A Novel Optimal Mapping Algorithm with Less Computational Complexity for Virtual Network Embedding

Haotong Cao, *Student Member, IEEE*, Yongxu Zhu, Gan Zheng, *Senior Member, IEEE*, and Longxiang Yang

***Abstract*—Network Virtualization (NV) is widely accepted as one enabling technology for future network, which enables multiple Virtual Networks (VNs) with different paradigms and protocols to coexist on the shared Substrate Network (SN). One key challenge in network virtualization is Virtual Network Embedding (VNE), which maps a virtual network onto the shared SN. Since VNE is NP-hard, existing efforts mainly focus on proposing heuristic algorithms that try to achieve feasible VN embedding in reasonable time, consequently the resulted embedding is not optimal. To tackle this difficulty, we propose a candidate assisted (CAN-A) optimal VNE algorithm with lower computational complexity. The key idea of the CAN-A algorithm lies in constructing the candidate substrate node subset and the candidate substrate path subset before embedding. This reduces the mapping execution time substantially without performance loss. In the following embedding, four types of node and link constraints are considered in the CAN-A algorithm, making it more applicable to realistic networks. Simulation results show that the execution time of CAN-A is hugely cut down compared with pure VNE-MIP algorithm. CAN-A also outperforms the typical heuristic algorithms in terms of other performance indices, such as the average virtual network request (VNR) acceptance ratio and the average virtual link propagation delay.**

***Index Terms*—Network virtualization, virtual network embedding, NP-hard, heuristic algorithm, execution time, candidate subset, virtual network request acceptance ratio, virtual link propagation delay.**

I. Introduction

RECENTLY, network virtualization has been recognized as one of the key technologies for the future network [1]-[5]. Network virtualization is widely accepted as the right candidate to overcome the ossification of current Internet to fundamental changes.

Though a great interest in network virtualization has been stimulated, several challenges still exist and prevent network virtualization from being implemented in real networking environment [4]. One main challenge is how to effectively and efficiently map multiple virtual networks (VNs), with individual virtual node and link requests, onto the shared substrate networks with limited node and link resources. This mapping challenge is known as virtual network embedding (VNE). Since VNE requires a simultaneous optimal assignment of virtual nodes and links of each VN, it is complex in computation and known as a NP-hard problem [8]. Multiple VNE algorithms have been proposed in the literature [9]-[25]. However, for simplifying the VNE, these algorithms either assume that the shared SN supports the path splitting [26][27], or conduct the node and link embedding in two separate stages. Consequently, the mapping results of these algorithms are not optimal.

This paper presents an optimal mapping algorithm with huge reduction of the algorithm execution time for virtual network embedding. The proposed algorithm is named as CAN-A (Candidate Assisted). Compared with existing non-optimal heuristic algorithms, CAN-A algorithm insures the optimal mapping solution for any given virtual network request. At the same time, the computational complexity of CAN-A is much lower compared with previous optimal VNE algorithms. The key idea of CAN-A algorithm lies in constructing the candidate substrate node subset and the candidate substrate path subset before conducting the integer linear programming based mapping. In CAN-A algorithm, four types of node and link constraints are considered, which are deduced from realistic network attributes, and make CAN-A applicable to real VN operation. Those constraints are node capacity, node location, link bandwidth, and link propagation delay [28][29][30]. Virtual node location and virtual link propagation delay are not considered as constraints in previous VNE algorithms. Extensive simulation results show that the execution time of CAN-A is hugely cut down, compared with the optimal VNE-MIP algorithm [14]. In addition, our CAN-A algorithm outperforms the selected typical heuristic algorithms in terms of other performance metrics such as average VNR acceptance ratio and average virtual link propagation delay.

Main contributions of this paper are summarized as follows:

1) A novel VNE algorithm CAN-A is proposed, which is optimal and less computationally complex. The reduction of computational complexity is achieved by constructing

Haotong Cao and Longxiang Yang are with the Key Laboratory of Broadband Wireless Communication and Sensor Network Technique, Ministry of Education, Nanjing University of Posts and Telecommunications, Nanjing 210003, China. (e-mail: 1015010309@njupt.edu.cn, yanglx@njupt.edu.cn).

Yongxu Zhu and Gan Zheng are with the Wolfson School of Mechanical, Electrical and Manufacturing Engineering, Loughborough University, Leicestershire LE11 3TU, U.K. (e-mail: y.zhu4@lboro.ac.uk, g.zheng@lboro.ac.uk).

Corresponding Author: Longxiang Yang.

two substrate candidate subsets, one for substrate nodes, the other one for substrate paths. The volume of each subset is much smaller than the corresponding full set. Afterwards, integer linear programming based approach is applied to calculate the optimal mapping within the reduced subsets.

2) The virtual node locations and the virtual link propagation delays are formulated as constraints in the CAN-A algorithm, which are not considered as constraints in previous papers. With the increase of delay sensitive services [28][29] on the network, strict and guaranteed transmission delay is needed for this kind of new services. We formulate the link propagation delay as a constraint into CAN-A algorithm, so as to the mapped virtual network provides delay guaranteed services to meet future service requirements. Location constraint is a natural attribute of network node. Location constraint in VNE algorithm makes VNE applicable to engineering networks.
3) Four sub-algorithms of CAN-A are proposed and evaluated in our paper. Each sub-algorithm adopts an objective function to optimize specific performance index. Four objectives are: total substrate cost function **CF**, total virtual link propagation delay function **LDF**, substrate cost and load balancing function **CLBF**, substrate cost and virtual link propagation delay function **CLDF**, respectively.
4) Extensive simulations are conducted in our paper. The computational complexity of CAN-A is compared with the pure mixed integer programming (MIP) algorithm, which is widely accepted as the optimal VNE algorithm. The simulation results indicate that the complexity of CAN-A is reduced hugely compared with VNE-MIP algorithm. In addition, simulation results validate that average VNR acceptance ratio and average virtual link propagation delay of CAN-A outperform typical heuristic algorithms.

The rest of this paper is organized as follows. Section II summarizes the related work. In Section III, virtual network embedding problem formulation, notations, embedding process and performance metrics are presented. Section IV details the CAN-A algorithm. Simulations of CAN-A against typical heuristic algorithms and pure MIP algorithm are illustrated in Section V. Finally, conclusion and future work are described in Section VI.

II. Related Work

VNE, formulated as an unsplittable flow problem [27], aims to realize optimal node and link mapping of each given VN simultaneously. Numerous VNE algorithms have been proposed in the literature. This section only discusses VNE algorithms that are closely related to the CAN-A algorithm. Any reader who has interest in other issues of VNE, such as service level assignment (SLA) [23] and VN survivability [24][33][34], can refer to references [6] and [7] for detailed surveys.

In addition, we discuss the CAN-A algorithm and carry out the simulation work in the fundamental VNE network scenario (one SN and several given VNRs in a discrete time event). Other complex network scenarios (e.g. occasional substrate node / link failure [24][33][34], multiple shared SNs [37][38]) can be extended on the basis of fundamental VNE network scenario.

A. Typical Heuristic VNE Algorithms

In the early stage of VNE research, most proposed VNE algorithms are heuristic [6]，compromising global optimality for short embedding execution time. This *subsection* introduces typical heuristic algorithms in the literature.

Zhu *et al.* [9] proposes a one-stage VNE heuristic algorithm. The goal of this heuristic algorithm is to achieve low load on substrate nodes and links, with the assumption of infinite substrate resources and no constraints of virtual node location. Assumptions made in this algorithm are not applicable in real networking environment. Therefore, this heuristic algorithm [9] is not applicable.

In reference [10], a typical heuristic algorithm, proposed by Yu *et al.*, applies the greedy method to implement the virtual node mapping in the first step. Dijkstra's algorithm [32] and multi-commodity flow approaches [27] are used to deal with the link mapping in the second step. Yu *et al.* also introduces the concept of path splitting to accept more proposed VNRs and relieve substrate link loading. However, this algorithm leads to a level of resource fragmentation that is unfeasible for accepting VNRs with large sizes. Only constraints of node capacity and link bandwidth are considered. In addition, there is no restriction of the VNE solution space. If the size of SN increases, computational complexity increases exponentially.

A backtracking algorithm based on sub-graph isomorphism is proposed in reference [11]. Reference [11] manages to online map virtual nodes and links per VN, taking constraints of node capacity and link bandwidth into consideration. Other node and link constraints, such as node location requirement, are not considered. The computation complexity exponentially grows with the size of VNR extending. In addition, the scale of substrate network is limited. The execution time grows exponentially with the substrate network scale expanding. Therefore, this backtracking algorithm is not applicable to real VNE implementation.

Cheng *et al*. [12] proposes a topology-aware node mapping approach that uses the *Markov Random Walk model* to measure the node capacity and joint link bandwidth. The virtual links, no matter splittable or unsplittable, are then mapped to substrate paths either using k-shortest path or multi-commodity flow method. However, reference [12] will lead to the problem of resource fragmentation in the long run, same to ref. [10].

The algorithm proposed in [14], based on the previous conference version [13], is widely accepted as a representative heuristic algorithm in VNE research area. Chowdhury *et al.* firstly models the VNE problem by applying the mixed integer programming (MIP) model of the optimization theory [31]. Due to the complexity of using pure MIP to solve a medium-sized network directly, Chowdhury *et al.* has to modify the MIP model into a linear programming (LP) model and relax the integer constraints. Chowdhury *et al.* uses the LP model to complete the virtual node mapping. If the solution of

LP model is feasible, the virtual links are assigned to corresponding substrate paths by using Dijkstra's algorithm [32] or multi-commodity flow [27] approach. Though Chowdhury *et al.* coordinates node and link mapping, the algorithm in [14] is not an efficient and effective VNE algorithm, in terms of average VNR acceptance ratio in the long term.

Mijumbi *et al*. [20], developed from [21] and [22], is another variation of [14] to solve the VNE problem. Node and link constraints considered are node capacity and link bandwidth. In the first step, [20] formulates the VNE problem by using the pure MIP model [14]. Relaxing the binary variables to take on continuous values, the MIP model is relaxed into a LP model. With solving the LP model and getting the node mapping results, the shortest path is selected again [32]. The authors then derive the corresponding dual MIP model. The dual model is used to ensure the selected paths legitimate. The VN assignment which consumes less substrate resources is preferred. The time complexity decreases a lot, compared with solving the pure MIP model directly. However, the VNR acceptance ratio of this heuristic algorithm can not behave well in the long term, compared with the heuristics [12][14].

B. Representative Exact VNE Algorithms

With the rapid development of VNE algorithms, some researchers also have turned to the exact algorithm research [7] to find the optimal embedding of each given VNR. This *subsection* discusses some representative exact VNE algorithms in the literature. Software tools (GLPK [35], CPLEX [36]) and suitable model (restricted ILP model) are commonly used in the exact VNE algorithms.

Houdi *et al.* [15] proposes an exact algorithm to map one VN across multiple SNs, fighting for the minimization of VN provision cost. However, the cost metric is not defined clearly in [15]. The authors use the *max-flow and min-cut method* to split one VN into several sub-VNs firstly. The number of sub-VNs is equal to the number of SNs. Then the LP model is adopted to map each part of the VN to each corresponding SN. Though splitting a VN reduces the complexity, [15] restricts the solution space and can not find the optimal embedding of the VN in the end. The topology and resource information of all substrate networks, often unknown in practical networking environment, are assumed to be available in [15]. Therefore, this algorithm is not applicable to the real networking implementation.

