Virtual Network Embedding Algorithm for One-to-One Site Protection

Ashiq Khan, Xueli An, David Perez-Caparros, Wolfgang Kiess

DOCOMO Communications Labs Europe GmbH, Munich, Germany

Email: {khan, an_de_luca, caparros, kiess}@docomolab-euro.com

Abstract—Network redundancy and protection have become an even more serious issue after the last tsunami in Japan. Future networks need to be planned and operated keeping such unprecedented failures in mind. In this regard, we consider Virtual Network Embedding (VNE) techniques as an useful tool to realize an optimization between redundancy and resource consumption. However, conventional VNEs mostly concentrate on single network embedding. As VNE is a computationally intractable problem, relaxation methods are often used to offer a workable polynomial time solution. Some of such relaxation methods like flow-splitting do not fit well in real network operation. In this paper, we propose a heuristic VNE algorithm for network site protection without flow splitting. We also provide a solution with joint site and link embedding to achieve better bandwidth consumption. Evaluation results show that our VNE algorithm performs close to theoretical thresholds and consumes less link resources in delivering a VNE solution in polynomial time compared to the conventional algorithm.

I. INTRODUCTION

Network virtualization is receiving high popularity in both academia and industry [1]-[7]. From the industrial perspective, it enables a network operator to optimize its hardware, software, and transport resources, and to introduce new business models [1]. An important challenge here is to realize the embedding of a Virtual Network (VN) into a Physical Network (PN). Such problem is widely known as Virtual Network Embedding (VNE).

Virtualization allows for isolation, which enables multiple isolated VNs to coexist on the same PN. In order to realize a cost efficient coexistence of VNs, a VNE algorithm needs to consider different cost issues of a network, e.g. link bandwidth (BW), computation resources like CPU, memory, etc. A VNE problem is an NP-hard problem [2] and therefore, many heuristic and greedy algorithms [2]-[7] have been proposed to achieve a workable, polynomial time solution. However, a VNE problem has a number of features which have been mostly ignored in the present literature. It obviously consists of node embedding and link embedding to realize a VN. However, the definition of node has not yet been made clear. In the networking context, there are end-nodes, like servers, and there are switching nodes, like routers. Prior works consider that the nodes are homogeneous, both in the VN request and in the PN where the VN request would be embedded into. In reality, a server node cannot be embedded into a switching node in PN. In this paper, we distinguish between switching nodes (e.g. routers) and end-nodes/computational nodes (e.g. servers). We call these end-nodes as sites. Different sites are interconnected by switching nodes. We envision that such sites can be used to deploy future Telco services e.g. MME, Serving Gateway (SGW), Packet Data Network Gateway (PGW) [8], which require high availability. Hence, site protection is necessary to support such services. Moreover, site protection has become even more crucial after the last tsunami in Japan where all network operators experienced devastating network failures over large geographical areas.

Many works propose flow-splitting by using Multi-Commodity Flow (MCF) in order to realize a polynomial time solution [2][3]. Although it is valid in the theoretical context, there is no network layer (L3) routing protocol that can actually realize flow splitting in a real world scenario. Stream Control Transmission Protocol [9] can do that on an end-to-end basis. However, splitting a flow at an intermediate switching node has been so far not applicable in real world. The main reasons are the extra complexity that it brings into network operation, and packet re-ordering at the end-nodes. For these reasons, we avoid flow-splitting in this paper. We consider site protection without flow splitting in the switching domain. Our objective is to create an embedding algorithm which provides one-to-one site protection for all sites. For a network operator, all the sites are equally important. This means that the embedding algorithm needs to find out one backup site for each primary site in a VN request and ensure connectivity in such a way that any number of site failures could be accommodated. As we mentioned before, a simple VNE is already NP-hard. All-site protection, in practice, means embedding the VN request twice, with extra links to ensure primary to backup site connectivity. Considering a nation-wide commercial network, the available heuristic/greedy polynomial time VNE algorithms will face significant scalability problems in such case. In this work, we propose a heuristic algorithm which accomplishes VNE with site protection and ensures network connectivity during any number of simultaneous site failures.

