Computer Networks 121 (2017) 89–99

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Secure virtual network embedding with flexible bandwidth-based revenue maximization☆

Cihangir Beşiktaş[a], Didem Gözüpek[a,*], Aydın Ulaş[b], Erhan Lokman[b]

[a] Department of Computer Engineering, Gebze Technical University, Turkey
[b] Argela Technologies, Turkey

ARTICLE INFO

Article history:
Received 15 December 2015
Revised 10 November 2016
Accepted 7 April 2017
Available online 8 April 2017

Keywords:
Virtualization
Virtual network embedding
Virtual network assignment
Resource allocation
Optimization
Integer linear programming
Heuristic algorithm

ABSTRACT

Network virtualization is an effective way to overcome the ossification of Internet by enabling multiple virtual networks to coexist on a shared infrastructure. Virtual network embedding is a resource allocation problem concerned with the assignment of physical resources to the virtual networks. Several security issues about virtual network embedding are hitherto unexplored. For instance, some virtual network operators may distrust each other and require that their virtual infrastructure is not cohosted on the same physical equipment. In this paper, we address this problem by proposing a virtual network embedding problem that ensures that the virtual networks of conflicting operators are mapped to different physical equipments. Furthermore, our problem formulation enables the virtual links to select among a range of discrete bandwidth values, each with a corresponding price and thereby realizing any possible revenue function. We evaluate the performance of our heuristic algorithm by comparison with the results obtained from our integer linear programming formulation using optimization software CPLEX.

1. Introduction

The explosive growth of Internet encourages the development of new technologies and applications; however, its large scale hinders their deployment. Since there are numerous service providers, applying a new architecture or technology requires mutual agreements among Internet Service Providers (ISPs) and necessitates changes in the routers and main computers. Therefore, Internet is increasingly becoming ossified. To deal with this problem, the concept of "network virtualization" has been proposed in the literature. In this approach, the substrate (physical) network provider offers a substrate (physical) network to support virtual networks [1]. This way, the deployment of new technologies becomes possible without any need to change the physical network or negotiate contracts between the ISPs [2]. An *infrastructure provider (InP)* deploys and maintains the network equipment and a *service provider (SP)* is responsible for the deployment of network protocols and the offering of end-to-end services, while a *virtual network provider (VNP)* assembles virtual resources from one or more InPs and a *virtual network operator (VNO)* installs, manages and operates the virtual network [3].

A major resource allocation challenge in virtual networks is the *Virtual Network Embedding (VNE)* problem, which is to embed virtual networks in a substrate network by adhering to some constraints like bandwidth requirements and optimizing a certain objective function such as revenue or energy efficiency. This problem is also known as Virtual Network Assignment problem in the literature. There are numerous variants of this problem in the literature with various constraints and objective functions such as CPU, disk, and memory requirements of the substrate and virtual links, load balancing, maximum length requirement for the virtual paths, requirement on the maximum number of virtual nodes or links that can be assigned to a certain substrate node or link, and economical benefits [4].

Authors in [5] proved that the VNE problem is NP-hard by reduction from the multi-way separator problem. Besides, another study [6] has proved by reduction from the unsplittable flow problem that the problem is still NP-hard even when the virtual nodes are already assigned and the problem is merely making the virtual link assignments by adhering to the bandwidth requirements. Some studies in the literature focus on the online version of the problem [7], whereas some other studies [8] use reoptimization techniques based on producing new solutions by modifying old

☆ This work is supported by Argela Technologies, Istanbul, Turkey, as part of the MILAT project supported by the Turkish Undersecretariat for Defense Industries (SSM). A preliminary version of this paper appeared in the Proceedings of IEEE NOMS 2016.

* Corresponding author.
E-mail addresses: cihangirbesiktas@gtu.edu.tr (C. Beşiktaş), didem.gozupek@gtu.edu.tr (D. Gözüpek), aydin.ulas@argela.com.tr (A. Ulaş), erhan.lokman@argela.com.tr (E. Lokman).

http://dx.doi.org/10.1016/j.comnet.2017.04.020
1389-1286/

solutions. In addition, there are also some studies [9] that consider the case where the resource demands in the virtual network request (VNR) are time-varying. The survey paper in [3] categorizes the rich literature about VNE problems according to various criteria such as centralized/distributed, static/dynamic, concise/redundant as well as according to their objectives such as providing QoS-compliant embeddings, maximizing the economical profit and providing survivable embeddings. An important open research issue indicated by Fischer et al. [3] is security. When virtual networks belonging to different VNOs share the same physical equipment, vulnerabilities occur due to the possible deployment of malicious software by one VNO to attack the resources of the other VNO. These attacks can range from encyrption attacks that retrieve unauthorized information by exploiting security vulnerabilities in the virtualization software to denial of service (DoS) attacks. Therefore, different VNOs may distrust one another and require that their virtual infrastructure is not cohosted on the same physical equipment [3]. In this paper, we address this open research issue by providing a virtual network embedding formulation that ensures that the resources allocated to the conflicting virtual networks do not share the same physical resources. To the best of our knowledge, this paper is the first one in the literature that provides a VNE formulation with such a feature. Unlike this paper, other secure VNE formulations [10–12] treat security as a resource (similar to bandwidth or CPU) demanded and offered by the virtual and substrate networks, respectively, and neglects the possible request of the VNOs to not share the same physical resources by the virtual networks of a certain set of other VNOs. Besides, this feature of our model is useful also to provide survivability by providing disjoint multiple paths for certain virtual links that request more survivability (will be explained in detail in Section 2).

Works in the literature [7,13–24] consider the case where each link has a certain bandwidth request. The only work in the literature that relaxes this requirement is [25], where the bandwidth requirement of some links are only probabilistically satisfied. When each virtual link has a fixed bandwidth requirement, in order for a VNR to be accepted, the bandwidth requirements of all virtual links (possibly together with some other requirements) have to be satisfied. However, this is a rather restrictive and unrealistic requirement since being obliged to satisfy a fixed bandwidth level for each virtual link can cause feasibility problems and in reality, virtual links are usually fine with a range of bandwidth values rather than a single one. For service level specifications, a range of bandwidth values is appropriate as long as the range conforms to the SLA specifications. The only possible drawback in having a range of bandwidth values is the possible increase in the computational complexity. This trade-off between flexibility and computational complexity needs to be taken into account while determining the range of bandwidth values. Besides, revenue function used in most works in the literature [7,13–24] is a linear function of the total bandwidth used by the VNR. The work in [26] pinpoints the unrealistic nature of the linear function and instead uses an exponential cost function. Unlike our work, the paper in [26] focuses on cost rather than revenue, where the cost represents traffic utilization. Their motivation for exponential cost function is because of the fact that the costs increase very rapidly as the traffic utilization increases. When a revenue function is used, linear function is again unrealistic because of the economies of scale; i.e., in practice, the per-unit revenue decreases as the offered bandwidth value increases. Certainly, an operator may opt to use a linear function; however, an exponential function better serves the market needs since it makes it more appealing to utilize more bandwidth due to economies of scale. Unlike other works in the literature, we propose in this paper a model where the infrastructure provider offers a range of discrete bandwidth values, each with a corresponding price. Note that our model can be tailored to work in the fixed bandwidth case by giving only a single bandwidth and price pair as input. Moreover, each virtual link has a minimum and maximum bandwidth requirement depending on the application (email, video etc.). A customer's willingness to pay can also be incorporated to our model by appropriately changing the maximum bandwidth requirement depending on the maximum price that the customer is willing to pay. Our virtual network embedding formulation assigns bandwidth levels to virtual links so that both the requirements of the links are satisfied and the revenue of the infrastructure provider is maximized. The flexibility that our model offers results in higher revenue; therefore, it is advantageous for the infrastructure provider. Furthermore, unlike other works in the literature, not only linear or exponential, but any revenue function (as a discrete function of offered bandwidth values) can be realized by our model.