Betero *et al.* [16], a variation of reference [14], adds the constraint of allowing more than one virtual node per VN to map onto the same substrate node and uses the MIP model [14] to deal with VNE problem directly. The goals of [16] are to minimize the consumption of link bandwidth and the number of active substrate nodes. This paper is also the first attempt to consider energy awareness in VNE area. Only node capacity and link bandwidth are taken into account in reference [16]. Due to the computational complexity of pure MIP model, the sizes of SN and VNRs are set very small. Therefore, exact algorithm proposed in ref. [16] can not be applied to real VNE implementation. On the basis of reference [16], Su Sen *et al.* [17][18] develops VNE energy aware algorithm.

While in reference [19], presented by Melo *et al.*, a pure ILP algorithm is proposed to solve VNE problem. The aims of [19] are to minimize the VNR mapping cost and achieve substrate load balancing. Two node and link constraints, node location requirement and virtual link propagation delay, are mentioned, but not analyzed. Node location constraint and virtual link propagation delay constraint are not conducted in the simulation work, either. Generally speaking, reference [19] is an exact algorithm and fights for the substrate load balancing, only considering node capacity and link bandwidth constraints. The computation complexity of pure ILP algorithm [19] is very high. Therefore, the pure ILP algorithm can not be applied in the real networking environment.

C. Summary

To summarize, because VNE problem is NP-hard, previous VNE heuristic algorithms make some unrealistic assumptions for simplification, such as VNRs known in advance, infinite substrate resources and ignoring virtual node or link constraint. In addition, most heuristic algorithms solve the VNE problem in two separate stages, using greedy method or LP model in node mapping stage and optimizing the link mapping by Dijkstra's algorithm [32] or multi-commodity flow [27] approach. For exact VNE algorithms, most of them use pure MIP / ILP model and aim to embed proposed VNRs as many as possible, thus leading to large algorithm execution time. In general, only node capacity and link bandwidth constraints are considered in previous VNE research. Other node and link constraints are not considered and formulated.

The CAN-A proposed in this paper differs from previous VNE algorithms mainly in four aspects. First of all, the CAN-A algorithm is a combination of heuristic (candidate subset construction) and exact (integer linear programming based approach) algorithm. The heuristic part contributes to selecting candidate substrate nodes and paths ahead. Therefore, the number of binary variables, which are used in integer linear programming based approach, will be largely cut down. This leads to the huge reduction of algorithm execution time. The optimal VN embedding will be found among the restricted solution space. Secondly, the CAN-A conducts node and link mapping of the given VNR in one stage. The node mapping is an important stage in VNE since it determines the efficiency of the following link mapping. Coordinating both stages will lead to better resource utilization [14]. CAN-A combines both stages into one stage which will further enhance the substrate network resource efficiency. Next, apart from general node and link constraints (node capacity and link bandwidth), virtual node location and virtual link propagation delay are incorporated as node and link constraints in CAN-A. Particularly, the virtual link propagation delay is formulated as node-link constraint for the first time in VNE algorithm. With the increasing of delay sensitive services on the network, guaranteed transmission delay is needed. The link propagation delay constraint in CAN-A aims to offer delay guaranteed VNR to meet future instant service requirements [30]. Finally, authors of this paper make a comprehensive simulation work comparing with typical heuristic algorithms and the exact MIP

algorithm. The efficiency and effectiveness of the CAN-A algorithm are validated.

III. Network Model and VNE Problem Description

This section elaborates on the VNE network model and VNE problem description. The substrate network and virtual network request models are described firstly. The main index notations throughout this paper are listed in Table I. The measurements of network resources are introduced in the following *subsection*. Afterwards, the embedding process of VNR is presented. For a better understanding of the VNR embedding process in the VNE system, a diagram is also plotted. Finally, several important VNE performance metrics are defined.

A. Network Model Description

1) *Substrate network*: The substrate network is modeled as a weighted undirected graph $G^S=(N^S, L^S)$, where N^S is the set of all substrate nodes and L^S is the set of all substrate links. Each substrate node $\boldsymbol{m} \in N^S$ is characterized by its node capacity, $\mathbf{C}^S_m$, and the node location, loc($\boldsymbol{m}$). The location of substrate node $\boldsymbol{m}$, loc($\boldsymbol{m}$), is defined on $\boldsymbol{x}$ and $\boldsymbol{y}$ coordinates in this paper. With respect to substrate links, each substrate link $\boldsymbol{mn}$ has a finite bandwidth B^S_{mn} and a predefined link propagation delay $\mathbf{D}^S_{mn}$. Due to the nature of undirected graph of SN, the value of $\mathbf{B}^S_{mn}$ is equal to $\mathbf{B}^S_{nm}$. In addition, the set of all substrate paths in the SN is denoted by notation $\mathbf{P}^S$. The $\mathbf{P}^S_{mn}$ is the set of all paths between end nodes $\boldsymbol{m}$ and $\boldsymbol{n}$ without any loop. $\mathbf{P}_{mn}$ is one path selected from the $\boldsymbol{mn}$ path set, $\mathbf{P}^S_{mn}$. In this paper, each single substrate link is assumed to have one unit time link propagation delay. That is to say, the link propagation delay of one selected path $\mathbf{D}_{mn}$ can be seen as the number of substrate links in the substrate path $\boldsymbol{mn}$.

The right part of Fig. 1 shows a substrate network. The numbers on the links are available link bandwidth, and numbers in rectangles are available node capacity. For simplicity, the location of each substrate node is omitted in this figure.

2) *Virtual network request:* A virtual network request is modeled as a weighted undirected graph $G^V=(N^V, L^V)$, where N^V is the set of all virtual nodes of the VNR, and L^V is the set of all virtual links of the VNR. Each virtual node $\boldsymbol{M}$ is characterized by the required node capacity, $\mathbf{C}^V_M$, and the desired virtual node location, Loc($\boldsymbol{M}$). The maximum allowed location deviation of virtual node $\boldsymbol{M}$ is LR($\boldsymbol{M}$). Expression 1 shows the location deviation relationship between virtual node $\boldsymbol{M}$ and its candidate substrate node $\boldsymbol{m}$. The location deviation must be within the radius LR($\boldsymbol{M}$) of $\boldsymbol{M}$. With respect to each virtual link $\boldsymbol{MN}$, it has a required bandwidth $\mathbf{B}^V_{MN}$ and maximum allowed link propagation delay $\mathbf{D}^V_{MN}$.

The left part of Fig. 1 presents two virtual network requests. The numbers over the links are required link bandwidth. The numbers in rectangles are node capacity demands of virtual nodes. The location requirement of any virtual node is set as a pair of real numbers on $\boldsymbol{x}$ and $\boldsymbol{y}$ coordinates, along with its required location deviation LR($\boldsymbol{M}$).

$$\mathrm{Dis}(\mathrm{loc}(\boldsymbol{m}), \mathrm{Loc}(\boldsymbol{M})) \le \mathrm{LR}(\boldsymbol{M}) \quad (1)$$

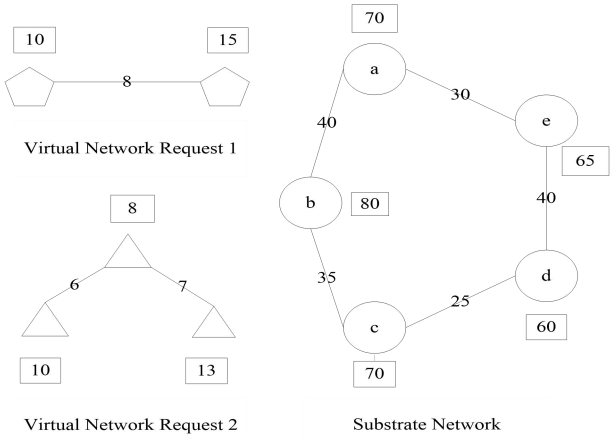

Fig. 1. One Substrate Network and Two Virtual Network Requests

3) *VNE Notations:* The main notations used throughout this paper are listed in Table I.

Table I
VNE Notations

$G(N^S,L^S)$	Substrate Network
$G(N^V,L^V)$	Virtual Network
$\boldsymbol{m}$, $\boldsymbol{n}$	Substrate Nodes
$\boldsymbol{mn}$, $\boldsymbol{nm}$	Substrate Links
$\boldsymbol{M}$, $\boldsymbol{N}$	Virtual Nodes
$\boldsymbol{MN}$, $\boldsymbol{NM}$	Virtual Links
$\mathbf{P}^S_{mn}$	Set of Paths that start from source node $\boldsymbol{m}$ to end node $\boldsymbol{n}$
$\mathbf{P}_{mn}$	One Selected Path from the path set $\mathbf{P}^S_{mn}$
$\mathbf{C}^S_m$	Node Capacity of Substrate Node $\boldsymbol{m}$
$\mathbf{C}^V_M$	Node Capacity Requirement of Virtual Node $\boldsymbol{M}$
loc($\boldsymbol{m}$)	Location of Substrate Node $\boldsymbol{m}$ on $\boldsymbol{x}$ and $\boldsymbol{y}$ coordinates
Loc($\boldsymbol{M}$)	Location Requirement of Virtual Node $\boldsymbol{M}$ on $\boldsymbol{x}$ and $\boldsymbol{y}$ coordinates
$\mathbf{B}^S_{mn}$	Link Bandwidth of the Substrate Link $\boldsymbol{mn}$
$\mathbf{B}_{mn}$	Link Bandwidth of one selected Substrate Path $\boldsymbol{mn}$
$\mathbf{B}^V_{MN}$	Link Bandwidth Requirement of the Virtual Link $\boldsymbol{MN}$
$\mathbf{D}^S_{mn}$	Link Propagation Delay of the Substrate Link $\boldsymbol{mn}$
$\mathbf{D}_{mn}$	Link Propagation Delay of one Selected Substrate Path $\boldsymbol{mn}$ from the path set $\mathbf{P}^S_{mn}$
$\mathbf{D}^V_{MN}$	Link Propagation Delay Requirement of Virtual Link $\boldsymbol{MN}$

B. Measurements of Network Resources

In VNE, to quantify the resource usage of substrate nodes, the remaining or available node capacity of a substrate node $\mathbf{R}^{S}(\boldsymbol{m})$ is defined as follows:

$$\mathbf{R}^{S}(\boldsymbol{m}) = \mathbf{C}^{S}_{m} - \sum_{M \uparrow m} \mathbf{C}^{V}_{M} \tag{2}$$

where $\boldsymbol{M}\uparrow \boldsymbol{m}$ denotes that the virtual node $\boldsymbol{M}$ is mapped onto the substrate node $\boldsymbol{m}$.

Similarly, the available bandwidth of each substrate link is determined by the bandwidth before the VN mapping and the consumed bandwidth after the VNE assignment. Expression 3 below shows the definition of available link bandwidth.

$$\mathbf{R}^{S}(\boldsymbol{mn}) = \mathbf{B}^{S}_{mn} - \sum_{MN \uparrow mn} \mathbf{B}^{V}_{MN} \tag{3}$$

where $\boldsymbol{MN}\uparrow\boldsymbol{mn}$ denotes that virtual link $\boldsymbol{MN}$ passes through the substrate link $\boldsymbol{mn}$ after conducting the embedding of the VNR.