The rest of the paper is organized as follows. Section II discusses related work. Section III presents our network model and formulates the VNE problem mathematically. In section IV, we present our site-protection VNE algorithm and provide the evaluation results in Section V. Section VI concludes the paper.

II. RELATED WORKS

In this section, we present the most relevant and recent works in terms of path splitting and network protection.
Authors in [2] have presented comprehensive VNE algorithms where the node embedding and the link embedding are well correlated. They relax the integer constraints of MIP to obtain a linear programming which can be solved in polynomial time for the node embedding. However, they use MCF algorithm with path/flow splitting to be able to complete the link embedding in polynomial time. We do not assume path/flow splitting in our work as it is not practically feasible in real networks.

Authors in [4][5][6] present different link protection VNE algorithms. A solution for single link failure has been presented in [4], as the authors consider multiple simultaneous link failures a rare event. As dedicated backup links are a waste of resources, they propose two shared link backup schemes, an on-demand one and a pre-allocated one. Authors in [5] take such shared backup path scheme one step further. They consider the migration of the end-nodes of a path to realize path protection. Authors in [6] propose a reactive backup method for protection against single link failures. Each physical link BW is divided and preserved for primary flow embedding and the rest for accommodating backup flows during failures. During a physical link failure, a reactive optimization is performed to reallocate the flows on the failed link in the rest of the network. In our work, we do not address link protection explicitly. The above-mentioned works could easily be used with our proposal.

The most relevant proposal to our work could be found in [7], as the authors address site protection. In [7], two site protection schemes are proposed. One scheme proposes one extra site to be shared for multiple primary site protection. In case of a site failure, the failed site is migrated to the shared backup site. The other one is an extension to this one-redundant embedding to multiple backup sites embedding. The entire PN can be modeled as a graph $G = (V, E)$, where $V$ and $E$ represent the set of vertices/nodes and the set of links within the PN, respectively. We use $A_p$ to denote the adjacency matrix of $G_p$, which is a $|V_p| \times |V_p|$ matrix. If there is an incident link between node $i$ and $j$, $A_p(i, j) = 1$; otherwise, $A_p(i, j) = 0$. Bidirectional graph is assumed in this work, hence $A_p(i, j) = A_p(j, i)$. A path from node $i$ to $j$ is denoted as $p(i \rightarrow j)$, which is the collection of the links along the path. The path length is given by $|p(i \rightarrow j)|$. $I$ and $J$ represent a set of sources and destinations, hence $p(i \rightarrow j)$ is the set of paths containing all the combinations of the sources and destinations. $P_p$ represents the set of all the feasible paths within $G_p$. If $\forall n_{p}^i, n_{p}^j \in V_p$, $p(i \rightarrow j) \in P_p$.

The set of switching nodes within the core transport network is given by $V_s$ and the set of site nodes are represented as $V_r$, hence we have $V_p = V_s \cup V_r$ on condition that $V_s \cap V_r = \phi$. Node capacity is specified by a $1 \times |V_p|$ array $C_p$, in which, the capacity of node $i$ is $C_p(i)$ and $\{C_p(i) \in \mathbb{R} : C_p(i) \geq 0\}$.

III. NETWORK MODEL AND PROBLEM DEFINITION

In future operational networks, a VNE request will come from a Virtual Network Operator (VNO) who wishes to provide service using the PN. Figure 1(a) shows an operational structure [1] with network Operational Support Systems (OSS). The PN consists of two components: the sites and the core transport network. The site nodes can be physical servers or Virtual Machines (VMs) which are used to host the services upon the request from a VNO. We envision that the site nodes with computation and storage capabilities can be used to host services like 3GPP core network nodes, e.g. MME, SGW, PGW, etc. The site nodes will be deployed in clouds in the PN as shown in Figure 1(a). The graphical interpretation of the scenario explained above is shown in Figure 1(b).
The link BW is specified by a \( |V_p| \times |V_p| \) matrix \( B_p \). If link \( e_p(i, j) \in E_p \), its available BW is given by \( B_p(i, j) \), whereas \( B_p(i, j) \in \mathbb{R}^+ \); otherwise, \( B_p(i, j) = 0 \). The available BW on a path \( p(i \rightarrow j) \) is denoted by \( f_p(i \rightarrow j) \), the value of which is limited by the intermediate link that has the minimum BW as, \( f_p(i \rightarrow j) = \min\{B_p(i, v_1), \ldots, B_p(v_p | i \rightarrow j | - 1, j)\} \), where \( v_i \) for \( \forall i \in [1, |p(i \rightarrow j)| - 1] \), is an intermediate node on the path.