Our formulation in this paper is able to handle multiple VNRs and provides admission control in addition to location awareness; i.e., it ensures that each virtual node is mapped to a substrate node that is compliant with the distance requirement of the virtual node. Moreover, our model is suitable both for online and offline settings. In an online setting, each VNR can be embedded as soon as the VNR request arrives by giving this VNR request as input to our problem. If this VNR is in conflict with any previously allocated VNR, then the graph given as input to our problem can be modified by excluding these previously allocated resources that are in conflict with the new arrival and thereby ensuring conflict-free operation. In an offline setting, VNRs that arrive during a certain time period can be buffered and then our model can be executed.

Contributions of this paper can be summarized as follows:

(i) To the best of our knowledge, our virtual network embedding problem in this paper is the first one in the literature that ensures that conflicting VNRs do not share the same substrate resources
(ii) To the best of our knowledge, this is the first paper that enables flexibility of choosing a certain bandwidth for each virtual link among a set of discrete bandwidth levels
(iii) To the best of our knowledge, this paper proposes the first VNE problem that enables the revenue to be any function of bandwidth allocated to the virtual links.

2. Problem formulation

2.1. Substrate network

We model the substrate network as an undirected graph and denote it by $G^S = (V^S, E^S)$, where V^S is the set of substrate nodes and E^S is the set of substrate links. Each substrate node $v^S \in V^S$ is associated with a location $loc(v^S)$. Each substrate link $(u, w) \in E^S$ among substrate nodes $u, w \in V^S$ is associated with a bandwidth capacity value $B_E(u, w)$ denoting the total amount of bandwidth of the link.

2.2. Virtual network requests (VNRs)

Let $\mathcal{G}$ be the set of VNRs and g be the index of a VNR such that $g \in \{1, 2, \cdots, |\mathcal{G}|\}$. Each VNR g is associated with a graph $G_g^V = (V_g^V, E_g^V)$, where V_g^V denotes the set of virtual nodes and E_g^V denotes the set of virtual links. Each VNR g has an associated non-negative value D_g^V expressing how far a virtual node $v_g^V \in V_g^V$ can be placed from the location specified by $loc(v_g^V)$. D_g^V can be expressed in terms of physical distance or in terms of permissible delay from $loc(v_g^V)$. Moreover, each virtual link $i \in E_g^V$ in a VNR g is associated with b_{min}^{gi} and b_{max}^{gi}, which denote the minimum and maximum requested bandwidth, respectively, of the link. The values of b_{min}^{gi} and

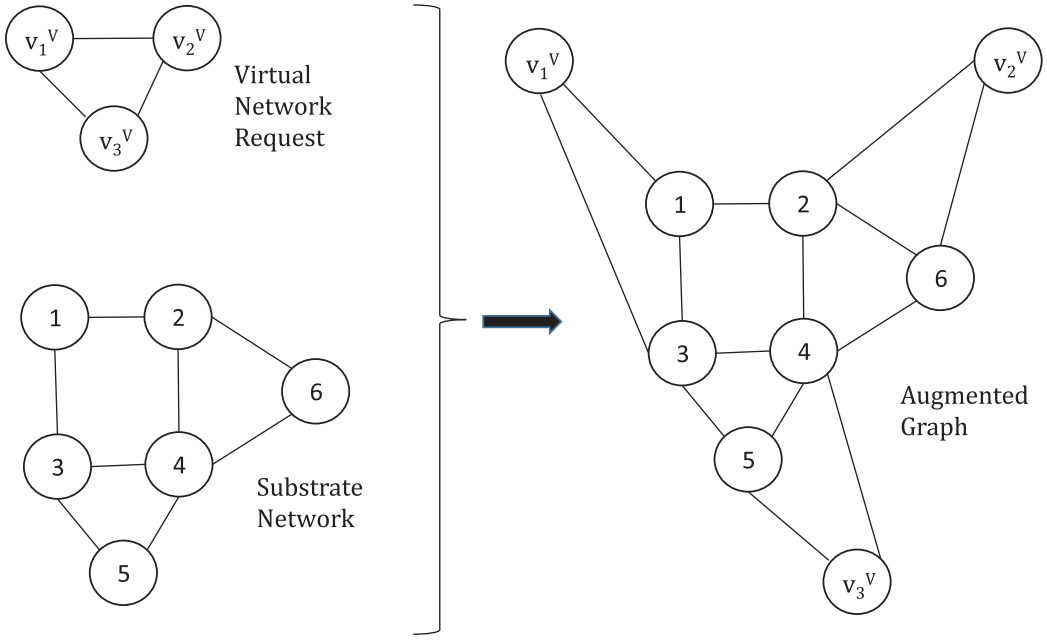


Fig. 1. An example augmented graph, where for instance substrate nodes 1 and 3 adhere to the location requirements of virtual node v_1^V.

b_{max}^{gi} depend on the application executed by the link. For instance, a real-time application usually requires more bandwidth, whereas low bandwidth is usually sufficient for an email application.

In our model, the infrastructure provider offers a discrete set of bandwidth values to the VNRs. Each bandwidth value has a corresponding price. Each link i of an accepted VNR g is assigned a certain bandwidth level by adhering to the b_{min}^{gi} and b_{max}^{gi} values. Each bandwidth level is denoted by an index k. b_k and r_k denote the bandwidth value and price, respectively, corresponding to the bandwidth index k.

2.3. Augmented graph construction

As in [7], we first extend the substrate graph G^S to create an *augmented graph* $G^{S'} = (V^{S'}, E^{S'})$ using the location requirement of the virtual nodes as the basis for the extension. For each virtual node v_g^V of each VNR g, we create a corresponding node $\mu(v_g^V)$ called *meta-node* in the augmented graph and connect it to all substrate nodes that it can be assigned to without violating the location requirement. These substrate nodes are the ones whose locations are within a radius D_g^V of the corresponding virtual node's location. In other words, if we let $\Omega(v_g^V) = \{v^S \in V^S \mid dist(loc(v_g^V), loc(v^S)) \leq D_g^V\}$ where $dist(a, b)$ denotes the distance between a and b, then $V^{S'} = V^S \cup \bigcup_g V_g^V$ and $E^{S'} = E^S \cup \bigcup_g \{(\mu(v_g^V), v^S) | v_g^V \in V_g^V, v^S \in V^S\}$ (consult Fig. 1 for an example augmented graph).

2.4. Integer linear programming formulation (ILP)

The input variables of our ILP formulation is in Table 1. We designate the endpoints of each link i of VNR g as source node s_{gi} and destination node d_{gi} where it is immaterial which endpoint is designated as source or destination. Table 2 provides the decision variables used in our ILP formulation.