As known in VNE, every virtual link per VNR may occupy one or more substrate paths after two end nodes of the virtual link are mapped. Though no link aggregation assumption (splittable flow) in this paper, one substrate path may accommodate one or more substrate links. On these basis, the available bandwidth of one substrate path $\boldsymbol{mn}$ is given by follows:

$$\mathbf{R}^{S}(\mathbf{P}_{mn}) = \min_{ab \in \mathbf{P}_{mn}} \mathbf{R}^{S}(\boldsymbol{ab}) \tag{4}$$

where one selected single substrate path $\boldsymbol{mn}$ is made up of one or more substrate links, the substrate link $\boldsymbol{ab}$ included.

C. Embedding Process of VNR

When a VNR arrives, the SN will firstly determine whether to accept the VNR or not, according to the available network resources of the SN. If the VNR is accepted, the SN will refer to one specific VNE algorithm for embedding the VNR and spare enough substrate resources to fulfill all resource requirements of the VNR. When the VNR expires, the allocated resources are returned to SN again. For a better understanding of VN embedding, the embedding process of one VNR is depicted in Fig. 2 on the right.

In general, the embedding of a VNR onto the shared SN is made up of two components: the component that deals with the mapping of virtual nodes, and the component that ensures the mapping of virtual links.

1) *Virtual Node Embedding*: Each virtual node of the same VNR must be assigned to a different substrate node of SN. The assignments of all virtual nodes are determined by a node-mapping function $\boldsymbol{M}_{N}(\,)$: $N^{V} \rightarrow N^{S}$.

$$\boldsymbol{M}_{N}(\boldsymbol{M}) \in N^{S}$$

$$\boldsymbol{M}_{N}(\boldsymbol{M}) = \boldsymbol{M}_{N}(\boldsymbol{N}), \text{if and only if } \boldsymbol{M}=\boldsymbol{N}$$

subject to

$$\mathbf{C}^{V}_{M} \le \mathbf{R}^{S}(\boldsymbol{M}_{N}(\boldsymbol{M})) \tag{5}$$

$$\mathrm{Dis}(\mathrm{loc}(\boldsymbol{M}_{N}(\boldsymbol{M})), \mathrm{Loc}(\boldsymbol{M})) \le \mathrm{LR}(\boldsymbol{M}) \tag{6}$$

where expression 5 aims to ensure that the node capacity requirement of any virtual node $\boldsymbol{M}$ must not exceed the remaining capacity of the selected substrate node $\boldsymbol{M}_{N}(\boldsymbol{M})$ that accommodates virtual node $\boldsymbol{M}$; expression 6 aims to ensure that the deviation between any virtual node $\boldsymbol{M}$ and the selected substrate node $\boldsymbol{M}_{N}(\boldsymbol{M})$ must not exceed the radius $\mathrm{LR}(\boldsymbol{M})$ of virtual node $\boldsymbol{M}$. During the node embedding component, both expressions must be fulfilled at the same time.

2) *Virtual Link Embedding*: In this paper, each virtual link of the same VNR is mapped onto a single substrate path (unsplittable flow) between two substrate nodes. Both substrate nodes have hosted two virtual nodes of the virtual link. This guarantees no interference among virtual links. The link embedding is implemented according to the link-mapping function $\boldsymbol{M}_{L}(\,)$: $L^{V} \rightarrow P^{S}$ for all virtual links of the VNR.

$$\boldsymbol{M}_{L}(\boldsymbol{M}, \boldsymbol{N}) \subseteq P^{S}(\boldsymbol{M}_{N}(\boldsymbol{M}), \boldsymbol{M}_{N}(\boldsymbol{N}))$$

subject to

$$\mathbf{B}^{V}_{MN} \le \mathbf{R}^{S}(\mathrm{P}_{\boldsymbol{M}_{N}(\boldsymbol{M})\boldsymbol{M}_{N}(\boldsymbol{N})}) \tag{7}$$

$$\mathbf{D}_{\boldsymbol{M}_{N}(\boldsymbol{M})\boldsymbol{M}_{N}(\boldsymbol{N})} \le \mathbf{D}^{V}_{MN} \tag{8}$$

where expression 7 is to ensure that the link bandwidth demand of any virtual link $\boldsymbol{MN}$ must not exceed the available bandwidth of the selected substrate path that accommodates virtual link $\boldsymbol{MN}$; expression 8 is to ensure that the propagation delay of the selected substrate path $\boldsymbol{M}_{N}(\boldsymbol{M})\boldsymbol{M}_{N}(\boldsymbol{N})$ must not violate the required virtual link $\boldsymbol{MN}$ propagation delay $\mathbf{D}^{V}_{MN}$. During the link embedding component, expression (7) and expression (8) must be fulfilled simultaneously.

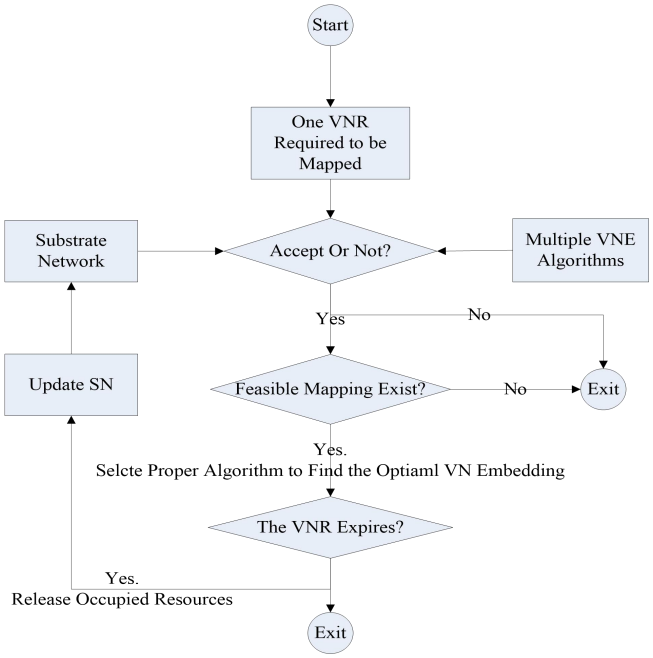

Fig. 2. The Diagram of One VNR Embedding

D. Performance Metrics

In order to assess the performance of any VNE algorithm and compare with other VNE algorithms, performance metrics are necessary to be defined. This *subsection* introduces some important performance metrics, commonly used in VNE.

1) *The revenue of a VNR*: Similar to the definition in [13] and [14], the revenue of a VNR is defined as :

$$\boldsymbol{Rev}(G^{V}) = \alpha \bullet \sum_{M \in N^{V}} \mathbf{C}^{V}_{M} + \beta \bullet \sum_{M \in L^{V}} \mathbf{B}^{V}_{MN} \tag{9}$$

As derived from the expression 9, the revenue of a VNR is set to be the sum of all virtual node capacity and virtual link

bandwidth of this VNR [6]. Weight factors (α and β) are used to balance different types of network resources.

2) *The cost of a VNR*: Expression 9 gives an insight into how much revenue can be earned by accepting a VNR. In order to evaluate how much substrate resources consumed for a VNR, expression 10 formulates the cost of an embedded VNR:

$$Cos(R^V)=\gamma \bullet \sum_{M\in N^V} C^V_M + \eta \bullet \sum_{MN\in L^V} \sum_{mn\in P^S_{mn}} D_{mn} \bullet B^V_{MN} \quad (10)$$

As assumed before, all single substrate links has the same propagation delay (one time unit). Therefore, the propagation delay of a substrate path indicates the number of substrate links in this path. The cost of substrate links is defined as the sum of the product of path propagation delay and consumed virtual link bandwidth.

3) *VNR acceptance ratio*: The VNR acceptance ratio, shown in expression 11 below, is used to evaluate an algorithm's ability of batch processing VNRs. VNR acceptance ratio is an important metric for the performance evaluation of VNE algorithms. It is determined by the number of successfully mapped VNRs **a'** and the number of proposed VNRs **a** in a timetable:

$$A^{VN} = \mathbf{a}'/\mathbf{a} \quad (11)$$

Some other metrics, such as average node / link utilization, average virtual link propagation delay, and execution time, can be found in reference [6] and [7]. Because of limited pages, details of these metrics are not discussed here.

IV. Candidate-Assisted (CAN-A) Algorithm for VNE

This section presents the CAN-A algorithm with huge reduction of mapping execution time, meanwhile without loss of the main performance indices. The candidate node subset and candidate path subset constructing approaches are discussed at first. Then, the variables and constraints of CAN-A algorithm are formulated. At last, four different objective functions of CAN-A algorithm are discussed. Each objective function represents one sub-algorithm of CAN-A algorithm. Notations and metrics used in this section are same to what are defined in Section III.

A. Candidate Subset Construction

Candidate subset construction aims to select candidate substrate nodes and candidate substrate paths for a given VNR. This process intends to reduce the execution time of the following embedding (Section IV-B&C&D), which is based on integer linear programming approach.

Every time a VNR comes, the location constraints of all virtual nodes are considered at first. The required location of one virtual node M is defined by x and y coordinates, (X_M, Y_M). In this phase, the virtual node M is mapped onto one substrate node m, (X_m, Y_m), within the allowed location deviation LR(M). Expressions 12 and 13, the detailed version of expression 6, aim to ensure that ***Euclidean Distance*** between M and m must not exceed the allowed location deviation LR(M) of virtual node M. All substrate nodes that are within the allowed location deviation LR(M) of virtual node M make up the candidate substrate node set of virtual node M. To sum up, the number of candidate substrate node sets is equal to the number of virtual nodes in the given VNR. To deal with the occasional situation where two different virtual nodes (M and N) have the same candidate substrate node, m, within their allowed location deviations. The smaller allowed location deviation of the virtual node, M or N, will prefer the substrate node m as its candidate substrate node.

$$\begin{aligned} &\mathrm{Dis}(\mathrm{loc}(M_N(M)), \mathrm{Loc}(M)) \\ &\quad = \sqrt{(X_M - X_m)^2 + (Y_M - Y_m)^2} \end{aligned} \quad (12)$$

$$\sqrt{(X_M - X_m)^2 + (Y_M - Y_m)^2} \le LR(M) \quad (13)$$

After candidate substrate node sets of all virtual nodes are constructed, the variable k, related to virtual link propagation delay, is assisted to construct the candidate substrate path sets of all virtual links.

To each virtual link of the VNR, the concrete value of variable k is different. For example, the value k of one virtual link MN is set to be 3. That is to say, the propagation delay of one selected substrate path (without any loop), starting from one candidate substrate node of the virtual node M to one candidate substrate node of the virtual node N, must not exceed 3. All the selected substrate paths, with their propagation delays no exceeding 3, make up the candidate substrate path set of the virtual link MN. To other remaining virtual links of the VNR, the corresponding candidate path sets are constructed, based on the required values of variable k. The reason for different virtual links having different values of k is that any two end nodes in the same VNR may have different link propagation delay demands in real networking environment. Overall, the value of variable k for each virtual link of the VNR is enable to assist to select candidate substrate paths whose total propagation delay is within its corresponding required virtual link propagation delay. That is to say, the procedure of constructing candidate substrate path sets assure the candidate substrate paths within the required virtual link propagation delay ahead. The following integer linear programming approach enables to find the optimal VNR embedding from the candidate substrate node and path sets. For a better understanding, the pseudo code of **Candidate Subset Construction** is shown below.