### B. Virtual Network Operator Request

The VNO request can come in different granularities. If a VNO wishes to operate its virtual network at router level and above, it can send a request in that detail. As this work is about site protection, we assume that a VNO sends its embedding request at site granularity. A VNO request consists of VNO nodes and links, which can also be represented by an undirected graph \( G_v = (V_v, E_v) \) with adjacency matrix \( A_v \). Besides the VN topology, the VNO request also provides the requirement constraints in terms of node capacity \( C_v \), link bandwidth \( B_v \), and delay limits \( D_v \). Delay can be defined by round trip time (RTT) or hop count. We assume that the dominant factor of delay is the processing time when packets pass through a switching node, hence we use hop count as a parameter to model delay. Figure 1(b) provides an example of a VNO request, which consists of four nodes with capacity, link BW, and delay requirements. Once a VNO request arrives, it will be mapped to the PN, which means that a number of site nodes will be selected within the PN, and the bandwidth among the site nodes will be reserved for hosting service and communication purposes.

To host Telco services like 3GPP core network nodes in VNs, carrier-grade availability is compulsory. The embedded VNs should be survivable and recoverable from site failures. Providing one-to-one backup for all the protected sites is one solution to achieve this required high availability. Therefore, along with one primary site, one backup site is also needed for mapping a VNO node. The backup site can be hot-standby and the running service states are synchronized between the primary and backup sites all the time, hence the backup site can take over the service immediately without interruption if the primary site is down. However, the backup mechanism is out of the scope of this work. In this work, we only focus on how to select and inter-connect the backup and primary sites.

### C. Problem Formulation

The VNE problem can be considered as a process with two stages: VNO node mapping and VNO link mapping. In the first stage, VNO nodes are mapped to site nodes in the PN using function \( M_v : V_v \rightarrow V_r \). To achieve one-to-one site protection for VNO node \( i \), a primary site node \( n_p^i \) and a backup site node \( n_b^i \) are selected together for VNO node mapping:

\[
M_v(e_v(i, j)) = \begin{cases} 
(n_p^i, n_b^i), & \forall n_v^i \in V_v, \forall n_p^i, \forall n_b^i \in V_r, n_p^i \neq n_b^i \end{cases}
\]

In the second stage, the feasible paths between all the mapped site nodes are established by using function \( M_l : E_v \rightarrow P_p \), where

\[
M_l(e_v(i, j)) = p(M_v(n_p^i) \rightarrow M_v(n_b^i)), \forall n_v^i, n_b^i \in V_v
\]

To guarantee the seamless service migration in the site failure scenario, for instance for a VNO request with two nodes and one link as shown in Figure 2 (a), we have to explicitly search for two primary and two backup nodes, and six links in-between all the primary and backup nodes to enable one-to-one site protection as illustrated in Figure 2 (b). This is the conventional method to handle one-to-one site protection. Hence, the total required BW to embed one VNO link is the summation of the reserved BW on all the six links. The required backup BW for the path between the primary and backup nodes might be smaller than the requested BW on the primary path, which depends on the selected backup mechanism as we mentioned before or the explicit request from the VNO. For analysis simplicity, we assume that the required backup BW is the same as the requested BW for the primary path. \( M_l(e_v(i, j)) \) defined in (2) is a path set with two kinds of paths: communication path set \( M_l^C(e_v(i, j)) \) and backup path set \( M_l^B(e_v(i, j)) \), where \( M_l^C(e_v(i, j)) = \{ p(i_p \rightarrow j_b), p(i_b \rightarrow j_b), p(i_p \rightarrow j_b), p(i_b \rightarrow j_b) \} \) and \( M_l^B(e_v(i, j)) = \{ p(i_p \rightarrow j_b), p(j_b \rightarrow j_b) \} \). The VNO request embedding issue can be formulated as an optimization problem as following:

**Objective:**

\[
\begin{align*}
\text{minimize} & \quad \sum_{x_v(i, j) \in E_v} B_v(i, j) \left( \sum_{p(x \rightarrow y) \in M_l^C(e_v(i, j))} |p(x \rightarrow y)| \right. \\
& \left. + \sum_{p(x \rightarrow y) \in M_l^B(e_v(i, j))} |p(x \rightarrow y)| \right) \\
\end{align*}
\]

**Resource Constraints:**

\[
C_v(i) \leq \min(C_v(i_p), C_v(i_b)), \forall n_v^i, n_b^i \in V_v
\]

\[
f_p^i - f_p^b \geq \sum_{j=1, j \neq i}^{V_v} B_v(i, j), \forall n_v^i, n_b^i \in V_v
\]

\[
f_p^i - f_p^b \geq B_v(i, j), \forall p(x \rightarrow y) \in M_l^C(e_v(i, j))
\]

\[
|x_v(i, j) - x_v(i_p, j_b)| \leq D_v(i, j), \forall p(x \rightarrow y) \in M_l^C(e_v(i, j))
\]

\[
\sum_{x_v(i, j) \in E_v} \left( \sum_{p(x \rightarrow y) \in M_l^C(e_v(i, j))} B_v(i, j) \right) + \sum_{j=1, j \neq i}^{V_v} B_v(i, j) \leq B_v(u, v)
\]

**Node and Link Constraints:**

\[
x_{i, j} \in \{0, 1\}, \forall n_v^i \in V_v, \forall n_v^j \in V_v
\]

\[
x_{i, j} = 1, x_{i, j_b} = 1, \sum_{j=1, j \neq i}^{V_v} x_{j, i_p} = 0 \quad \text{and} \quad \sum_{j=1, j \neq i}^{V_v} x_{j, i_b} = 0
\]

\[
\bar{x}_{uv} \in \{0, 1\}, \forall n_v^u, n_v^v \in V_v, \forall p(u, v) \in E_p
\]

\[
\sum_{i=u, p(k, j) \in E_p} \bar{x}_{ik} - \sum_{j=u, p(k, j) \in E_p} \bar{x}_{kj} = \begin{cases} 
-1, & k = s, n_v^i \in V_r \\
1, & k = d, n_v^j \in V_r \\
0, & k \notin V_r
\end{cases}
\]

Equation (4) represents the capacity constraint from a VNO request, which means that the selected primary and backup site nodes should have sufficient capacity to host it. Equation (5) is the BW constraint for the path between the primary and backup site nodes, which are mapped from VNO node \( i \).
implies that the reserved BW for this path is the summation of all the incoming and outgoing traffic from VNO node i. Equation (6) and (7) are the BW and delay constraints for the communication paths. Equation (8) implies that the total BW of all the flows passing through the physical link \( u \rightarrow v \) is limited by its available BW \( B_p(u, v) \), in which \( 1_A(a) \) is an indicator function and \( 1_A(a) = 1 \) if \( a \in A \), otherwise, \( 1_A(a) = 0 \).

In equation (9), \( x_{i,j} \) is a binary variable, which is 1 if site node \( j \) is selected as a primary or a backup node for VNO node \( i \). Otherwise, it is zero. Equation (10) ensures that one site node can only accommodate one VNO node (primary or backup) for one VNO request. \( \bar{x}_{uv} \) introduced in equation (11) is also a binary variable which is equal to the indicator function \( 1_p(s \rightarrow d)(E_p(u, v)) \) defined in equation (8). Equation (12) limits that, for all the intermediate switching nodes on the path \( p(s \rightarrow d) \), the number of incoming links is equal to the number of outgoing links.

