10%, 29.12%, 67.33%, 51.27%, respectively, when compared with DViNE, DViNE-LB, DViNE-KSP and DViNE-LB-KSP.

Under the Poisson traffic model, the relation between time-average revenue and blocking probability in stationary stage can be approximated by,

$$\bar{\mathcal{R}} \simeq \lambda(1 - p_b)\bar{\mathcal{R}}_0, \tag{14}$$

where $\bar{\mathcal{R}}_0$ denotes the average revenue generated by the accepted VNRs from an LC-VNE algorithm. Eq. (14) indicates that as different algorithms can provide different $\bar{\mathcal{R}}_0$, a lower blocking probability may not necessarily lead to a larger time-average revenue. We compare the results on time-average revenue in Fig. 8(b), which show that

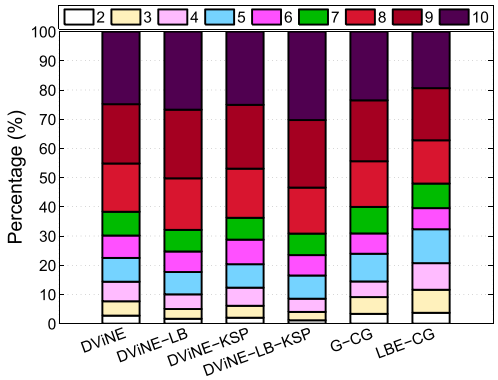
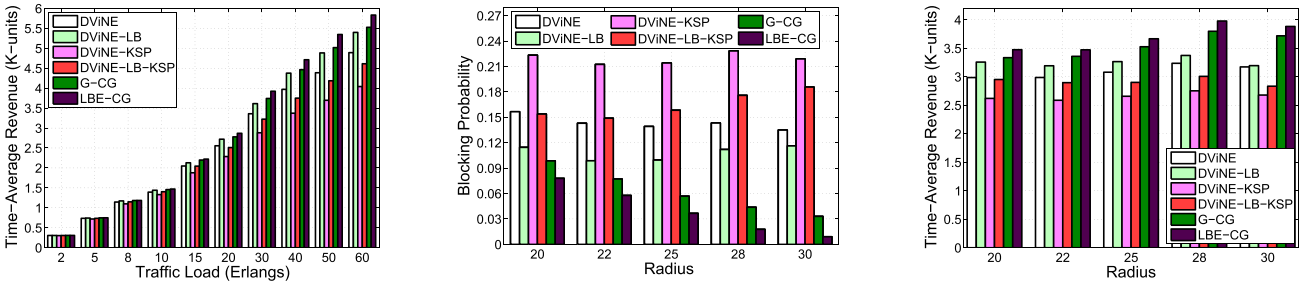


Fig. 9. Percentages of blocked VNRs with different sizes.

our CG-based algorithms also provide larger time-average revenue. The revenue gains from G-CG and LBE-CG over DViNE, DViNE-LB, DViNE-KSP and DViNE-LB-KSP are 8.77%, 2.19%, 21.65%, 10.69%, and 12.25%, 5.46%, 25.54%, 14.24%, respectively. However, according to Eq. (14) and the results in Fig. 8(a), if we assume that all the algorithms provide the same $\bar{\mathcal{R}}_0$, the revenue gains should be 5.62%, 0.98%, 13.08%, 5.36%, and 7.37%, 2.66%, 14.96%, 7.12%, respectively. The differences here suggest that our proposed algorithms also achieve larger $\bar{\mathcal{R}}_0$ than the benchmarks, which can also be verified by analyzing the percentages of blocked VNRs versus their sizes (*i.e.*, numbers of VNs).

The results on percentages of blocked VNRs with different sizes are plotted in Fig. 9. With Fig. 9, we can easily figure out that our proposed algorithms provide larger $\bar{\mathcal{R}}_0$, since they block smaller percentages of VNRs that have relatively large sizes. The standard deviations of the percentages obtained by DViNE, DViNE-LB, DViNE-KSP, DViNE-LB-KSP, G-CG, and LBE-CG for VNRs with different sizes are 0.0715, 0.0863, 0.0767, 0.0934, 0.0680, and 0.0493, respectively, which indicates that LBE-CG achieves the best fairness among VNRs with different sizes. This observation can be explained as follows. Since they consider the node and link mappings separately, the benchmarks may result in improper node mapping that may make the subsequent link mapping consume unnecessarily large bandwidth resources and eventually lead to link mapping failures. When the VNR's size becomes larger, this negative effect becomes more severe. Consequently, it is more difficult for these algorithms to embed VNRs with larger sizes. Although G-CG achieves integrated node and link mapping, it may cause congestion on