Candidate Subset Construction

Input: One given VNR and the updated SN
Output: Candidate Substrate Nodes Subset and Candidate Substrate Paths Subset of VNR for the next mapping

1. ***while*** there is unprocessed virtual nodes and virtual links ***do***
2. Picking out unprocessed virtual nodes and virtual links
3. ***for*** each virtual node of the VNR ***do***
4. select the corresponding candidate substrate nodes of each virtual node (VNR), according to Expressions 12 and 13
5. ***end for***
6. ***for*** each virtual link MN in the VNR
8. Select the candidate substrate paths of the virtual link MN, according to the corresponding value of variable k (k represents the required virtual link propagation delay of virtual link MN). Different virtual links have different k values
9. ***end for***
10. ***end while***

After completing the **Candidate Subset Construction** process, candidate substrate node sets and candidate path sets of the given VNR have been constructed. In the second phase, we formulate the mapping / embedding algorithm via using the integer linear programming approach. The constructed candidate node subsets and candidate path subsets are much smaller than the whole node sets and the whole path sets. This will lead to huge reduction of mapping execution time in the second phase obviously.

B. Variables

The binary variable $\boldsymbol{x}$ is used for the assignment of any virtual node $\boldsymbol{M}$ to any substrate node $\boldsymbol{m}$. $\boldsymbol{x}$ is defined in expression 14. With respect to the virtual links per VN, the binary variable $\boldsymbol{y}$ is defined in expression 15.

1) *Virtual Node Variable*:

$$\boldsymbol{x}_M^m = \begin{cases} 1, & \textit{virtual node } \boldsymbol{M} \textit{ is mapped onto substrate node } \boldsymbol{m} \\ 0, & \textit{else} \end{cases} \quad (14)$$

2) *Virtual Link Variable*:

$$\boldsymbol{y}_{MN}^{mn} = \begin{cases} 1, & \textit{virtual link } \boldsymbol{MN} \textit{ is mapped onto one subtrate path } \boldsymbol{mn} \\ 0, & \textit{else} \end{cases} \quad (15)$$

C. Constraints

To ensure the correct embedding of virtual nodes and links and obey to the conservation law of substrate nodes and links, concerning constraints are formulated and presented below.

1) *Virtual Nodes' Mapping to Substrate Nodes*: Expression 16 below aims to assure that once any virtual node $\boldsymbol{M}$ is to be mapped, only one substrate node $\boldsymbol{m}$ is assigned to that virtual node $\boldsymbol{M}$.

$$\forall \boldsymbol{M} \in \boldsymbol{N}^V, \sum_{\boldsymbol{m}} \boldsymbol{x}_M^m = 1 \quad (16)$$

2) *Substrate Nodes' Mapping to Virtual Nodes*: Expression 17 is to ensure that each substrate node in the SN can accommodate no more than one virtual node $\boldsymbol{M}$. Expressions 16 and 17 vividly describe the mapping relationship between virtual node $\boldsymbol{M}$ and substrate node $\boldsymbol{m}$ in VNE.

$$\forall \boldsymbol{m} \in \boldsymbol{N}^S, \sum_{\boldsymbol{M}} \boldsymbol{x}_M^m \le 1 \quad (17)$$

3) *Conservation of Node Capacity*: Expression 18 aims to ensure that each selected substrate node $\boldsymbol{m}$ must have reserved available node capacity for accommodating the node capacity demand of virtual node $\boldsymbol{M}$ that maps onto $\boldsymbol{m}$.

$$\forall \boldsymbol{m} \in \boldsymbol{N}^S, \sum_{\boldsymbol{M}} \boldsymbol{x}_M^m \bullet \boldsymbol{C}_M^V \le \boldsymbol{C}_m^S \quad (18)$$

4) *Assignment of Virtual Links to Substrate Paths*: Unsplittable flow constraint [27] is applied in our paper. Each virtual link $\boldsymbol{MN}$ is allocated to one selected substrate path with fulfilling the node requirements of two end nodes. Due to the goal of achieving the minimization of average virtual link propagation delay, it is essential to adopt to the unsplittable flow constraint. Multi-commodity flow constraint will contribute to the increase of virtual link propagation delay. Simulation in the Section V proves this issue true. Expression 19 shows the unsplittable flow constraint of virtual links.

$$\forall \boldsymbol{MN} \in \boldsymbol{L}^V, \exists \boldsymbol{mn} \in \boldsymbol{P}_{mn}^S, \sum_{\boldsymbol{mn}} (\boldsymbol{y}_{MN}^{mn} - \boldsymbol{y}_{NM}^{mn}) = \boldsymbol{x}_M^m - \boldsymbol{x}_M^n \quad (19)$$

5) *Link Bandwidth Conservation*: Expression 20 aims to ensure that the selected substrate path $\boldsymbol{mn}$ must spare enough link bandwidth to fulfill the bandwidth demand of virtual link $\boldsymbol{MN}$ that is mapped onto the selected substrate path $\boldsymbol{mn}$.

$$\exists \boldsymbol{mn} \in \boldsymbol{P}_{mn}^S, \forall \boldsymbol{MN} \in \boldsymbol{L}^V, \sum_{\boldsymbol{mn}} \boldsymbol{B}_{MN}^V \bullet (\boldsymbol{y}_{MN}^{mn} + \boldsymbol{y}_{NM}^{mn}) \le \boldsymbol{B}_{mn} \quad (20)$$

6) *Propagation Delay Limitation of Virtual Links*: The virtual link propagation delay constraint is shown in expression 21. The propagation delay of one selected substrate path $\boldsymbol{mn}$, mapping onto virtual link $\boldsymbol{MN}$, must not violate the required virtual link propagation delay $\mathbf{D}_{MN}^V$. In previous VNE research, the virtual link propagation delay has not been considered and formulated. Together with the variable k for selecting candidate substrate paths of virtual link $\boldsymbol{MN}$ (*subsection* A), expression 21 further assures that any selected substrate path propagation delay $\boldsymbol{D}_{mn}$ must be within the required virtual link propagation delay $\mathbf{D}_{MN}^V$. Virtual link $\boldsymbol{MN}$ maps onto the substrate path $\boldsymbol{mn}$.

$$\forall \boldsymbol{MN} \in \boldsymbol{L}^V, \exists \boldsymbol{mn} \in \boldsymbol{P}_{mn}^S, \sum_{\boldsymbol{mn}} \boldsymbol{D}_{mn} \bullet (\boldsymbol{y}_{MN}^{mn} + \boldsymbol{y}_{NM}^{mn}) \le \boldsymbol{D}_{MN}^V \quad (21)$$

D. Objective Functions

Objective function of CAN-A algorithm enables to optimize the allocation of substrate resources. In this *subsection*, we propose four different objective functions. Each objective function is regarded as one sub-algorithm of the CAN-A algorithm.

1) *Cost Function (**CF**)*: In general, the dominating aspect in VNE research is to minimize the cost of accepted VNRs in order that more substrate resources can be reserved and available. Leaving more substrate resources will lead to accommodating more proposed VNRs. According to the expression 10, the $\boldsymbol{CF}$ is constructed. The $\boldsymbol{CF}$, aiming at consuming less substrate node capacity and link bandwidth, is formulated in expression 22 below. α and β are weight factors in this expression. Similar to references [14] and [19], α and β are set to be equal in this paper.

$$\min\ \alpha \bullet \sum_{\boldsymbol{M}} \boldsymbol{x}_M^m \bullet \boldsymbol{C}_M^V + \beta \bullet \sum_{\boldsymbol{MN}} \sum_{\boldsymbol{mn}} (\boldsymbol{y}_{MN}^{mn} + \boldsymbol{y}_{NM}^{mn}) \bullet \boldsymbol{B}_{MN}^V \bullet \boldsymbol{D}_{mn} \quad (22)$$

2) *Link Propagation Delay Function (**LDF**)*: Different from the objects of previous algorithms [14][19], this part aims to minimize overall link propagation delay of accepted VNRs. As known to all, the link propagation delay affects Quality of Service (QoS) a lot, especially for delay sensitive services. Delay sensitive service needs guaranteed delay, for instance 5ms maximal end-to-end delay. However, it is hard to study the propagation delay of each concrete network link in detail. For simplicity, an assumption that each substrate link has the same unit time propagation delay is made in this paper. The object function $\boldsymbol{LDF}$ is shown in expression 23. The value of α is equal to the weight factor in $\boldsymbol{CF}$.

$$\min\ \alpha \bullet \sum_{\boldsymbol{MN}} \sum_{\boldsymbol{mn}} (\boldsymbol{y}_{MN}^{mn} + \boldsymbol{y}_{NM}^{mn}) \bullet \boldsymbol{D}_{mn} \quad (23)$$

3) *Cost and Load Balancing Function (**CLBF**)*: Expression 22 goes well in situations where the SN has abundant network resources. When the degree of flexibility in the resource assignment process is of great importance, it is necessary to consider VN cost and substrate load balancing at the same time. Both two goals can be constructed and achieved. Expression 24 below is the objective function ***CLBF***. Similar to references [14] and [19], α and β are set to be equal.

$$\min \alpha \bullet \sum_{m}\sum_{M}\frac{x_M^m \bullet C_M^V}{R^S(m)} + \beta \bullet \sum_{mn}\sum_{MN}\frac{(y_{MN}^{mn}+y_{NM}^{mn}) \bullet B_{MN}^V \bullet D_{mn}}{R^S(mn)} \quad (24)$$

4) *Cost and Link Propagation Delay Function (**CLDF**)*: ***CLDF*** in this part takes the embedding cost of SN and the total link propagation delay of accepted VNs into account at the same time. On the one hand, ***CLDF*** manages to minimize the cost of accepted VNRs. On the other hand, the overall link propagation delay of accepted VNRs is also to be minimized. ***CLDF*** is formulated in expression 25. According to the simulation, conducted in Section V, weight factors (α and β) are set to be smaller than γ, aiming to highlight the total link propagation delay in the object ***CLDF***.

$$\begin{aligned}\min\ & \alpha \bullet \sum_{M}\sum_{m} x_M^m \bullet C_M^V + \beta \bullet \sum_{MN}\sum_{mn}(y_{MN}^{mn}+y_{NM}^{mn}) \bullet B_{MN}^V \bullet D_{mn} \\ & + \gamma \bullet \sum_{MN}\sum_{mn}(y_{MN}^{mn}+y_{NM}^{mn}) \bullet D_{mn}\end{aligned} \quad (25)$$

V. Performance Evaluation

This section presents the simulation settings followed by the simulation results. This section elaborates on quantifying the advantages of CAN-A algorithm. Typical heuristic algorithms and the pure MIP algorithm are selected to compare with CAN-A by varying the number of VNRs per timetable [14]. The number of VNRs per timetable in this paper is equivalent to the number of VNRs per time window [14][19].