IV. OUR PROPOSAL

One-to-one site protection is expensive in terms of bandwidth. Because the required bandwidth is not only reserved from the primary to primary sites, but also the path between primary to backup and backup to backup sites to handle any number of primary site failures. Therefore, our main objective is to reduce the bandwidth consumption to embed the VNO requests.

At first, we select potential candidate site nodes in the PN with enough resources to accommodate the requested nodes. In the second step, we form primary-backup site node pairs based on a predefined network distance in between. Once such candidate pairs for each VNO node have been selected, the problem space is significantly reduced. Then, we embed the links among the pairs which satisfy both the BW and delay requirements from the VNO request. Multiple candidate site nodes give the flexibility to find out viable paths and optimize BW consumption.

A. Primary-Backup Pair Searching

We consider all the computing and storage resources (physical/virtual machines) connected to the same access router as one logical site node and we assume that each switching node does not attach more than one site node. To be fault-tolerant, a VNO node needs to be mapped to two site nodes to form a primary-backup pair. Therefore, the path length between the primary and backup site nodes should be equal or higher than 3. Moreover, we limit the path length between the primary and backup site nodes by a threshold \( d_{th} \). Hence, we have one more constraint for objective (3): For a candidate primary-backup pair of VNO node \( i \), we have

\[
3 \leq |p(i_p \rightarrow i_b)| \leq d_{th}.
\]  

The primary-backup node distance constraint \( d_{th} \) is a network operational parameter which can be determined by a VNO. It should be noted that \( d_{th} \) is not a compulsory constraint. In order to geographically distribute primary and backup sites, a backup site node can also be considered even though the path length between the primary-backup pair is longer than \( d_{th} \). However, a large value of primary-backup pair path length may increase communication cost for service backup to achieve site protection and network downtime while migration from primary to backup sites due to longer network distance.

Searching all the candidate primary-backup pairs for all the VNO nodes is time and computing resource consuming, and it is also unnecessary especially for a large scale PN. Greedy node mapping algorithm is applied here for primary and backup site nodes mapping. For a VNO node \( i \), its capacity requirement is \( C_{vi}(i) \) and its BW requirement is the summation of the BW required from all its incident links \( \sum_{j=1}^{n_v} B_l(i,j) \). We first sort the VNO nodes according to their required capacity in decreasing order as \( v_i = \{n_{vi}^1, n_{vi}^2, ... n_{vel}^l\} \) and use this sequence to map the VNO nodes to the site nodes in the PN. The rationale behind this is that it is more difficult to embed a node with high capacity requirement than a low capacity VNO node. Site nodes with sufficient capacity are considered as candidates for VNO node mapping and any two candidate site nodes form a candidate primary-backup pair. However, searching the optimized primary-backup pair is complex in terms of computing resource and time. Geographical constraints could be added to assist site node selection like in [2], but they are not considered here due to space constraints. We simply limit the number of selected site nodes per each VNO node by a fixed number \( n_{nm} \) (e.g. \( n_{nm} = 5 \)). In the next step, the candidate pairs are sorted according to the hop count in increasing order. If several pairs have the same hop count, the pair with higher capacity is put on top of the pair with lower capacity. We choose the first \( n_{th} \) pairs as the selected primary-backup pairs. If the number of candidate pairs is smaller than \( n_{th} \), all the pairs are selected. For a VNO node \( i \), it has \( n_i \) candidate primary-backup pairs, where \( n_i \leq n_{th} \). All the site nodes in the candidate primary-backup pairs is given by \( V_{ph} = \bigcup_{i=1}^{n_i} \{i_{ph}, i_{bs} \} \), where \( i_{ph} \) and \( i_{bs} \) represents the primary and backup nodes from the \( x^{th} \) candidate pair respectively. Primary and backup nodes are randomly decided within the primary-backup pairs. We assume that each site node can only be mapped to one VNO node within one VNO request. Therefore, after establishing \( V_{ph} \), all the nodes within \( V_{ph} \) are reduced from the site nodes as \( V_s = V_c \cap (V_{ph})^c \), where \((V_{ph})^c\) is the absolute complement of \( V_{ph} \). The searching of primary-backup pairs for the rest of VNO nodes continues within the updated set of \( V_c \) until all the VNO nodes form
their own primary-backup pair sets. Primary-backup paths are found in this step as described above. By combining them with the primary-primary paths, we ensure a complete connectivity between all nodes in two primary-backup pairs. The pseudo code of this Primary-Backup Pair Searching (PBPS) algorithm is shown in Algorithm 1.