The objective function of our ILP formulation aims to maximize the total revenue as follows:

$$\max \sum_{g=1}^{|\mathcal{G}|} \sum_{i=1}^{|E_g^V|} \sum_{k=1}^{K} r_k z_k^{gi} \tag{1}$$

We first model the security requirement that conflicting VNRs are assigned disjoint substrate resources as follows:

$$t_w^g + t_w^{g'} \leq 2 - C_{gg'} \quad \forall g \neq g',\ \forall w \in V^S, \tag{2}$$

The following are capacity constraints. Constraint (3) ensures that the total flow on each substrate link does not exceed the capacity of the link:

$$\sum_{g=1}^{|\mathcal{G}|} \sum_{i=1}^{|E_g^V|} (f_{uw}^{gi} + f_{wu}^{gi}) \leq B_{uw} \quad \forall u, w \in V^{S'} \tag{3}$$

If a substrate node w is not used by VNR g, i.e., if $t_w^g = 0$, then constraint (4) guarantees that the substrate links incident to w does not carry any flow belonging to any virtual link of VNR g. Recall that the decision variable t_w^g is used in constraint (2) to model the security requirement that VNRs which require not to use the same substrate resources with each other are assigned distinct resources. In other words, constraint (4) makes the capacity of the substrate links incident to a substrate node that is not used by a certain VNR effectively zero. This way, conflicting VNRs are assigned not only to distinct substrate nodes, but also to distinct substrate links.

$$\sum_{u \in V^S} \sum_{i=1}^{|E_g^V|} (f_{uw}^{gi} + f_{wu}^{gi}) \leq t_w^g \times B_{uw} \quad \forall g,\ \forall w \in V^S \tag{4}$$

The following are flow constraints. Constraint (5) ensures that the sum of the flows entering and exiting a node must be equal except for the source and destination nodes. Besides, constraint (6) ensures that the flows should exit from a source node, whereas constraint (7) guarantees that the flows should enter a destination node.

$$\sum_{w \in V^{S'}} f_{uw}^{gi} - \sum_{w \in V^{S'}} f_{wu}^{gi} = 0 \quad \forall g, \forall i,\ \text{and } \forall u \in V^{S'} \setminus \{s_{gi}, d_{gi}\} \tag{5}$$

Table 1
Table for input variables.

Input variable	Explanation
V^S	= Set of substrate nodes
E^S	= Set of substrate links
V_g^V	= Set of virtual nodes in VNR g
E_g^V	= Set of virtual links in VNR g
$V^{S'}$	= Set of all nodes in the augmented graph $G^{S'}$
b_k	= Bandwidth value corresponding to bandwidth level k
r_k	= Revenue corresponding to bandwidth level k
K	= Total number of bandwidth levels
B_{uw}	= Capacity of the link between substrate nodes u and w
$C_{gg'}$	= $\begin{cases}1, & \text{if there is a conflict between VNR } g \text{ and } g'\text{; i.e., they cannot share any substrate resource}\\ 0, & \text{otherwise}\end{cases}$
s_{gi}	= Source node of virtual link i in VNR g
d_{gi}	= Destination node of virtual link i in VNR g
b_{min}^{gi}	= Minimum bandwidth requested by virtual link i of VNR g
b_{max}^{gi}	= Maximum bandwidth requested by virtual link i of VNR g

Table 2
Table for decision variables.

Decision variable	Explanation
z_k^{gi}	= $\begin{cases}1, & \text{if bandwidth level } k \text{ is assigned to virtual link } i \text{ of VNR } g\\ 0, & \text{otherwise}\end{cases}$
f_{uw}^{gi}	= Total amount of flow from node u to w belonging to virtual link i of VNR g in the augmented graph
f'_{uwgi}	= $\begin{cases}1, & \text{if some bandwidth level is assigned to the link between } u \text{ and } w \text{ in the augmented graph in the direction}\\ & \text{from } u \text{ to } w \text{ for virtual link } i \text{ of VNR } g\\ 0, & \text{otherwise}\end{cases}$
x_{uw}^{g}	= $\begin{cases}1, & \text{if substrate node } w \text{ is used by virtual node } u \text{ of VNR } g\\ 0, & \text{otherwise}\end{cases}$
y_g	= $\begin{cases}1, & \text{if VNR } g \text{ is accepted}\\ 0, & \text{otherwise}\end{cases}$
d_{kuw}^{gi}	= $\begin{cases}1, & \text{if } x_{uw}^{g} \times z_k^{gi} = 1\\ 0, & \text{otherwise}\end{cases}$
t_w^g	= $\begin{cases}1, & \text{if substrate node } w \text{ is used by VNR } g\\ 0, & \text{otherwise}\end{cases}$

$$\sum_{w\in V^S} f_{uw}^{gi} - \sum_{w\in V^S} f_{wu}^{gi} = \sum_{k=1}^{K} b_k z_k^{gi} \quad \forall g, \forall i, \text{ and } u = s_{gi} \tag{6}$$

$$\sum_{w\in V^S} f_{uw}^{gi} - \sum_{w\in V^S} f_{wu}^{gi} = -\sum_{k=1}^{K} b_k z_k^{gi} \quad \forall g, \forall i, \text{ and } u = d_{gi} \tag{7}$$

The following constraint ensures that the bandwidth assigned to each virtual link of an accepted VNR obeys its minimum and maximum requested values:

$$b_{min}^{gi} \times y_g \le \sum_{k=1}^{K} b_k z_k^{gi} \le b_{max}^{gi} \times y_g \quad \forall g, \forall i \tag{8}$$

The following constraints linearize the term $d_{kuw}^{gi} = x_{uw}^{g} \times z_k^{gi}$:

$$d_{kuw}^{gi} \le z_k^{gi} \quad \forall g, \forall i, \forall k, \forall u, w \in V^{S'} \tag{9}$$

$$d_{kuw}^{gi} \le x_{uw}^{g} \quad \forall g, \forall i, \forall k, \forall u, w \in V^{S'} \tag{10}$$

$$d_{kuw}^{gi} \ge z_k^{gi} + x_{uw}^{g} - 1 \quad \forall g, \forall i, \forall k, \forall u, w \in V^{S'} \tag{11}$$

The following constraints ensure that the amount of flow used by a virtual link i of VNR g in the direction from a node u to a node w in the augmented graph is equal to the bandwidth assigned to that virtual link. Furthermore, these constraints also ensure that the flow related to a virtual link does not pass in the augmented graph through the virtual nodes that are not endpoints of this virtual link:

$$f_{uw}^{gi} = \begin{cases} \sum_{k=1}^{K} b_k d_{kuw}^{gi} & \forall g, i, \ \forall w \in V^S, \ u = s_{gi} \\ 0, & \forall g, i, \ \forall w \in V^S, \ \forall u \in V^{S'} \setminus V^S \text{ and } u \ne s_{gi} \end{cases} \tag{12}$$

$$f_{wu}^{gi} = \begin{cases} \sum_{k=1}^{K} b_k d_{kuw}^{gi} & \forall g, i, \ \forall w \in V^S, \ u = d_{gi} \\ 0, & \forall g, i, \ \forall w \in V^S, \ \forall u \in V^{S'} \setminus V^S \text{ and } u \ne d_{gi} \end{cases} \tag{13}$$

The following constraint models the relationship between decision variables f'_{uwgi} and f_{wu}^{gi}:

$$f'_{uwgi} \le f_{wu}^{gi} \le f'_{uwgi} \times B_{uw} \quad \forall g, \forall i, \forall u, w \in V^{S'} \tag{14}$$

The following constraint avoids path splitting so that the bandwidth corresponding to a virtual link is carried over a single path in the substrate network:

$$\sum_{w\in V^{S'}} f'_{uwgi} \le 1; \forall g, \forall i, \forall u \in V^{S'} \tag{15}$$

Some works in the literature employ path splitting, whereas some others avoid it. By removing constraints (14) and (15), our formulation becomes able to employ path splitting. However, as will be discussed later, the fact that our formulation can ensure that conflicting VNRs are assigned to distinct substrate resources can be used in order to provide survivability such that a certain virtual link can be assigned a certain number of disjoint paths in the substrate network at the expense of a higher price paid to the infrastructure provider. Therefore, we opt to avoid path splitting in order to control the number of paths that a virtual link is assigned to on the substrate network.