(a) (b) (c)

Fig. 10. Simulation results. (a) Time-average revenue vs. traffic loads. (b) Blocking probability vs. radius. (c) Time-average revenue vs. radius.

some critical SLs due to its greedy nature, which will block future embedding. LBE-CG not only achieves integrated node and link mapping, but also considers load-balance. Hence, it leaves more resources for future embedding and relieves SLs' congestions.

4) Sensitivity Analysis: To evaluate the sensitivity of the algorithms, we compare the results on blocking probability and time-average revenue under different traffic loads and with different radius, *i.e.*, different sizes of candidate SN sets.

The results on blocking probability and time-average revenue under different traffic loads are plotted in Figs. 8(c) and 10(a), respectively. It can be seen that our proposed algorithms always achieve lower blocking probability and larger time-average revenue than the four benchmarks. The advantage of our CG-based algorithms can be traced back to the integrated node and link mapping. Hence, even though the preprocessing in *Algorithm* 2 reduces the solution space of LC-VNE, they still provide better blocking performance and larger revenue. In Fig. 8(c), we observe that the differences on blocking probabilities from those that consider load-balance or not (*e.g.*, DViNE and DViNE-LB) are very small, when the traffic load is relatively low. This is because the congestion due to the greedy nature of the algorithms that do not consider load-balance is not severe when the traffic load is low.

The preprocessing can make our CG-based algorithms perform worse, when the candidate SN sets get larger and the overlaps among them become more obvious. Hence, we plot the results on the blocking probability and time-average revenue for radius ρ changing from 20 to 30 in Figs. 10(b) and 10(c). In these simulations, the traffic load is fixed as 25 Erlangs. The sizes of the candidate SN sets, *i.e.*, the number of candidates for each VN, for radius ρ changing from 20 to 30 are plotted in Fig. 11. We observe that our proposed algorithms still always achieve lower blocking probability and larger time-average revenue than the four benchmarks. For our proposed algorithms, the blocking probability decreases and the time-average revenue increases with the increase of the radius, while for the benchmarks, their performance gets slightly worse when the radius increases. Note that when the radius increases, the number of candidate SNs increases, the solution space gets larger. This might make the solutions obtained by the proposed algorithms perform better. However, as the node mapping of the benchmarks suffers from the disadvantages of LP relaxation and rounding, there is no performance guarantee, it is also possible to end up with worse solutions or even infeasible ones. This explains why their performance can even get slightly worse when the radius increases.

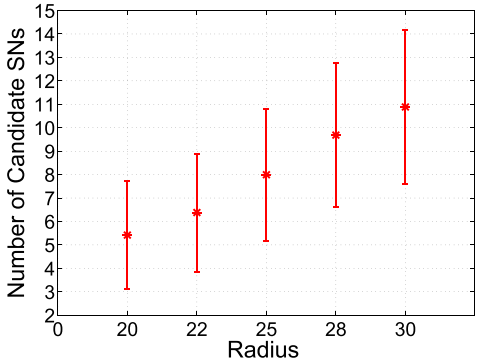


Fig. 11. Number of candidate SNs for each VN vs. radius.

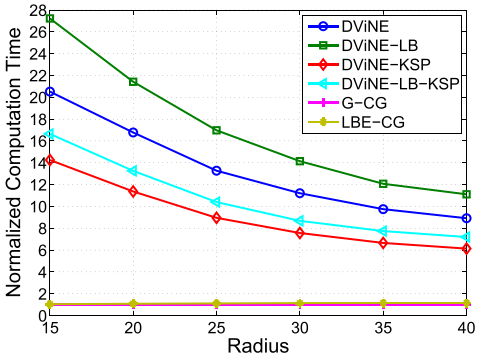


Fig. 12. Normalized computation time per VNR.