Algorithm 1 The PBPS algorithm

1: procedure PBPS($G_v$, $G_e$)
2: \[\begin{align*}
    l_{nm} & \leftarrow \text{sort } V_r \text{ (site capacity)} \\
    & \text{for } x \leftarrow 1, |V_r| \text{ do} \\
    & \quad l_{nm} \leftarrow \phi \\
    & \quad i' \leftarrow l_{x}(x) \\
    & \quad \text{for all } n_i \in V_r \text{ do} \\
    & \quad \quad \text{if } C_0(i') \geq C_0(i) \text{ then} \\
    & \quad \quad \quad l_{nm} \leftarrow l_{nm} \cup i \\
    & \quad \quad \text{end if} \\
    & \quad \quad \text{if } |l_{nm}| \leq n_{nm} \text{ then} \\
    & \quad \quad \quad \text{break} \\
    & \quad \quad \text{end if} \\
    & \quad \text{end for} \\
    & \quad r \leftarrow 1, l_{\phi}(x) = \phi \\
    & \text{for } j \leftarrow 1, |l_{nm}| \text{ do} \\
    & \quad \text{for } k \in [1, |l_{nm}|] \text{ do} \\
    & \quad \quad \text{if } s = \frac{p(l_{\phi}(x) \to l_{\phi}(j))}{d_{ij}} \leq d_{ij} \text{ then} \\
    & \quad \quad \quad l_{\phi}(x) \leftarrow l_{\phi}(x) \cup \{l_{nm}(i), l_{nm}(j)\} \\
    & \quad \quad \text{end if} \\
    & \quad \quad \text{if } l_{\phi}(x) = n_{th} \text{ then} \\
    & \quad \quad \quad \text{break} \\
    & \quad \quad \text{end if} \\
    & \quad \text{end for} \\
    & \quad \text{end for} \\
    & \text{end procedure}
\end{align*}\]

B. VNO Link Embedding

After selecting the primary-backup pairs for the VNO nodes, the VNO links between VNO nodes are mapped to the paths on the PN.

1) Node Degree Dependent Link Embedding: VNO links are mapped sequentially based on the node degree of the VNO nodes. First, we sort the VNO nodes according to their node degrees in a decreasing way $V_i = \{n_i^0, n_i^1, \ldots, n_i^{m_i-1}\}$. The VNO links are mapped from higher node degree VNO nodes to lower node degree VNO nodes. We assume that the node degree of VNO node $i$ is $k$ and the node IDs of its neighbors are denoted as $i^1, i^2, \ldots, i^k$. All the VNO links that pass through the same VNO node, e.g., $e_{i_x}(i, i^2)$ for $x \in [1, k]$, have the same priority for mapping. In our algorithm, we sort $i^x$ in increasing order and map $e_{i_x}(i, i^2)$ using the sorted neighbor sequence. In the next step, the shortest paths between all the candidate pairs are established. Constraint-based Shortest Path First (CSPF) [10] can be used to determine the path with sufficient BW between all pairs. If the path length of a candidate pair cannot satisfy (13), this pair is removed from the candidate pair set. For a certain VNO node, the BW constraint for its primary-backup pair is the summation BW of all links passing through this VNO node. This has two effects. Firstly, it correlates the node embedding with the link embedding. There is no point of selecting a node as the candidate which only has enough node capacity but not enough BW for ingress/egress links. Secondly, as will be explained below, we only find out primary-primary paths in the PN for inter VNO node link embedding.