The following constraints are related to the admission control of VNRs. If a VNR g is not accepted, then constraint (16) ensures that none of the substrate nodes are used by any virtual node of VNR g as follows:

$$x^g_{uw} \leq y^g; \forall g, \forall u, w \in V^{S'} \tag{16}$$

Likewise, if a VNR g is not accepted, then constraint (17) ensures that no bandwidth level is assigned to any virtual link of the VNR as follows:

$$z^{gi}_k \leq y^g; \forall g, \forall i, \forall k \tag{17}$$

Similarly, if a VNR g is not accepted, then constraint (18) ensures that the amount of flow corresponding to any virtual link of the VNR equals zero in all substrate links:

$$f^{gi}_{uw} \leq B_E(u,w) y^g; \forall g, \forall i, \forall u, w \in V^{S'} \tag{18}$$

The following constraint (19) ensures that if a VNR g is accepted, then exactly one bandwidth level is assigned to each virtual link of the VNR for which the maximum requested bandwidth value is positive. Likewise, if a VNR is not accepted, this constraint also ensures that no bandwidth level is assigned to any virtual link of the VNR.

$$\sum_{k=1}^{K} z^{gi}_k = y_g \quad \forall g, \forall i \text{ such that } b^{gi}_{max} > 0 \tag{19}$$

While giving the input to our ILP, we set $b^{gi}_{max} = 0$ for the links incident to the nodes that correspond to the virtual nodes in the augmented graph since they do not correspond to any substrate link.

The following constraint ensures that each virtual node of an accepted VNR is assigned to exactly one substrate node:

$$\sum_{w \in V^S} x^g_{uw} = y_g \quad \forall g, \forall u \in V^{S'} - V^S \tag{20}$$

The following constraint ensures that a substrate node can host at most one virtual node of an accepted VNR:

$$\sum_{u \in V^{S'} - V^S} x^g_{uw} \leq y_g \quad \forall g, \forall w \in V^S \tag{21}$$

The following constraint ensures that there is no flow between the virtual nodes since each virtual node is connected to some substrate node(s) in the augmented graph:

$$f^{gi}_{uw} = 0 \quad \forall g, \forall u, w \in V^{S'} - V^S \tag{22}$$

Finally, the constraints about the range of values that each decision variable can take is modeled as follows:

$$f^{gi}_{uw} \geq 0 \quad \forall u, w \in V^{S'} \tag{23}$$

$$z^{gi}_k, f'_{uwgi}, x^g_{uw}, y_g, d^{gi}_{kuw}, t^g_w \in \{0, 1\} \quad \forall g, \forall i, \forall k, \ \forall u, w \in V^{S'} \tag{24}$$

Let us note that our ILP formulation in (1)–(24) can be used to provide survivability as follows. For each virtual link that requires more than one disjoint path to be established, construct another graph obtained by the removal of such links. Furthermore, for each such virtual link, construct another graph that consists of merely that link together with its endpoints and take a replica of this graph as many times as the number of disjoint paths that this virtual link requires. We then add constraints which ensure that all these graphs are accepted or rejected simultaneously. We also add constraints which ensure that if a certain virtual node exists in more than one graph, then they are mapped to the same substrate node. Moreover, we replace constraint (2) by another constraint which makes sure that substrate nodes that these graphs use are distinct except for the substrate node that the virtual nodes representing the same virtual node in the original graph is mapped to. For instance, consider the graph in Fig. 2 where the virtual link ab requires two disjoint paths and all other virtual links require only one path. Then the graphs g_1, g_2, and g_3 are given as input to the ILP formulation, which is modified as in the following. Firstly, we add the following constraint, which ensures that all graphs are accepted or rejected simultaneously:

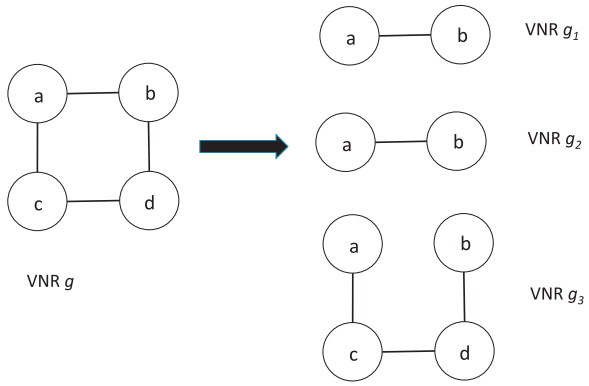


Fig. 2. Virtual link ab requires two disjoint paths, whereas all other virtual links require one path.

$$y_{g_1} = y_{g_2} = y_{g_3} \tag{25}$$

As with usual augmented graph creation, a separate node is inserted to the augmented graph for each virtual node. However, for virtual nodes that correspond to the same virtual node in the original graph g, i.e., nodes a and b in Fig. 2, the following constraints are added.

$$x^{g_1}_{aw} = x^{g_2}_{aw} = x^{g_3}_{aw} \quad \forall w \in V^{S'} \tag{26}$$

$$x^{g_1}_{bw} = x^{g_2}_{bw} = x^{g_3}_{bw} \quad \forall w \in V^{S'} \tag{27}$$

Furthermore, constraint (2) is replaced with the following set of constraints:

$$t^{g_1}_w + t^{g_2}_w \leq 1 + x^{g_1}_{aw} \tag{28}$$

$$t^{g_1}_w + t^{g_3}_w \leq 1 + x^{g_1}_{aw} \tag{29}$$

$$t^{g_2}_w + t^{g_3}_w \leq 1 + x^{g_1}_{aw} \tag{30}$$

$$t^{g_1}_w + t^{g_2}_w \leq 1 + x^{g_1}_{bw} \tag{31}$$

$$t^{g_1}_w + t^{g_3}_w \leq 1 + x^{g_1}_{bw} \tag{32}$$

$$t^{g_2}_w + t^{g_3}_w \leq 1 + x^{g_1}_{bw} \tag{33}$$

3. Proposed heuristic algorithm

Recall that VNE problem is NP-Hard even in its special cases [5,6]. Hence, we provide in this section a polynomial-time heuristic algorithm for the optimization problem we formulated in (1)–(24). Algorithm 1 presents the pseudocode of our proposed heuristic algorithm.

Our algorithm first finds in Line 1 a set of VNRs among which there is no conflict. To this end, we construct a conflict graph where the vertex set corresponds to the set of VNRs and there is an edge between two vertices if and only if there is a conflict

Algorithm 1 Heuristic algorithm.

1: Find a set $\mathcal{G}' \subseteq \mathcal{G}$ s.t. $C_{g'} = 0\ \forall g, g' \in \mathcal{G}'$ via [27]
2: Sort $\mathcal{G}'$ in descending order of total revenue $\sum_{g' \in \mathcal{G}'} \sum_{i=1}^{|E^V_{g'}|} b^{g'i}_{max}$
3: $\Omega^a \leftarrow \emptyset$ ▷ Set of accepted VNRs
4: $\Omega^r \leftarrow \emptyset$ ▷ Set of rejected VNRs
5: $D_g \leftarrow \emptyset$ ▷ Set of accepted VNRs that are in conflict with VNR g
6: **for all** $g \in \mathcal{G}'$ **do**
7: **if** (AssignVNR˜˜$(g, G^V_g) == Infeasible$) **then** ▷ Described in Table III
8: $\Omega^r \leftarrow \Omega^r \cup \{g\}$
9: **else**
10: $\Omega^a \leftarrow \Omega^a \cup \{g\}$
11: **end if**
12: **end for**
13: **for all** $g \in \mathcal{G} \setminus \mathcal{G}'$ **do**
14: **for all** $g' \in \Omega^a$ s.t. $C_{gg'} = 1$ **do**
15: $D_g \leftarrow D_g \cup \{g'\}$
16: **end for**
17: **if** $\sum_{i=1}^{|E^V_g|} r^{gi}_{max} \geq \sum_{g' \in D_g} \sum_{i=1}^{|E^V_{g'}|} r^{g'i}_{assigned}$ **then**
18: **for all** $g' \in D_g$ **do**
19: $B_{uw} \leftarrow B_{uw} + b^{g'i}_{assigned}\ \forall u, w$ on paths allocated to g'
20: **end for**
21: **if** AssignVNR˜$(g, G^V_g) == Infeasible$ **then**
22: **for all** $g' \in D_g$ **do**
23: $B_{uw} \leftarrow B_{uw} - b^{g'i}_{assigned}\ \forall u, w$ on paths allocated to g'
24: **end for**
25: **else**
26: $\Omega^a \leftarrow \Omega^a \cup \{g\}$ ▷ VNR g is accepted
27: **for all** $g' \in D_g$ **do**
28: $\Omega^r \leftarrow \Omega^r \cup \{g'\}$ ▷ All other VNRs in conflict with g are rejected
29: $\Omega^a \leftarrow \Omega^a \setminus \{g'\}$
30: **end for**
31: **end if**
32: **end if**
33: **end for**