5) Time Complexity: In order to compare the time complexity of the algorithms, we use the computation time of G-CG as the basis, and summarize the results on the normalized computation time per VNR in Fig. 12. Note that, for our CG-based algorithms, we include the computation time of the pre-calculation of the K-shortest paths, the preprocessing (*i.e.*, *Algorithm* 2), the CG construction (*i.e.*, *Algorithm* 3), and the MCMC heuristics (*i.e.*, *Algorithm* 4). The simulation environment is Matlab 2012b running on a computer with 3.10 GHz Intel Core i3-2100 CPU and 4.00 GB RAM. The LPs in the benchmarks are solved with the GNU Linear Programming Kit (GLPK) package [43]. It can be seen that our proposed algorithms require much less computation time than the benchmarks, especially when the radius is relatively small. This is because the benchmarks are based on MCF and need to solve LPs, which can cause high time complexity. Moreover, as DViNE and DViNE-LB need to solve LPs twice for each VNR, they take the longest computation time.

VII. Conclusion

In this work, we focused on solving LC-VNE efficiently. Firstly, by leveraging graph bisection, we proved the $\mathcal{NP}$-completeness of LC-VNE and provided the in-approximability result of it. Then, we proposed two novel heuristics based on CG to achieve integrated node and link mapping for LC-VNE. Specifically, with CG, we reduced LC-VNE to the minimum-cost maximum clique problem. Extensive numerical simulations were conducted and the results showed that our proposed algorithms could provide smaller gaps to the optimal solutions, better blocking performance and larger time-average revenue than the existing ones, with much lower time complexity.

References

[1] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization," *Computer*, vol. 38, no. 4, pp. 34–41, Apr. 2005.

[2] P. Lu, L. Zhang, X. Liu, J. Yao, and Z. Zhu, "Highly efficient data migration and backup for big data applications in elastic optical inter-data-center networks," *IEEE Netw.*, vol. 29, no. 5, pp. 36–42, Sep./Oct. 2015.

[3] R. Mao *et al.*, "Overcoming the challenge of variety: Big data abstraction, the next evolution of data management for AAL communication systems," *IEEE Commun. Mag.*, vol. 53, no. 1, pp. 42–47, Jan. 2015.

[4] N. M. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Comput. Netw.*, vol. 54, pp. 862–876, Apr. 2010.

[5] A. Khan, A. Zugenmaier, D. Jurca, and W. Kellerer, "Network virtualization: A hypervisor for the Internet?" *IEEE Commun. Mag.*, vol. 50, no. 1, pp. 136–143, Jan. 2012.

[6] *Cisco Network Virtualization Solutions*. [Online]. Available: http://www.cisco.com/en/US/netsol/ns658/.

[7] *Hyper-V Network Virtualization Overview*. [Online]. Available: http://technet.microsoft.com/en-us/library/jj134230.aspx.

[8] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, 4th Quarter, 2013.

[9] D. Andersen, "Theoretical approaches to node assignment," Unpublished Manuscript, pp. 1–13, Dec. 2002.

[10] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 17–29, Apr. 2008.

[11] X. Cheng *et al.*, "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, pp. 38–47, Apr. 2011.

[12] Z. Wang, Y. Han, T. Lin, H. Tang, and S. Ci, "Virtual network embedding by exploiting topological information," in *Proc. IEEE GLOBECOM*, Dec. 2012, pp. 2603–2608.

[13] S. Zhang, Z. Qian, J. Wu, S. Lu, and L. Epstein, "Virtual network embedding with opportunistic resource sharing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 816–827, Mar. 2014.

[14] M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 206–219, Feb. 2012.

[15] Q. Hu, Y. Wang, and X. Cao, "Resolve the virtual network embedding problem: A column generation approach," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 410–414.

[16] L. Gong, W. Zhao, Y. Wen, and Z. Zhu, "Dynamic transparent virtual network embedding over elastic optical infrastructures," in *Proc. IEEE ICC*, Jun. 2013, pp. 3466–3470.

[17] C. Papagianni *et al.*, "On the optimal allocation of virtual resources in cloud computing networks," *IEEE Trans. Comput.*, vol. 62, no. 6, pp. 1060–1071, Jun. 2013.

[18] A. Leivadeas, C. Papagianni, and S. Papavassiliou, "Efficient resource mapping framework over networked clouds via iterated local search-based request partitioning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1077–1086, Jun. 2013.

[19] Y. Jin, Y. Wen, Q. Chen, and Z. Zhu, "An empirical investigation of the impact of server virtualization on energy efficiency for green data center," *Comput. J.*, vol. 56, pp. 977–990, Aug. 2013.