2) Joint Primary and Backup Paths Embedding: Using the conventional one-to-one site protection approach as shown in Figure 2 (b), a mesh needs to be created among primary and backup nodes to ensure connectivity during any number of site failures. For example, for the VNO request of two connected nodes $A$ and $B$, in the conventional approach, we need to find out $A$ and $B$, their respective backup nodes $A'$ and $B'$ and then interconnect all four of these. In our approach, we only connect the primary-backup pairs $A-A'$ and $B-B'$, and then we connect the primary nodes $A$, $B$, as shown in Figure 2 (c). In this way, we reuse a part of the primary-primary path to ensure connectivity among all 4 nodes if both $A$ and $B$ fail simultaneously or if any of them fails. The rationale behind this is, if the primary and backup sites are not too far away from each other (which would perform badly while migrating from primary to backup nodes), the paths in between two primary nodes and their backups would overlap for a large section. Therefore, to map two VNO nodes, finding out three paths would suffice. If $B$ fails, the path can be switched to $B'$ from the intersection point of $A-B$ and $B-B'$. Path searching between primary sites is a standard routing problem, which can be solved by many existing shortest path algorithms [10]. CSPF is used in this work for route discovery in the PN, in which the required BW is the constraint. If the shortest path found in this way does not meet the delay requirement, we conclude that the other paths wont.

3) Embedded Link Adjustment due to Bandwidth Limitation: After mapping the VNO nodes, the VNO link embedding process may not be successful in one trial. For instance, site node $i$ and $j$ are selected for VNO node mapping, but there is no path that can be found between them after running CSPF because of BW scarcity. If this scenario occurs during the VNO link embedding process, instead of using the BW constraint, we find out all paths between site nodes $i$ and $j$ that meet the delay constraint. We first take the shortest one and check if it overlaps with any previously embedded paths. If there are overlapping links, we check if releasing the previously embedded path could help to release enough BW to embed the current path. If not, we go to the next
shortest path between site nodes \( i \) and \( j \), and do the same as above. If releasing the previously embedded path can help to embed the current one, we release the previous one, embed the current VNO link and re-embed the previous VNO link as has been explained before. In order to avoid an algorithm loop, we mark all the overlapped links in the PN. If further VNO link embedding overlaps on such marked links again, we exit the algorithm and declare a failure. This is because releasing a previously embedded path going through the marked links means that the released path will not have any other option to be embedded. If all the paths satisfying delay constraints do not overlap with any previously embedded paths, path searching fails, which means that there is no path available in the PN with enough BW to meet the requirement.

V. SIMULATION AND PERFORMANCE EVALUATION

A. Simulation Settings

1) Physical Network: To evaluate the performance of our algorithm, we have implemented a MATLAB based discrete event simulator. We randomly generate \( |V_p| \) switching nodes in a circular area to form the core transport network. Each switching node selects the five closest neighbors in terms of distance as its direct neighbors. We also generate \( |V_r| \) site nodes, which are randomly attached to one of the switching nodes. Any two site nodes cannot attach to the same switching node. The major simulation related parameters are listed in Table I. The capacity and bandwidth of each site node and switching node are uniformly distributed within the range \([100, 300]\) and \([100, 200]\) respectively.

2) VNO Request: For one VNO request, the number of requested VNO nodes is uniformly distributed between \([2, 5]\). The probability of connectivity between every two VNO nodes is 0.5. If all the nodes within the VNO graph are not connected via single or multiple hops, the VNO graph is regenerated. The bandwidth and capacity requirement is uniformly distributed within the range \([1, 5]\). The delay limit for a VNO request is set to 10 hops. The path length limit between a pair of primary and backup site nodes is 4 hops. The number of selected primary and backup candidate pair \( r_{th} \) is set to 1.

B. Performance Evaluation

In this section, we evaluate the performance of our proposed algorithm to achieve one-to-one site protection. We simulate the arrival of VNO requests as discrete events. To test the maximum number of VNO requests that can be embedded in a PN, we do not define the lifetime of a VNO request. Hence, once a VNO request is embedded in the PN, it will remain in the network for the rest of the simulation.