A. Simulation Environment and Settings

To evaluate the CAN-A, a discrete event simulator in JAVA has been implemented. The open source software tool GLPK [35] is used to solve the linear programming problem of CAN-A. Since network virtualization is still an emerging field, the actual characteristics of the substrate network and virtual networks are still not well understood. Therefore, synthetic network topologies are used to evaluate VNE algorithms in this paper. The simulation is implemented in the '*Simulation Platform for Scotfield Cao*' [39], originating from the open-source project ALEVIN [40]. The CAN-A algorithm, coded by the authors, is added to '*Simulation Platform for Scotfield Cao*'. Part codes of '*Simulation Platform for Scotfield Cao*' are made available to public [41].

In this paper, the substrate network is generated by using *the Waxman topology approach* [42], which is integrated as a separate function module in '*Simulation Platform for Scotfield Cao*'. Two substrate networks with different sizes are generated. The larger one is a medium-scaled substrate network. The number of substrate nodes is set to be 40. Set α=0.4 and β=0.3 in the *Waxman model* of substrate network. The node capacity and link bandwidth of substrate nodes and substrate links are integers uniformly distributed between 50 and 100. Each substrate node is randomly located within a uniformly distributed position between 0 and 100 on ***x*** and ***y*** coordinates. The link propagation delay of all substrate links are set to be one time unit. With respect to the other substrate network, it is small size. The number of substrate nodes is set to be 20. Other parameters of the small-sized substrate network are same to what are set in the medium-sized substrate network.

VNRs are generated by using *Waxman topology approach*, too. It is set that VNRs arrive in a *Poisson process*. Simulations are evaluated by varying VNRs arrival rate from 2 VNs per timetable to 9 VNs per timetable. The arrival rate increases by 1 each time. Each VNR has an exponentially distributed lifetime with an average value of μ=1000 time units. These parameters are similar to what are set in references [14] and [19]. To each VNR, the number of virtual nodes is an integer and follows a uniform distribution between 2 and 8. Link parameters of each VNR are same to what are set in substrate links. The node capacity and bandwidth requirements of virtual nodes and virtual links are integers, uniformly distributed between 1 and 20. Virtual nodes are randomly located within a uniformly distributed position between 0 and 100 on ***x*** and ***y*** coordinates. The maximum allowed deviation of each virtual node is an integer and uniformly distributed between 3 and 8. Each virtual link propagation delay is an integer and uniformly distributed between 1 and 4.

To avoid an insight into two opposite scenarios, such as a very high or low VNR acceptance ratio, many trials are necessary to be conducted for each arrival rate. In this paper, 30 trials are made for each VNR arrival rate. In each trial, a new set of VNRs and a new substrate network are generated. All simulations are set to run up to 50000 time units to remove the initial transient phase effect [43] and to present the steady-state performance. Based on these settings, a confidence interval of 95% is achieved for all simulation results.

All simulations for different VNE algorithms are run on a Window 8 Desktop, with an Intel® Core (TM) CPU i7-4790 3.6GHz Processor and 16.00G RAM Machine. The time per VN embedding is registered. The GLPK version 4.58 is used to solve the linear programming problem of CAN-A and VNE-MIP. Among four sub-algorithms, weight factors (α and β) are set to be 1 in this paper. γ is set to be 10. To the VNRs proposed in a timetable, the VNR with smaller revenue is preferred [10]. Evaluation metrics are derived from the Section III-D and reference [6]. Simulation results are depicted in Section V-D & E.

B. Selected VNE Algorithms for Comparison

Ten VNE algorithms, including the CAN-A, make up the simulation part. The remaining nine VNE algorithms are Greedy Node Mapping with Shortest Path Link Mapping (G-SP) [10], Greedy Node Mapping with Multi-Commodity Flow Link Mapping (G-MCF) [10], Randomized Coordinated Node Mapping with Shortest Path Link Mapping (R-ViNE-SP) [14], Deterministic Coordinated Node Mapping with Shortest Path Link Mapping (D-ViNE-SP) [14], Randomized Coordinated Node Mapping with Multi-Commodity Flow Link Mapping (R-ViNE-MCF) [14], Deterministic Coordinated Node Mapping with Multi-Commodity Flow Link Mapping (D-ViNE-MCF) [14], Node Ranking Node Mapping with Shortest Path Link Mapping (NR-SP) [12], Node Ranking

Node Mapping with Multi-Commodity Flow Link Mapping (NR-MCF) [12] and the pure Mixed Integer Programming VNE algorithm (VNE-MIP) [14] (see Appendix). These algorithms are representative VNE algorithms in VNE research. These algorithms are all slightly modified to fit into the simulation of our paper. All VNE algorithms are enumerated in Table II below, along with short descriptions.

Table II
VNE ALGORITHMS for COMPARISON

Notation	Description
G-SP [10]	Greedy Node Mapping with Shortest Path Link Mapping
G-MCF [10]	Greedy Node Mapping with Multi-Commodity Flow Link Mapping
R-ViNE-SP [14]	Randomized Coordinated Node Mapping with Shortest Path Link Mapping
D-ViNE-SP [14]	Deterministic Coordinated Node Mapping with Shortest Path Link Mapping
R-ViNE-MCF [14]	Randomized Coordinated Node Mapping with Multi-Commodity Flow Link Mapping
D-ViNE-MCF [14]	Deterministic Coordinated Node Mapping with Multi-Commodity Flow Link Mapping
NR-SP [12]	Node Ranking Node Mapping with Shortest Path Link Mapping
NR-MCF [12]	Node Ranking Node Mapping with Multi-Commodity Flow Link Mapping
VNE-MIP [14]	Pure Mixed Integer Programming based VNE Algorithm
CAN-A-CF	CAN-A for Minimizing Overall Substrate Cost
CAN-A-LDF	CAN-A for Minimizing Overall Virtual Link Propagation Delay
CAN-A-CLBF	CAN-A for the Minimization of Overall Substrate Cost and Load Balancing
CAN-A-CLDF	CAN-A for Minimizing Overall Substrate Cost and Virtual Link Propagation Delay

C. *Simulation Comparisons*

The simulation work is composed of two parts. One part is the simulation comparison which includes the heuristics, VNE-MIP and CAN-A-CLBF. The aims of this part are to prove that the execution time of CAN-A is hugely cut down compared with MIP algorithm. Since VNE-MIP requires a very long time (in excess of about half an hour for a single embedding involving a SN of 50 nodes and a VNR of 10 nodes) to perform each VN embedding, network parameters in this part are restricted to a SN with 20 nodes and VNRs with nodes from 2 to 5. The other part is to use a medium-scaled substrate network of 40 nodes and excludes the pure VNE-MIP. This part elaborates on highlighting the efficiency and effectiveness of CAN-A, compared with the typical heuristics in the literature.

D. *Exact and Optimal Algorithm CAN-A-CLBF*

Derived from [14], the pure VNE-MIP, based on the MIP model, has been widely accepted as an exact and optimal algorithm in the literature. However, the NP-hard nature of MIP leads to high computation complexity. The VNE-MIP is not applicable in medium or large-scaled network scenarios. To prove that CAN-A-CLBF can be treated as another exact and optimal algorithm, a simulation is made in this *subsection*. The CAN-A-CLBF, having the same objective function with VNE-MIP, is selected from the CAN-A algorithm. Four representative heuristic algorithms, G-SP, G-MCF, ViNE-SP and ViNE-MCF, are also selected. Deterministic and randomized rounding techniques are not particularly labeled. Thus labeling ViNE-SP and ViNE-MCF. Simulation results are plotted ranging from Fig. 3 to Fig. 7 as a function of VNRs arrival rate in a timetable.

1) *Quality of CAN-A-CLBF*: Fig. 3 below shows the VNR acceptance ratio as a function of VNRs arrival rate. It is evident that CAN-A-CLBF has a close average VNR acceptance ratio to what is obtained by VNE-MIP. Other heuristic algorithms have a percentage of, at least 10%, lower than VNE-MIP and CAN-A-CLBF. That is to say, CAN-A-CLBF is able to behave as well as the VNE-MIP, in small-scaled network scenario. Fig. 4 and Fig. 5 are also plotted to indirectly prove the embedding ability of CAN-A-CLBF. The node and link utilization of CAN-A-CLBF are very close to that of VNE-MIP. The node utilization of CAN-A-CLBF is higher than the heuristics in general. It is owing to the fact that the CAN-A-CLBF enables to accommodate more proposed VNs by using the mathematical programming method after conducting the candidate subset processing. Therefore, CAN-A-CLBF uses the substrate network resources to their full capacity. The heuristics solve VNE problem only by referring to the variable-relaxed mathematical model or pure heuristic methods. A feasible solution is found and the optimal mapping is not able to be achieved in many cases. With respect to the link utilization in Fig. 5, the reason is same to what is explained for the node utilization in Fig. 4.

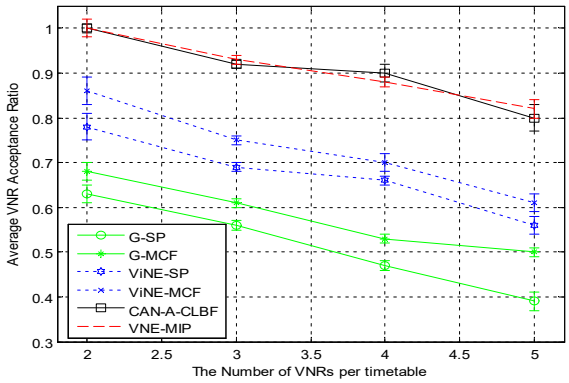

Fig. 3. VNR Acceptance Ratio as a function of VNRs Arrival Rate

In Fig. 6, it is apparent to discover that the average virtual link propagation delay of VNE algorithms that do not support path splitting, such as G-SP, ViNE-SP and CAN-A-CLBF, is lower than algorithms that support path splitting, G-MCF, ViNE-MCF and VNE-MIP. For example, to G-SP, a virtual link may just occupy two substrate links, making up a substrate path, in the link mapping, with link bandwidth and link propagation delay constraints fulfilled. To G-MCF, the virtual link is mapped onto two different substrate paths. One path occupies two substrate links while the other occupies three. The

average virtual link propagation is 2.5 substrate links. Therefore, the average virtual link propagation delay of G-SP is lower than G-MCF.