between the two VNRs. In this conflict graph, finding a maximum independent set would yield the highest number of non-conflicting VNRs; however, maximum independent set problem is NP-Hard. Therefore, we employ a heuristic algorithm for the maximum independent set problem in this step by first implementing a heuristic algorithm for the minimum vertex cover problem via greedily picking the vertex with the highest degree and then removing it along with its neighborhood in each iteration [27]. We then take the complement of this set to find $\mathcal{G}'$. Our motivation here is to initially assign as much VNRs as possible without having to deal with conflict requirements.

In Step 2, we sort the VNRs by their total revenue calculated according to the maximum bandwidth requested by each virtual link. Our goal here is to greedily assign the VNR with high revenue so that total revenue increases. Steps 6–12 aim to assign each VNR in the sorted set $\mathcal{G}'$. If a feasible solution cannot be found, then the VNR is put into the set of rejected VNRs Ω^r in Line 8; otherwise, it is put into the set of accepted VNRs Ω^a in Line 10.

We then proceed with assigning the VNRs in $\mathcal{G} \setminus \mathcal{G}'$. In Lines 14–16, we put all VNRs g' in set Ω^a that have conflict with VNR g into set D_g. Line 17 checks whether accepting VNR g and rejecting all VNRs in D_g generates higher revenue. Here, r^{gi}_{max} refers to the revenue corresponding to b^{gi}_{max} and $r^{g'i}_{assigned}$ refers to the revenue corresponding to the bandwidth currently allocated to link i of VNR g'. Our algorithm then checks whether a feasible resource assignment for VNR g can be made if the resources allocated to the VNRs in D_g are released. Lines 18–20 simulates the release of bandwidth allocated to the VNRs in D_g. If a feasible assignment cannot be made for VNR g in Line 21, then Lines 22–24 reallocate the bandwidth previously allocated to the VNRs in D_g. On the other hand, if a feasible assignment can be made for VNR g, then Lines 26–29 accept VNR g and reject all VNRs in D_g.

Table 3 outlines the part of our algorithm to assign resources to a certain VNR on the substrate network. *Widest path problem*, a.k.a. the bottleneck shortest path problem or the maximum capacity path problem, is to find a path between two designated vertices in a given graph while maximizing the weight of the minimum-weight edge on the path. A similar problem called the *minimax path problem* asks for the path that minimizes the maximum weight of any of its edges. Any algorithm for the minimax path problem, and vice versa, can be transformed into an algorithm for the widest path problem by replacing every edge weight by its negation [29]. Edge weights correspond to the remaining capacity of each substrate link in our case. Lines 3–4 of function As-

Table 3
VNR assignment. The references cited in this table is [28].

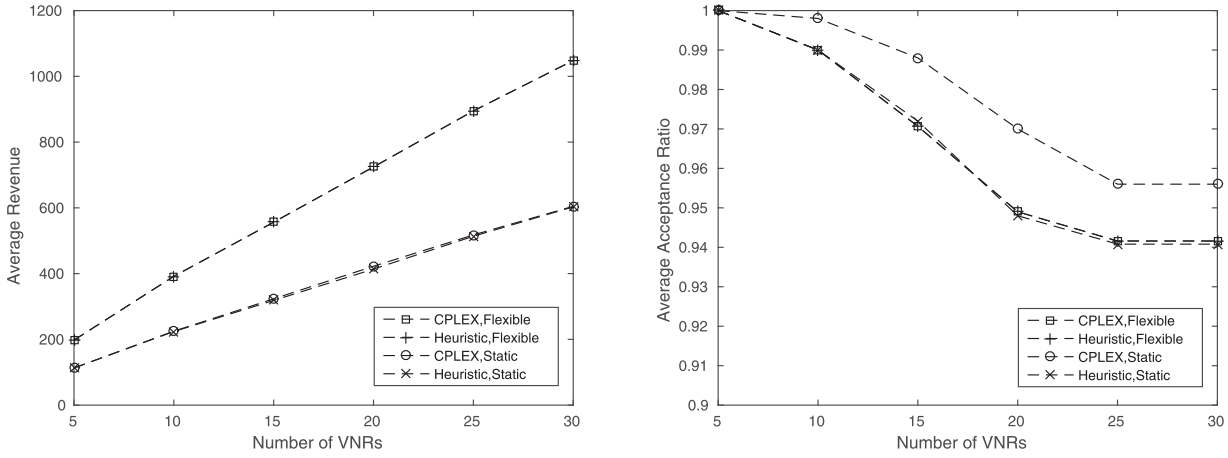
1: **function** AssignVNR $(g, G_g^V = (V_g^V, E_g^V))$
2: Update the augmented graph $G^{S'} = (V^{S'}, E^{S'})$ by adding G_g^V
3: $M \leftarrow \max B_{uw}$
4: $B_{uw} \leftarrow M + 1 - B_{uw}\ \forall u, w \in V^{S'}$
5: $\Gamma \leftarrow \emptyset$
6: **for all** $i \in E_g^V$ **do**
7: Run the heuristic algorithm in [28] for the minimax path problem for s_{gi} and d_{gi}, which returns a path p with minimum bandwidth b in the original augmented graph
8: **if** $b < b_{min}^{gi}$ **then**
9: **return** $Infeasible$
10: **else**
11: Assign path p to link i with bandwidth b_k such that k is the highest bandwidth level with $b_k \leq b$
12: $\Gamma \leftarrow \Gamma \cup \{p\}$
13: **end if**
14: **for all** i' s.t. $ii' \in E_g^V$ **do**
15: $s_{gi'} \leftarrow$ Neighbor of s_{gi} on path p
16: $t_{gi'} \leftarrow$ Neighbor of d_{gi} on path p
17: **end for**
18: **end for**
19: **return** Γ
20: **end function**

signVNRperform this negation. For each link of the VNR, line 7 of function AssignVNRemploys a heuristic algorithm for the widest path problem by implementing a heuristic algorithm for the minimax path problem in the negated graph. If the bandwidth value b returned by line 7 is smaller than the minimum required bandwidth b_{min}^{gi} for link i, then an infeasible solution is returned for the entire VNR. Otherwise, the found path is assigned to the virtual link together with the maximum possible bandwidth corresponding to the available bandwidth levels. Note that assigning a path for a virtual link also implies assigning each endpoint of the virtual link to a substrate node. Paths corresponding to the virtual links incident to a specific virtual node have to ensure that the virtual node is assigned to the same substrate node. To this end, when the first incident link of a virtual node is assigned some path, Lines 14–17 of function AssignVNRupdates the source (destination) node of all virtual links incident to this virtual node as the substrate node assigned to this first virtual link, i.e., the neighbor of the virtual node s_{gi} (d_{gi}) on the assigned path. Our heuristic algorithm clearly has a polynomial-time complexity since it consists of a polynomial number of iterations and each operation including the heuristics for maximum independent set and widest path problems takes polynomial time.