[20] L. Gong and Z. Zhu, "Virtual optical network embedding (VONE) over elastic optical networks," *J. Lightw. Technol.*, vol. 32, no. 3, pp. 450–460, Feb. 1, 2014.

[21] L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," in *Proc. IEEE INFOCOM*, Apr./May 2014, pp. 1–9.

[22] H. Jiang, L. Gong, and Z. W. Zuqing, "Efficient joint approaches for location-constrained survivable virtual network embedding," in *Proc. IEEE GLOBECOM*, Dec. 2014, pp. 1810–1815.

[23] H. Jiang, Y. Wang, L. Gong, and Z. Zhu, "Availability-aware survivable virtual network embedding in optical datacenter networks," *J. Opt. Commun. Netw.*, vol. 7, pp. 1160–1171, Dec. 2015.

[24] Z. Zhu, P. Lu, J. J. P. C. Rodrigues, and Y. Wen, "Energy-efficient wideband cable access networks in future smart cities," *IEEE Commun. Mag.*, vol. 51, no. 6, pp. 94–100, Jun. 2013.

[25] D. Zhang *et al.*, "Fine-grained localization for multiple transceiver-free objects by using RF-based technologies," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 6, pp. 1464–1475, Jun. 2014.

[26] J. Yao, P. Lu, L. Gong, and Z. Zhu, "On fast and coordinated data backup in geo-distributed optical inter-datacenter networks," *J. Lightw. Technol.*, vol. 33, no. 14, pp. 3005–3015, Jul. 15, 2015.

[27] P. Lu, Q. Sun, K. Wu, and Z. Zhu, "Distributed online hybrid cloud management for profit-driven multimedia cloud computing," *IEEE Trans. Multimedia*, vol. 17, no. 8, pp. 1297–1308, Aug. 2015.

[28] J. Zhao, S. Subramaniam, and M. Brandt-Pearce, "Virtual topology mapping in elastic optical networks," in *Proc. ICC*, Jun. 2013, pp. 3904–3908.

[29] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA, USA: Freeman, 1990.

[30] A. Belbekkouche, M. M. Hasan, and A. Karmouch, "Resource discovery and allocation in network virtualization," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 4, pp. 1114–1128, 4th Quarter, 2012.

[31] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," in *Proc. WWW*, Apr. 1998, pp. 107–117.

[32] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social Netw.*, vol. 1, no. 3, pp. 215–239, Jul. 1979.

[33] A. Schrijver, *Theory of Linear and Integer Programming*. New York, NY, USA: Wiley, 1986.

[34] S. Ma *et al.*, "Demonstration of online spectrum defragmentation enabled by OpenFlow in software-defined elastic optical networks," in *Proc. OFC*, Mar. 2014, pp. 1–3.

[35] D. B. West, *Introduction to Graph Theory*, 2nd ed. New York, NY, USA: Pearson Education Inc., 2001.

[36] D.-Z. Du, K.-I. Ko, and X. Hu, *Design and Analysis of Approximation Algorithms*. New York, NY, USA: Springer, Nov. 2011.

[37] S. Arora and C. Lund, "Approximation algorithms for NP-hard problems," in *Hardness of Approximations*. Boston, MA, USA: PWS-Kent, 1997, pp. 399–446.

[38] M. Brown and R. Tarjan, "A fast merging algorithm," *J. ACM*, vol. 26, pp. 211–226, Apr. 1979.

[39] J. Hastad, "Clique is hard to approximate within $n^{1-\epsilon}$," in *Proc. 37th FOCS*, Oct. 1996, pp. 627–636.

[40] R. A. Brualdi, *Introductory Combinatorics*, 5th ed. New York, NY, USA: Pearson Education Inc., 2009.

[41] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. INFOCOM*, Mar. 1996, pp. 594–602.

[42] J. Y. Yen, "Finding the K shortest loopless paths in a network," *Manage. Sci.*, vol. 17, pp. 712–716, Jul. 1971.

[43] *GNU Linear Programming Kit (GLPK) Package*. [Online]. Available: http://www.gnu.org/software/glpk/.

Long Gong, photograph and biography not available at the time of publication.

Huihui Jiang, photograph and biography not available at the time of publication.

Yixiang Wang, photograph and biography not available at the time of publication.

Zuqing Zhu, photograph and biography not available at the time of publication.