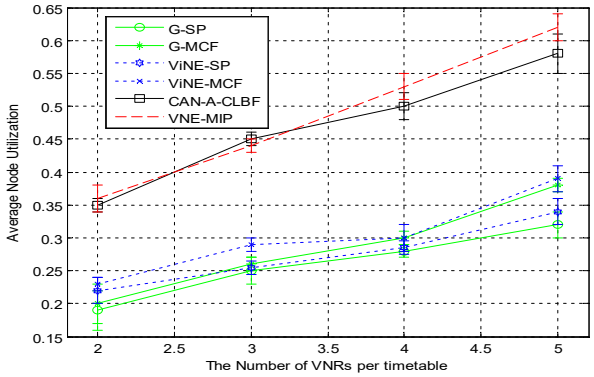

Fig. 4. Node Utilization as a function of VNRs Arrival Rate

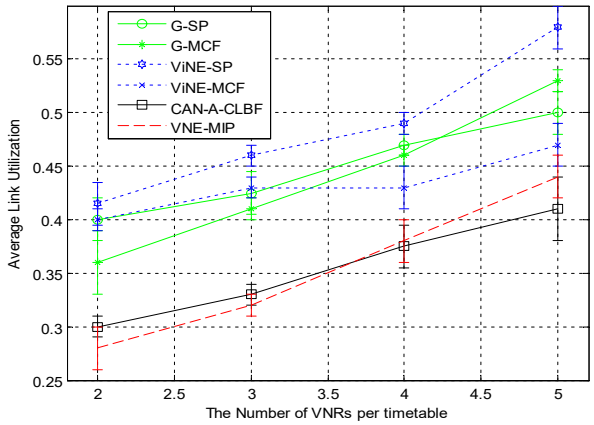

Fig. 5. Link Utilization as a function of VNRs Arrival Rate

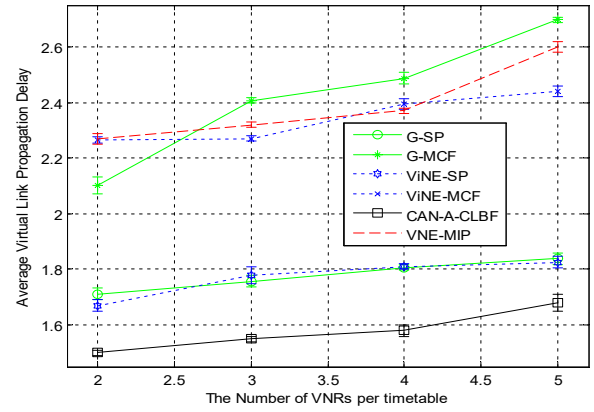

Fig. 6. Virtual Link Propagation Delay as a function of VNRs Arrival Rate

Numerical results are plotted as shown in Fig. 6, after a lot of trials. For a better performance of average virtual link propagation delay, CAN-A ought not to support path splitting, as mentioned in Section-IV.

2) *Time Complexity of CAN-A-CLBF*: With respect to the time complexity, Fig. 7 illustrates that the G-SP, G-MCF, ViNE-SP and ViNE-MCF have comparatively low execution time than that of CAN-A-CLBF. G-SP is not a mathematical algorithm and is therefore of low time complexity. While to the other three heuristics, it lies in solving the relaxed LP model. Though solved in polynomial time, they have higher execution time than the G-SP. To the CAN-A-CLBF, candidate substrate node and path sets of the given VNR are constructed ahead. The number of binary variables decreases a lot. The following ILP model is restricted to be solved in reasonable time. However, the execution time of ILP model is still higher than the relaxed LP model through numerous simulations. To the pure VNE-MIP, the execution time grows exponentially in Fig. 7. The pure VNE-MIP algorithm is therefore applicable with the size of network scenario expanding.

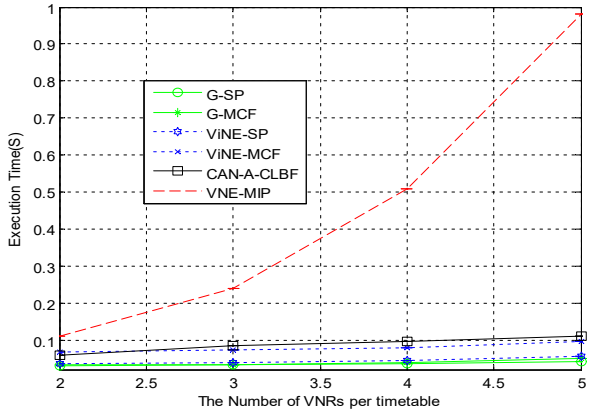

Fig. 7. Execution Time as a function of VNRs Arrival Rate

According to the illustrations in 1) and 2), it can be concluded that CAN-A-CLBF can be regarded as another exact and optimal VNE algorithm in VNE. Compared with non-optimal heuristic algorithms, CAN-A algorithm insures the optimal mapping solution for any given virtual network request. At the same time, the computational complexity is much lower when comparing with the optimal VNE-MIP algorithm.

E. Efficiency and Effectiveness of CAN-A Algorithm

1) *VNR Acceptance Ratio*: Fig. 8 presents the average VNR acceptance ratio as a function of VNRs arrival rate. VNR acceptance ratio is an important metric in evaluating different VNE algorithms. Fig. 8 shows that the acceptance ratio of all selected algorithms almost decays linearly with the variation on the VNRs. This decay shows the fact that there are no infinite substrate resources for embedding increasing proposed VNRs. In addition, the CAN-A outperforms the selected heuristics. The differences between the best heuristic and CAN-A-CF is at least 10%. It runs as expected because pure ILP model is applied to solve the VNE problem after candidate subset processing. Therefore, optimal mapping of each VNE problem, with regard to one concrete object, can be obtained in restricted solution space. With respect to the heuristics, a feasible mapping is tried to be found. Another reason for the results is the nature of the CAN-A. The CAN-A considers the universe of all possible embedding solutions in the restricted solution space, rather than a few solutions. For example, the first case with 2 VNRs in a timetable, the CAN-A will accept all VNRs, while the acceptance ratios of the remaining heuristics are between 68% and 88%. The CAN-A-CLBF achieves the highest acceptance ratio while the lowest ratio is obtained by G-SP.

On the whole, the CAN-A outperforms the heuristics in all

scenarios. CAN-A is an efficient and effective VNE algorithm. If the feasible assignment of a network scenario exists, the CAN-A will find out the optimal assignment in allowed convergence accuracy and restricted solution space. However, it is not always the case for the heuristic algorithms. Frequently a feasible assignment will be found out though it might be the optimal mapping.

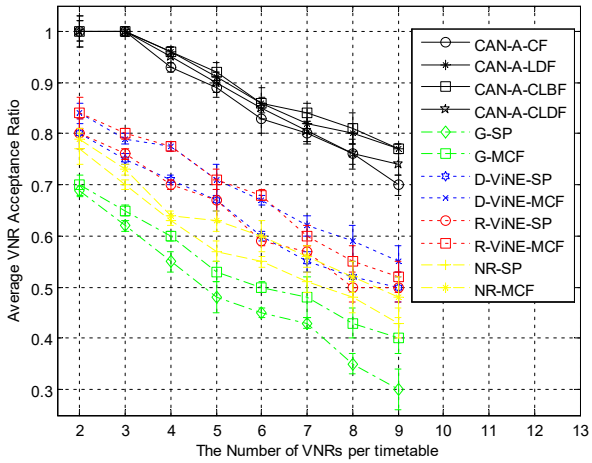

Fig. 8. VNR Acceptance Ratio as a function of VNRs Arrival Rate

2) *Node Utilization*: The average node utilization as a function of VNRs arrival rate is depicted in Fig. 9. With the arrival rate increasing, the node utilization of all selected algorithms increases, too. When the number of VNRs is small, i.e., 2 VNRs, the node utilization does not go beyond 28% and 34% for the heuristics and CAN-A respectively. As illustrated in Fig. 9, the CAN-A consumes more resources of substrate nodes than the heuristic algorithms. It runs as expected according to the performance of VNR acceptance ratio. The reason for CAN-A having a larger node utilization than the heuristics lies in CAN-A's ability of accommodating more VNRs than the heuristics. When the number of VNRs is large, the CAN-A algorithm is able to embed VNRs effectively and loads the substrate nodes to full capacity.

The node utilization also shows a dependency on the VNR acceptance ratio, as it can be perceived from Fig. 8 and Fig. 9. To get a better understanding of this dependency, the product of VNR acceptance ratio and node utilization is plotted in Fig. 10. It is obvious that the heuristics, such as G-SP, D-ViNE-SP and NR-SP, have the same behavior for all VNRs. The products of the heuristics are nearly constants. However, it is not same to the CAN-A. The product of CAN-A increases until 6 VNRs per timetable. The product nearly keeps the same behavior with arrival rate increasing beyond 6. It is owing to the fact that the CAN-A keeps with the increase until the VNE problem changes from an optimization problem (enough substrate resources for mapping VNRs) to a feasible problem (no sufficient substrate resources for embedding VNRs). With respect to different objects of CAN-A, the number of VNRs has a little difference, shown in Fig. 10. The product of CAN-A-CF is a constant from 5 VNRs per timetable. When to the other three sub-algorithms, the VNR arrival rate is up to 6.

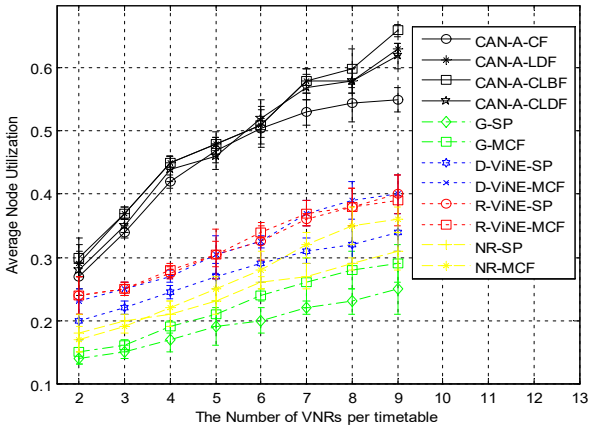

Fig. 9. Node Utilization as a function of VNRs Arrival Rate

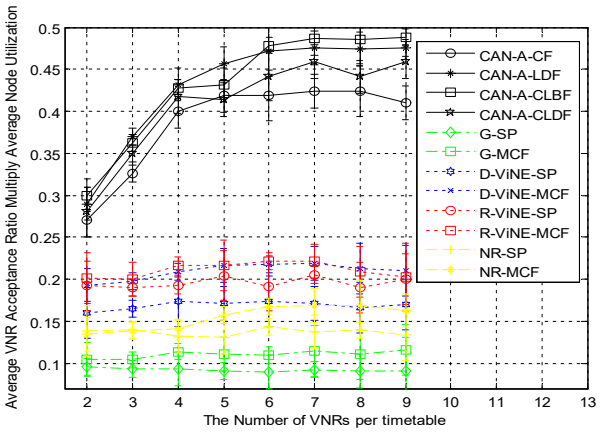

Fig. 10. VNR Acceptance Ratio Multiply Node Utilization as a function of VNRs Arrival Rate

3) *Link Utilization*: The link utilization is plotted in Fig. 11 and does not have the same behavior according to the VNRs arrival rate for all algorithms (Fig. 9, for the node utilization). It is evident that the CAN-A algorithm, especially for CAN-A-CLBF, has lower utilization on the substrate links; in the extreme scenario (9 VNRs per timetable) the average link utilization of CAN-A-CLBF is around 51%. With respect to the highest link utilization, R-ViNE-SP and D-ViNE-SP have link utilization of around 72% and 70%.

The differences in link utilization result from the link mapping strategy. For instance, if the object is to consume less bandwidth, the shortest path method (SP) will be adopted and the corresponding average link utilization will be high. Multi-commodity flow method (MCF) may contribute to the decrease of link utilization. Since the node and link mappings are two separate components in VNE, the performance of node mapping will have a great influence on the following link mapping stage. In Fig. 11, these algorithms (i.e. NR-SP, G-SP), which use greedy node mapping, have a lower link utilization though MCF is adapted in the link mapping. Based on this, it is of great importance to simultaneously map virtual nodes and links according to the mathematical programming model. Therefore, the CAN-A behaves well in link utilization.