4. Simulation results

As in [7,14,30], we generate the substrate and virtual network topologies using GTITM [31]. The capacity of each substrate link is uniformly distributed between 15 and 30. We evaluate the impact of our flexible bandwidth scheme by comparison with the static bandwidth scheme results. In the flexible bandwidth scheme, each virtual link has minimum required bandwidth $b_{min} = 6$ and maximum required bandwidth $b_{max} = 14$. In the static bandwidth scheme, the bandwidth requested by each virtual link is $b_{max} = b_{min} = 10$. The offered bandwidth values are {6, 8, 10, 12, 14} and the revenue values corresponding to them are {2, 5, 8, 11, 14}, respectively. In other words, if the bandwidth value is 6, then its revenue is 2; if the bandwidth value is 8, then its revenue is 5 etc. We take the mean value of 50 experiments. Each experiment consists of a randomly generated substrate network such that each pair of substrate nodes are adjacent to each other with probability 0.5. The number of virtual nodes in each VNR is uniformly distributed between 2 and 10. A similar simulation scenario exists in [14].

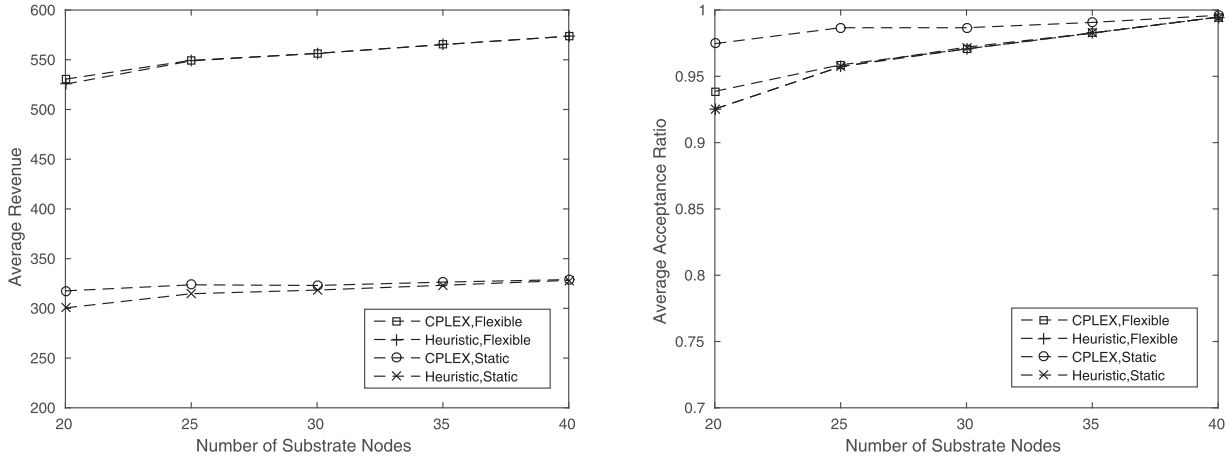
We first evaluate the impact of the number of VNRs. We set the substrate network size to 30 nodes and the conflict ratio to 20%;



(a) Average revenue with varying number of VNRs

(b) Average acceptance ratio with varying number of VNRs

Fig. 3. Comparison of CPLEX and heuristic algorithm outputs of flexible and static bandwidth schemes for varying number of VNRs.



(a) Average revenue with varying substrate network size

(b) Average acceptance ratio with varying substrate network size

Fig. 4. Comparison of CPLEX and heuristic algorithm outputs of flexible and static bandwidth schemes for varying substrate network size (number of substrate nodes).

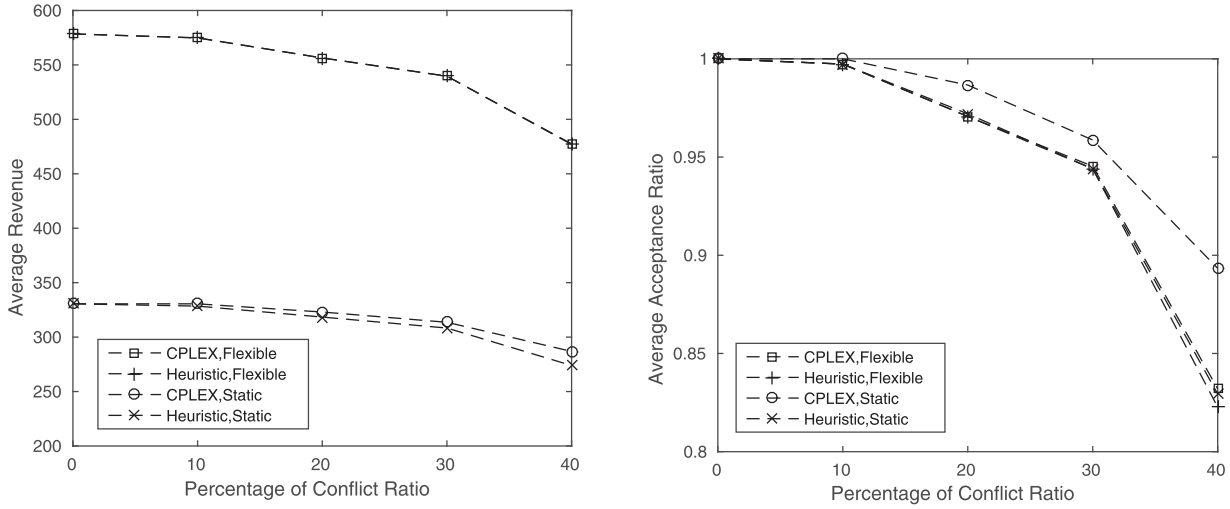
i.e., 20% of the VNR pairs have conflict. Fig. 3a demonstrates that the average revenue increases linearly as the number of VNRs increases. Furthermore, flexible bandwidth management scheme we propose in this paper yields significantly better results than the fixed bandwidth scheme. Moreover, the performance of our heuristic algorithm is close to the performance of CPLEX results. Fig. 3b displays the acceptance ratio of VNRs in the same simulation scenario. The average acceptance ratio decreases as the number of VNRs increases. In addition, the performance of our heuristic algorithm is close to the performance of CPLEX solutions. Furthermore, Fig. 3b also demonstrates that the static bandwidth scheme may yield higher acceptance ratio since the flexible bandwidth scheme may reject a certain set of VNRs in order to accept a VNR, most/all of the links of which can be allocated their maximum bandwidth value.

We then evaluate the impact of the substrate network size by varying the number of substrate nodes. We set the number of VNRs to 15 and the conflict ratio to 20%. Fig. 4a demonstrates that the average revenue increases as the substrate network size increases. However, when compared with Fig. 3a, the impact of substrate network size on revenue is less significant than the impact of the number of VNRs. Furthermore, flexible bandwidth scheme results in significantly higher revenue than the fixed bandwidth scheme. Fig. 4b shows the acceptance ratio of VNRs in the same simulation scenario. The average acceptance ratio increases as the substrate network size increases since a larger network can accommodate more VNRs. The performance of our heuristic algorithm is very close to the performance of CPLEX. Furthermore, flexible bandwidth scheme results in significantly higher revenue.

We then evaluate the impact of the conflict ratio. We set the number of VNRs to 15 and the size of the substrate network to 30. Fig. 5a shows that the revenue drops as the conflict ratio increases. The performance of our heuristic algorithm stays close to the performance of CPLEX even at higher conflict ratios. Fig. 5b shows the acceptance ratio of VNRs in the same simulation scenario. The average acceptance ratio naturally decreases as the conflict ratio increases.