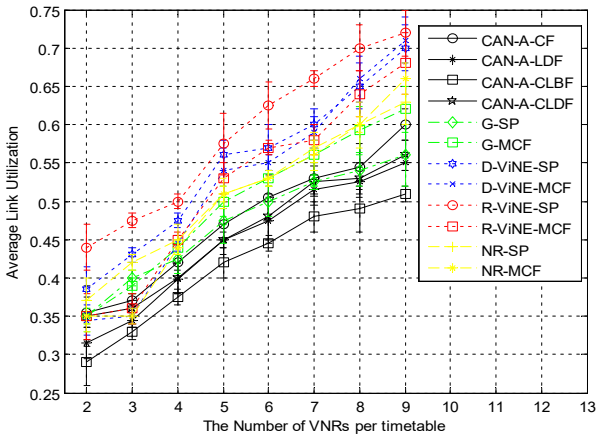

Fig. 11. Link Utilization as a function of VNRs Arrival Rate

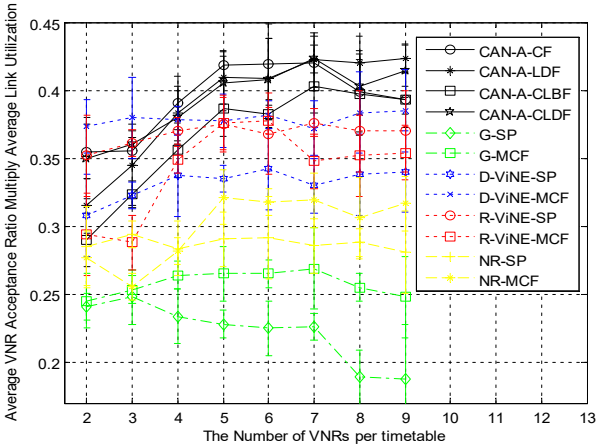

Fig. 12. VNR Acceptance Ratio Multiply Link Utilization as a function of VNRs Arrival Rate

The link utilization also has a dependency on the VNR acceptance ratio, as observed in Fig. 8 and Fig. 11. To further understand the dependency, the VNR acceptance ratio multiply the link utilization is plotted in Fig. 12. Overall, CAN-A increases significantly until the arrival rate is increased to 5. CAN-A starts to decrease or increase to a stable constant after 6 VNRs in a timetable. With respect to the selected heuristic algorithms, these products are almost constants, except G-SP.

4) *Virtual Link Propagation Delay*: Fig. 13 shows the average virtual link propagation delay as a function of VNRs arrival rate. Comparatively speaking, it runs as expected that with the number of VNRs increasing, the average virtual link propagation delay of all selected algorithms increases, same to the node / link utilization illustrated before. To all selected algorithms, it is more likely to have the substrate paths, consisting of more substrate links, mapped onto virtual links in order that more proposed virtual network requests can be accommodated.

In addition, two main discoveries can be also derived from the Fig. 13. The first discovery is that the VNE algorithms, such as G-MCF, that support path splitting and adopt to the multi-commodity flow method, will lead to a larger average virtual link propagation delay than other algorithms, i.e. G-SP, that do not support path splitting. This is true as illustrated in above *subsection D*. The second discovery is that the proposed CAN-A has a better virtual link propagation delay than all the heuristics. This discovery lies in the candidate subset construction and ILP model of CAN-A. With running the CAN-A, at most one intermediate node is allowed in the link mapping stage. The substrate propagation link is set to be one time unit in this paper. Therefore, the average virtual link propagation of CAN-A is within 1 and 2. One indirect proof of supporting the second discovery is that the advantage of VNR acceptance ratio of CAN-A model is not apparent than that shown in reference [19], which does not take the constraint of virtual link propagation into consideration. The CAN-A algorithm sacrifices partial VNR acceptance ratio performance for improving the performance of average virtual link propagation delay.

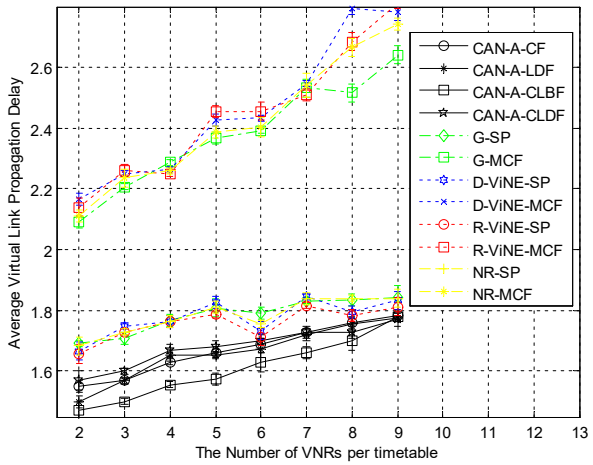

Fig. 13. Virtual Link Propagation Delay as a function of VNRs Arrival Rate

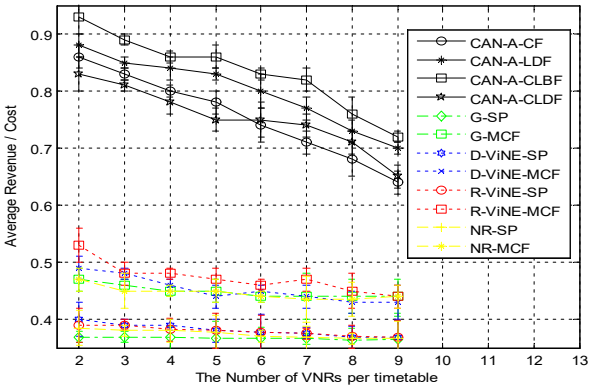

Fig. 14. Revenue / Cost as a function of VNRs Arrival Rate

5) *Revenue / Cost*: Fig. 14 illustrates the revenue / cost as a function of VNRs arrival rate. The parameters in expression (9) and (10), such as α and β, are set to be 1. The revenue / cost metric is used to evaluate the resource efficiency of different embedding algorithms. Derived from the Fig. 14, the revenue / cost of all selected algorithms decrease with the number of VNRs increasing. The slope of CAN-A is larger than the remaining heuristics. It is also obvious that the efficiency of the heuristics is around or lower than 50%. With respect to the optimal CAN-A algorithm, it has a good efficiency. These revenue / cost ratios are all higher than 60%. Particularly, the

CAN-A-CLBF outperforms among the four sub-algorithms. The revenue/cost of CAN-A-CLBF is over 85% when the number of VNRs in a timetable is no more than 5.

6) *Execution Time*: An important metric for evaluating VNE algorithms is the time that is required to embed the proposed VNRs in a timetable. This metric is known as so called execution time [6]. In this paper, the execution time as a function of the VNRs arrival rate is depicted in Fig. 15.

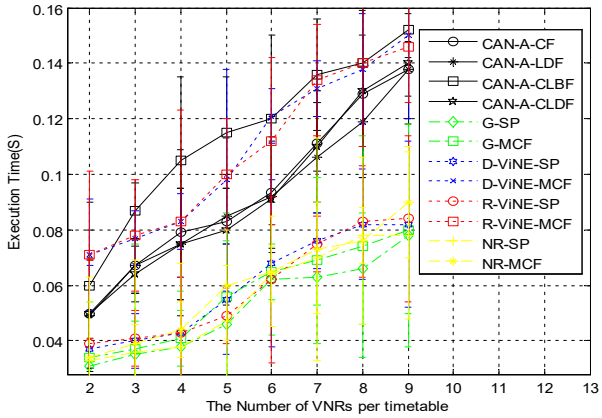

Fig. 15. Execution Time of Selected Algorithms as a function of VNRs Arrival Rate

To make the execution time of VNE algorithm convincing and worthy illustrated, several items must be set same in advance: i. all algorithms must run on the same computer; ii. all algorithms must run by using the same platform (e.g. '*Simulation Platform for Scotfield Cao*' in this paper); iii. the time to embed proposed VNRs must depend on the physical characteristics (e.g. CPU) of the computer and the nature of the VNE algorithm; iv. the same LP solver (GLPK in this paper) and same LP solving method (i.e. branch and bound method) must be adopted to solve CAN-A, D / R-ViNE-SP / MCF and G / NR-MCF.

Through illustrating the Fig. 15, two main behaviors can be observed. One behavior is that the execution time of all algorithms increases with the number of VNRs increasing. This is expected as a function of increased VNRs. More VNRs to be embedded, higher execution time. The other behavior is that algorithms which perform linear programming operations, D / R-ViNE-SP and G / NR-MCF, have more execution time than the pure heuristics in general, such as G / NR-SP. In addition, some other VNE algorithms, i.e. D / R-ViNE-MCF, perform two linear programming operations. Therefore, D / R-ViNE-MCF has higher execution time than one linear programming algorithm, e.g. D / R-ViNE-SP, G / NR-MCF. To CAN-A, candidate substrate nodes and paths are selected ahead. Therefore, the number of binary variables decreases a lot. The following ILP model is able to be solved in reasonable time, with regard to each objective function. Constraints of node capacity, link bandwidth and the virtual link propagation delay are considered simultaneously. Due to the restricted ILP model [26], programming and concrete network scenario, the execution time of CAN-A is able to be controlled within the range of heuristic algorithms that perform one or two linear programming operations.

VI. Conclusion and Future Work

This paper presents an optimal VNE mapping algorithm CAN-A with less computational complexity. The CAN-A algorithm can significantly reduce the algorithm complexity by constructing candidate substrate node and path subsets which are much smaller than the full sets, before performing the integer linear programming based mapping.

Virtual node location and virtual link propagation delay are formulated as node and link constraints in CAN-A algorithm. Previous VNE algorithms just embed proposed VNRs with constraints of node capacity and link bandwidth. Introducing the constraint of virtual link propagation delay in CAN-A is able to ensure the virtual link propagation delay within the required value. Four different objective functions are proposed in the CAN-A algorithm, thus forming four different sub-algorithms: the ***CF*** which aims to minimize the cost of the substrate network per VNR; the ***LDF*** that fights for the minimization of the total virtual link propagation delay of mapped VNs; the ***CLBF*** which aims to minimize the cost of the substrate network and tries to achieve the load balancing of the substrate network; the ***CLDF*** that has the goal of minimizing the cost per VNR and ensuring the minimization of virtual link propagation delay of mapped VNs.

Comprehensive simulations are conducted to assess the CAN-A algorithm. Results vividly show that the CAN-A algorithm outperforms typical heuristic algorithms. In terms of average VNR acceptance ratio, the difference between the heuristics and CAN-A is, at least, 10% (Fig. 8). The node utilization of CAN-A is higher than the heuristics because CAN-A is able to accommodate more proposed VNRs than the heuristics (Fig. 9). With respect to the average virtual link propagation delay of each given VNR, CAN-A outperforms the best-behaved heuristic (Fig. 13). All possible mapping assignments are considered in CAN-A. Therefore, the optimal mapping with the lowest average virtual link propagation delay, is able to be achieved by the CAN-A algorithm.