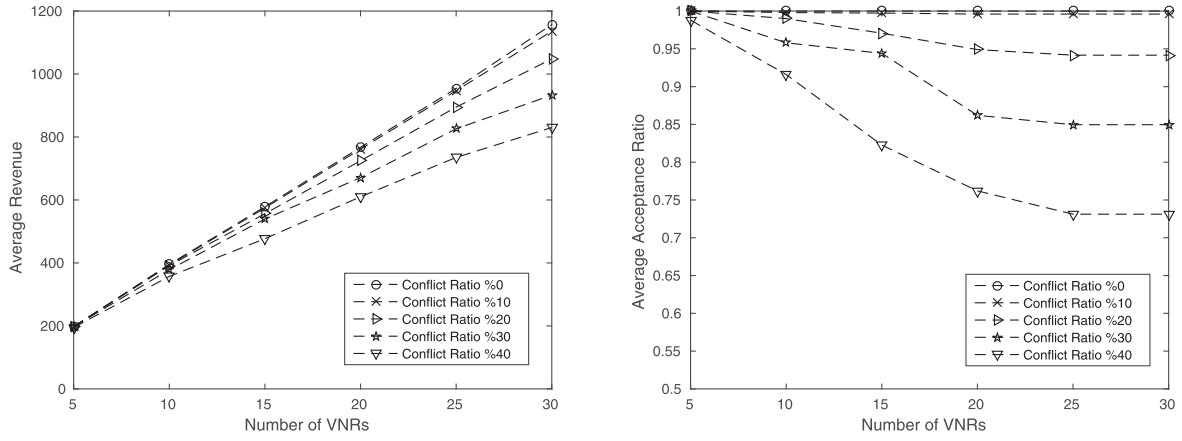
Fig. 6 evaluates the joint impact of conflict ratio and varying number of VNRs. We set the size of the substrate network to 30 nodes. Fig. 6 displays the CPLEX results of the flexible bandwidth case. As the conflict ratio increases, the linear relationship between the number of VNRs and revenue remains although revenue inevitably drops in this case. Fig. 6b shows that the drop rate of acceptance ratio increases as the conflict ratio increases.

Fig. 7 evaluates the joint impact of conflict ratio and varying substrate network size. We set the number of VNRs to 15.



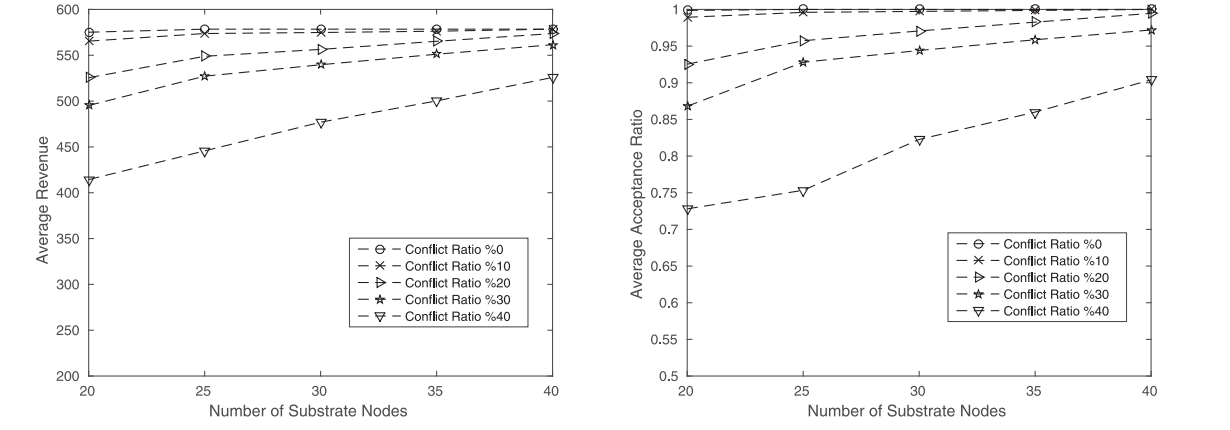
(a) Average revenue with varying conflict ratio (b) Average acceptance ratio with varying conflict ratio

Fig. 5. Comparison of CPLEX and heuristic algorithm outputs of flexible and static bandwidth schemes for varying conflict ratio.



(a) Average revenue with varying conflict ratio and number of VNRs (b) Average acceptance ratio with varying conflict ratio and number of VNRs

Fig. 6. CPLEX output of flexible bandwidth scheme for varying conflict ratio and number of VNRs.



(a) Average revenue with varying conflict ratio and substrate network size (b) Average acceptance ratio with varying conflict ratio and substrate network size

Fig. 7. CPLEX output of flexible bandwidth scheme for varying conflict ratio and substrate network size.

Fig. 7 displays the CPLEX results of the flexible bandwidth case. Substrate network size has more impact on the acceptance ratio and revenue as the conflict ratio increases. Fig. 7a shows that the drop in the average revenue as the conflict ratio increases is higher on a small substrate network. Fig. 7b illustrates that similar behavior exists for the acceptance ratio.

5. Conclusion

In this paper, we have formulated a secure virtual network embedding problem as an integer linear program. Our formulation ensures that the virtual networks of conflicting virtual network operators are not cohosted on the same physical equipment. Our formulation also offers flexible bandwidth management by enabling the virtual links to select among a range of possible bandwidth values, each having a different revenue and thereby realizing any possible revenue function. We propose a polynomial-time heuristic algorithm and evaluate the performance of our algorithm by comparison with the results obtained from our integer linear programming formulation using optimization software CPLEX. Our simulation results demonstrate that our heuristic algorithm yields very close results to the values obtained from CPLEX. Furthermore, our numerical evaluation also demonstrates that our flexible bandwidth management scheme results in higher revenue and higher virtual network admission ratio compared to a fixed bandwidth scheme consisting of only a single bandwidth level requested by each virtual link.

Possible future work is to extend this work by providing fairness to the virtual network operators in terms of acceptance ratio of their virtual network requests. This way, the tradeoff between the revenue of the virtual network provider and satisfaction of the virtual network operators can be addressed.