We identify the following directions for future study. With respect to the time convergence and algorithm execution time, path generation (column generation) [20][21] will be attempted in the CAN-A algorithm. The link propagation delay of each substrate link, set to be a time unit in this paper, can vary according to the ***Euclidean Distance*** between two end nodes. As to the virtual link propagation delay requirement, we only investigate how to optimize the average virtual link propagation delay of each virtual link in the given VNR. We are going to further optimize the total link propagation delay for some particular network topologies (e.g. ring topology, star topology). In addition, we intend to study the VNE problem in a real testbed environment and evaluate our CAN-A through a prototype implementation. Finally, it is worthwhile to extend the CAN-A algorithm to solve wireless virtual network embedding (WVNE) problem[44][45][46].

Appendix

VNE-MIP

This part presents the mixed integer programming (MIP) formulation of VNE-MIP, a modified version of [14], for solving the VNE problem. Two kinds of variables are

defined. f_{mn}^{i} is the allocated amount of flow on the substrate link mn for the i th virtual link. x_{Mm} is the binary variable which has the value "1" when virtual node M is mapped onto substrate node m.

$$\text{minimize} \sum_{m \in N^S} \frac{\alpha}{R^S(m) + \gamma} \sum_{M \in N^V} x_{Mm} \bullet C_M^V + \sum_{mn \in L^S} \frac{\beta}{R^S(mn) + \delta} \sum_i f_{mn}^i$$

Subject to

Node Constraints:

$$\forall M \in N^V, \quad \sum_{m \in N^S} x_{Mm} = 1$$

$$\forall m \in N^S, \quad \sum_{M \in N^V} x_{Mm} \le 1$$

Flow Constraints:

Source nodes:

$$\forall i \in L^V, \quad \sum_{m \in N^S} f_{mt_i}^i - \sum_{m \in N^S} f_{t_i m}^i = B_i^V$$

Sink nodes:

$$\forall i \in L^V, \quad \sum_{m \in N^S} f_{t_i m}^i - \sum_{m \in N^S} f_{mt_i}^i = -B_i^V$$

Intermediate nodes:

$$\forall mn \in L^S, \forall i \in L^V, \quad \sum_{n \in N^S} f_{mn}^i = \sum_{n \in N^S} f_{nm}^i$$

Capacity Constraints:

$$\forall mn \in L^S, \quad \sum_i (f_{mn}^i + f_{nm}^i) \le R_{mn}^S$$

$$\forall m \in N^S, \forall M \in N^V, \quad x_{Mm} \bullet C_M^V \le R_m^S$$

Domain Constraints:

$$\forall mn \in L^S, \quad 0 \leqslant f_{mn}^i \le R_{mn}^S$$

$$\forall m \in N^S, \forall M \in N^V, \quad x_{Mm} \in \{0,1\}$$

ACKNOWLEDGMENT

The authors would like to thank Dr. Philippe Owezarski and reviewers for their detailed reviews and constructive comments, which have helped improve the quality of this paper. This paper was supported by the National Basic Research Program of China (973 Program) under Grant 2013CB329104, the National Natural Science Foundation of China under Grant 61372124 and 61427801, Postgraduate Research & Practice Innovation Program of Jiangsu Province under Grant KYCX17_0784, the UK EPSRC under Grant number EP/N007840/1.

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, 2005.

[2] N. Chowhury and R. Boutaba, "Network virtualization: state of the art and research challenges,"*IEEE Commun. Mag.*, vol. 47, no. 7, pp. 20-26, July 2009.

[3] N. Chowhury and R. Boutaba, "A survey of network virtualization,"*Comput. Net.*, vol. 64, no. 5, pp. 862-876, 2010.

[4] J. Turner and D. Taylor, "Diversifying the internet," in *Proc. IEEE GLOBECOM*, 2005, vol. 2, pp. 755-760

[5] N. Feamster, L. Gao, and J. Rexford, "How to lease the internet in your spare time,"*Comput. Commun. Rev.*, vol. 37, no. 1, pp. 61-64, 2007.

[6] A. Fischer, J. Botero, M. Till Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey,"*IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888-1906, 4th Quart. 2013.

[7] H. Cao, L. Yang *et al.*, "Exact Solutions of VNE: A Survey"*China Commun.*, vol. 13, no. 6, pp. 48-62, June 2016.

[8] D. Andersen, "Theoretical approaches to node assignment," 2002 [Online].Available:http://www.cs.cmu.edu/~dga/papers/andersenassign.ps.

[9] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Pro. 2006 IEEE INFOCOM*, pp. 1-12.

[10] M. Yu, Y. Yi, *et al.*, "Rethinking virtual network embedding: substrate support for path splitting and migration,"*SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 17-29, Mar. 2008.

[11] J. Lischka and H. Karl, "A virtual network embedding algorithm based on subgraph isomorphism detection," in *Proc. 1st ACM Workshop VISA*, 2009, pp. 81-88.

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

[13] M. Chowhury, M. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 783-791.

[14] M. Chowhury, M. 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] I. Houidi, W. Louati, and D. Zeghlache, "A distributed virtual network mapping algorithm," in *Proc. 2008 IEEE ICC*, pp. 5634-5640.

[16] J. Botero *et al.*, "Energy efficient virtual network embedding,"*IEEE Commun. Lett.*, vol. 16, no. 5, pp. 756-759, May 2012.

[17] S. Su, *et al.*, "Energy-Aware Virtual Network Embedding,"*IEEE/ACM Trans. Netw.*, vol. 22, no. 5, pp. 1607-1620, Oct. 2014.

[18] S. Su, *et al.*, "Energy-aware virtual network embedding through consolidation," in *Proc. IEEE INFOCOM WS-CCSES: Green Networking and Smart Grids*, 2012, pp. 2708-2713.

[19] M. Melo *et al.*, "Optimal Virtual Network Embedding: Node-Link Formulation,"*IEEE Trans. Netw. and Serv. Manag.*, vol. 10, no. 4, pp. 356-368, Dec. 2013.

[20] R. Mijumbi et al., "A Path Generation Approach to Embedding of Virtual Networks,"*IEEE Trans. Netw. and Serv. Manag.*, vol. 12, no. 3, pp. 334-348, Sep. 2015.

[21] A. Jarray and A. Karmouch, "Decomposition approaches for virtual network embedding with one-shot node and link mapping,"*IEEE/ACM Trans. Netw.*, vol. 23, no. 3, pp. 1012-1025, Jun.2015.

[22] 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.

[23] T. Trinh, H. Esaki, and C. Aswakul, "Quality of service using careful overbooking for optimal virtual network resource allocation," in *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology(ECTI-CON), 2011 8th International Conference on*, May 2011, pp. 296-299.

[24] M. Rahman and R. Boutaba, "SVNE: Survivable Virtual Network Embedding Algorithms for Network Virtualization,"*IEEE Trans. Netw. and Serv. Manag.*, vol. 10, no. 2, pp. 105-118, Jun. 2013.

[25] Long Gong, Huihui Jiang, Yixiang Wang and Zuqing Zhu, "Novel Location-Constrained Virtual Network Embedding(LC-VNE) Algorithms Towards Integrated Node and Link Mapping,"*IEEE/ACM Trans. Netw.*, vol. 24, no. 6, pp. 3648-3661, 2016.
[26] J. Kleinberg, "Approximation algorithms for disjoint path problems," Ph.D. Dissertation, MIT, Cambridge, MA, 1996.
[27] S. Kollopoulos and C. Stein, "Improved approximation algorithms for unsplittable flow problems," in *Proc. IEEE FOCS*, 1997, pp. 426-435.
[28] A. F. Molisch, *Wireless Local Area Networks*, New York, NY, USA: Wiley-IEEE , 2012.
[29] J. Stribling, V. Arunarthi, C. Knittle, D. Murayama and M. Emmendorfer, "Implementing Qos in SIEPON," *IEEE Commun. Mag.*, vol. 50, no. 9, pp. 128-135, 2012.
[30] S. Andreev, O. Galinina, A. Pyattaev *et al.*, "Exploring Synergy between Communications, Caching, and Computing in 5G-grade Deployments," *IEEE Commun. Mag.*, vol. 54, no. 8, pp. 60-69, 2016.
[31] A. Schrijiver, *Theory of Linear and Integer Programming*. New York, NY, USA: Wiley, 1998.
[32] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*, 2nd ed. New York, NY, USA:McGraw-Hill, 2001.
[33] M. Khan, N. Shahriar, R. Ahmed, and R. Boutaba, "Multi-path link embedding for survivability in virtual networks,"*IEEE Trans. Netw. and Serv. Manag.*, vol. 13, no. 2, pp. 253-266, 2016.
[34] S. Chowdhury *et al.*, "Dedicated Protection for Survivable Virtual Network Embedding,"*IEEE Trans. Netw. and Serv. Manag.*, vol. 13, no. 4, pp. 913-926, May. 2016.
[35] "GLPK(GNU Linear Programming Kit)," Jul. 2016. Available: http://www.gnu.org/software/glpk/.
[36] "IBM ILOG Optimization Products," Jul. 2016. Available: www=01.ibm.com/software/websphere/products/optimization.
[37] D. Dietrich, A.. Rizk, and P. Papadimitriou, "Multi-provider virtual network embedding with limited information disclosure,"*IEEE Trans. Netw. and Serv. Manag.*, vol. 12, no. 2, pp. 188-201, May. 2015.
[38] T. Mano, T. Inoue, D. Ikarashi, K. Hamada, K. Mizutani, and O. Akashi, "Efficient virtual network optimization across multiple domains without revealing private information, "*IEEE Trans. Netw. and Serv. Manag.*, vol. 13, no. 3, pp. 477-488, Jun. 2016.
[39] H. Cao, S. Hu and L. Yang, "New Functions added to ALEVIN for evaluating virtual network embedding," in *2nd Internation Conference on Computer and Communications, 2016 IEEE, Oct*. 2016, pp. 2411-2414.
[40] A, Fischer, J. F. Botero, M. Duelli, *et al*., "ALEVIN-a framework to develop, compare, and analyze virtual network embedding algorithms,"*Electronic Commun. of the EASST*, vol. 37, pp. 1-12, 2011.
[41] http://pan.baidu.com/s/1gekPZrl.
[42] B. M. Waxman, "Routing of multipoint connections,"*IEEE J. Sel. Areas Commun.*, vol 6, no. 9, pp. 1617-1622, 1988.
[43] R. Jain, The Art of Computer Systems Performance Analysis. John Wiley & Sons, 1991, vol. 182. Available: http://www.cmg.org/proceddings/1991/91INT145.pdf.
[44] S. Abdelwahab, B. Hamdaoui， M. Guizani*et al.*, "Efficient Virtual Network Embedding with Backtrack Avoidance for Dynamic Wireless Networks,"*IEEE Trans. on Wire. Comm.*, vol. 15, no. 4, pp. 2669-2683, Apr. 2016.
[45] C. Liang, and F. Yu, "Wireless network virtualization: a survey, some research issues and challenges,"*IEEE Commun. Sur. & Tuto.*, vol. 17, no.1, pp. 358-380, 2015.
[46] G. Liu, F. Yu, H. Ji and V. Leung, "Distributed Resource Allocation in Virtualized Full-Duplex Relaying Networks," *IEEE Trans. on Veh. Tech.*, vol. 65, no. 10, pp. 8444-8460, Oct. 2016.