References

[1] N.M.K. Chowdhury, R. Boutaba, A survey of network virtualization, Comput. Networks 54 (5) (2010) 862–876.
[2] T. Anderson, L. Peterson, S. Shenker, J. Turner, Overcoming the internet impasse through virtualization, Computer (Long Beach Calif) 38 (4) (2005) 34–41.
[3] A. Fischer, J.F. Botero, M. Till Beck, H. De Meer, X. Hesselbach, Virtual network embedding: a survey, IEEE Commun. Surv. Tut. 15 (4) (2013) 1888–1906.
[4] A. Haider, R. Potter, A. Nakao, Challenges in resource allocation in network virtualization, in: 20th ITC Specialist Seminar, 18, 2009, p. 20.
[5] D.G. Andersen, Theoretical Approaches to Node Assignment, Technical Report, Computer Science Department, Carnegie Mellon University, 2002. [Online] Available: http://repository.cmu.edu/cgi/viewcontent.cgi?article=1079&context=compsci.
[6] C. Guo, G. Lu, H.J. Wang, S. Yang, Secondnet: A Data Center Network Virtualization Architecture with Bandwidth Guarantees, Technical Report, MSR-TR-2010-81, Microsoft Research, 2010.
[7] N. Chowdhury, M.R. Rahman, R. Boutaba, Virtual network embedding with coordinated node and link mapping, in: IEEE INFOCOM, 2009, pp. 783–791.
[8] N.F. Butt, M. Chowdhury, R. Boutaba, Topology-Awareness and Reoptimization Mechanism for Virtual Network Embedding, Springer, 2010.
[9] S. Zhang, Z. Qian, J. Wu, S. Lu, L. Epstein, Virtual network embedding with opportunistic resource sharing, IEEE Trans. Parallel Distrib. Syst. 25 (3) (2014) 816–827.
[10] S. Liu, Z. Cai, H. Xu, M. Xu, Security-aware virtual network embedding, in: IEEE International Conference on Communications (ICC), 2014, pp. 834–840.
[11] L.R. Bays, R.R. Oliveira, L.S. Buriol, M.P. Barcellos, L.P. Gaspary, Security-aware optimal resource allocation for virtual network embedding, in: International Conference on Network and Service Management, 2012, pp. 378–384.
[12] P. Chau, Y. Wang, Security-awareness in network virtualization: a classified overview, in: IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS), 2014, pp. 545–550.
[13] A. Xiao, Y. Wang, L. Meng, X. Qiu, W. Li, Topology-aware remapping to survive virtual networks against substrate node failures, in: IEEE Asia-Pacific Network Operations and Management Symposium (APNOMS), 2013, pp. 1–6.
[14] M. Chowdhury, M.R. Rahman, R. Boutaba, Vineyard: virtual network embedding algorithms with coordinated node and link mapping, IEEE/ACM Trans. Network. 20 (1) (2012) 206–219.
[15] X. Cheng, S. Su, Z. Zhang, K. Shuang, F. Yang, Y. Luo, J. Wang, Virtual network embedding through topology awareness and optimization, Comput. Networks 56 (6) (2012) 1797–1813.
[16] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, J. Wang, Virtual network embedding through topology-aware node ranking, ACM SIGCOMM Comput. Commun. Rev. 41 (2) (2011) 38–47.
[17] L. Richter Bays, R. Ruas Oliveira, L.S. Buriol, M. Pilla Barcellos, L.P. Gaspary, A heuristic-based algorithm for privacy-oriented virtual network embedding, in: IEEE Network Operations and Management Symposium (NOMS), 2014, pp. 1–8.
[18] G. Sun, H. Yu, V. Anand, L. Li, A cost efficient framework and algorithm for embedding dynamic virtual network requests, Future Gen. Comput. Syst. 29 (5) (2013) 1265–1277.
[19] I. Fajjari, N. Aitsaadi, G. Pujolle, H. Zimmermann, Adaptive-vne: a flexible resource allocation for virtual network embedding algorithm, in: IEEE Global Communications Conference (GLOBECOM), 2012, pp. 2640–2646.
[20] S. Abdelwahab, B. Hamdaoui, M. Guizani, Bird-vne: Backtrack-avoidance virtual network embedding in polynomial time, in: IEEE Global Communications Conference (GLOBECOM), 2014, pp. 4983–4989.
[21] A. Jarray, A. Karmouch, Decomposition approaches for virtual network embedding with one-shot node and link mapping, IEEE/ACM Trans. Network. 23 (3) (2015) 1012–1025.
[22] L. Gong, Y. Wen, Z. Zhu, T. Lee, Toward profit-seeking virtual network embedding algorithm via global resource capacity, in: IEEE INFOCOM, 2014, pp. 1–9.
[23] S.B. Masti, S.V. Raghavan, Vna: An enhanced algorithm for virtual network embedding, in: IEEE Computer Communications and Networks (ICCCN), 2012, pp. 1–9.
[24] I. Fajjari, N. Aitsaadi, G. Pujolle, H. Zimmermann, Vne-ac: Virtual network embedding algorithm based on ant colony metaheuristic, in: IEEE International Conference on Communications (ICC), 2011, pp. 1–6.
[25] T. Trinh, H. Esaki, C. Aswakul, Quality of service using careful overbooking for optimal virtual network resource allocation, in: International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2011, pp. 296–299.
[26] H. Kim, S. Lee, Greedy virtual network embedding under an exponential cost function, in: International Conference on Information Networking (ICOIN), 2012, pp. 216–221.
[27] Java implementation of a greedy algorithm for vertex cover problem, 2015. [Online]. Available: http://jgrapht.org/javadoc/org/jgrapht/alg/VertexCovers.html#findGreedyCover-org.jgrapht.UndirectedGraph.
[28] Mini-max algorithm java implementation, 2015. [Online]. Available: https://gist.github.com/vy/6580214.
[29] A.P. Punnen, A linear time algorithm for the maximum capacity path problem, Eur. J. Oper. Res. 53 (3) (1991) 402–404.
[30] Y. Zhu, M.H. Ammar, Algorithms for assigning substrate network resources to virtual network components., in: IEEE INFOCOM, 1200, 2006, pp. 1–12.
[31] Gt-itm, 2015. [Online]. Available: http://www.cc.gatech.edu/projects/gtitm/.

Cihangir Beşiktaş received the B.S. degree in computer engineering from Istanbul Technical University, Istanbul, Turkey, in 2009, the M.S. degree in computer engineering from Gebze Technical University, Kocaeli, Turkey, in 2012. He is currently studying his Ph.D. education in computer engineering at Gebze Technical University. He has been working as a Senior Researcher at Information Technologies Institute, TUBITAK BILGEM, Kocaeli, Turkey since 2009. His main research interests are software defined networking and cloud computing.

Didem Gözüpek received the B.S. degree (high honors) in telecommunications engineering from Sabancı University, Istanbul, Turkey, in 2004, the M.S. degree in electrical engineering from the New Jersey Institute of Technology (NJIT), Newark, NJ, USA, in 2005, and the Ph.D. degree in computer engineering from Bogazici University, Istanbul, Turkey, in 2012. She is an Associate Professor with the Computer Engineering Department, Gebze Technical University, Kocaeli, Turkey. From 2005 to 2008, she worked as an R&D Engineer in a telecommunications company in Istanbul. Her main research interests are structural and algorithmic graph theory, approximation algorithms, and optimization problems in communication networks. Dr. Gözüpek received the CAREER Award from the Scientific and Technological Research Council of Turkey (TUBITAK) in 2014, the Dr. Serhat Özyar Young Scientist of the Year Honorary Award in 2013, the Bogazici University Ph.D. Thesis Award in 2012, and ASELSAN PhD Fellowship in 2011. She was a finalist for the Google Anita Borg Memorial Scholarship in 2009.

Aydın Ulaş received his B.S., M.S. and Ph.D. degrees in computer engineering from Boğaziçi University, İstanbul, in 1999, 2001 and 2008 respectively. He worked as a post-doctoral researcher and assistant professor in Università degli Studi di Verona, Italy and worked on the FP7 project SIMBAD (Similarity based Pattern Analysis and Recognition). He is currently working as a senior researcher and solution architect for Argela technologies on SDN and NFV for the telecommunications industry. He is the Turkish consortium lead for the Celtic+ project SIGMONA and guest lecturing in various universities. His research interests include SDN&NFV, machine learning for networking, classifier combination, statistical comparison of classification algorithms, medical imaging, bioinformatics, and machine learning. He is a reviewer for several international conferences and journals and he is a senior member of IEEE Computational Intelligence Society and IAPR Turkish chapter (TÖTİAD).

Erhan Lokman received the B.S. degree in electronics and communication engineering from Istanbul Technical University, Istanbul, Turkey, in 1992, the M.S. degree in electronics and communication engineering from Istanbul Technical Univesity, Istanbul, Turkey, in 2005. From 1993 to 2000, he worked as an R&D Engineer and as a System Design Engineer in Alcatel (Istanbul TURKEY, Antwerpen Belgium and Dulles, Washington DC. From 2000 to 2003, he continued his technology survey in Oksijen Technology and led many cutting edge and field proven projects and products. He continues his carrier in Argela Technologies as R&D Director in Software Defined Networks and Network Function Virtualization areas. He has 20+ years of experience in Telecommunications and has extensive knowledge in Telecom Networks, Value Added Services, Fixed Mobile Convergence, VoIP, IMS and IPTV areas. He has also contributed in many patents and papers.