V. Simulations and Performance Evaluation

A. Simulation Settings

1) Physical Network: To evaluate the performance of our algorithm, we have implemented a MATLAB-based discrete event simulator. We randomly generate \(|V_p|\) switching nodes in a circular area to form the core transport network. Each switching node selects the five closest neighbors in terms of distance as its direct neighbors. We also generate \(|V_r|\) site nodes, which are randomly attached to one of the switching nodes. Any two site nodes cannot attach to the same switching node. The major simulation related parameters are listed in Table I. The capacity and bandwidth of each site node and switching node are uniformly distributed within the range \([100, 300]\) and \([100, 200]\) respectively.

2) VNO Request: For one VNO request, the number of requested VNO nodes is uniformly distributed between \([2, 5]\). The probability of connectivity between every two VNO nodes is 0.5. If all the nodes within the VNO graph are not connected via single or multiple hops, the VNO graph is regenerated. The bandwidth and capacity requirement is uniformly distributed within the range \([1, 5]\). The delay limit for a VNO request is set to 10 hops. The path length limit between a pair of primary and backup site nodes is 4 hops. The number of selected primary and backup candidate pair \(r_{th}\) is set to 1.

B. Performance Evaluation

In this section, we evaluate the performance of our proposed algorithm to achieve one-to-one site protection. We simulate the arrival of VNO requests as discrete events. To test the maximum number of VNO requests that can be embedded in a PN, we do not define the lifetime of a VNO request. Hence, once a VNO request is embedded in the PN, it will remain in the network for the rest of the simulation.

V. SIMULATION AND PERFORMANCE EVALUATION

A. Simulation Settings

1) Physical Network: To evaluate the performance of our algorithm, we have implemented a MATLAB-based discrete event simulator. We randomly generate \(|V_p|\) switching nodes in a circular area to form the core transport network. Each switching node selects the five closest neighbors in terms of distance as its direct neighbors. We also generate \(|V_r|\) site nodes, which are randomly attached to one of the switching nodes. Any two site nodes cannot attach to the same switching node. The major simulation related parameters are listed in Table I. The capacity and bandwidth of each site node and switching node are uniformly distributed within the range \([100, 300]\) and \([100, 200]\) respectively.

2) VNO Request: For one VNO request, the number of requested VNO nodes is uniformly distributed between \([2, 5]\). The probability of connectivity between every two VNO nodes is 0.5. If all the nodes within the VNO graph are not connected via single or multiple hops, the VNO graph is regenerated. The bandwidth and capacity requirement is uniformly distributed within the range \([1, 5]\). The delay limit for a VNO request is set to 10 hops. The path length limit between a pair of primary and backup site nodes is 4 hops. The number of selected primary and backup candidate pair \(r_{th}\) is set to 1.

B. Performance Evaluation

In this section, we evaluate the performance of our proposed algorithm to achieve one-to-one site protection. We simulate the arrival of VNO requests as discrete events. To test the maximum number of VNO requests that can be embedded in a PN, we do not define the lifetime of a VNO request. Hence, once a VNO request is embedded in the PN, it will remain in the network for the rest of the simulation.
results shown in Figure 3 indicate that our proposed algorithm consumes much less bandwidth to embed a VNO request and it can also accept more VNO requests than the conventional algorithm due to its higher resource allocation efficiency.

3) Path Length: With the same simulation setup described in the previous subsection, we depict the total primary path length (paths from primary site to primary site) and backup path length (paths from backup site to backup site) for each VNO request. The results are shown in Figure 4. As we can see, by using our proposed algorithm, the path length of the backup path is always equal or longer than the primary path. This phenomenon occurs because, after assigning the primary-backup pairs for a VNO request, we first select the shortest path in-between the primary-primary site nodes which satisfies the BW requirement. By doing so, the primary path is always the optimized solution. Due to the fact that our proposed algorithm tries to reuse part of the primary path to reduce the bandwidth reservation for backup paths, the selection of backup paths might not be optimized. This may result in longer backup path lengths than the primary paths. In this work, we do not optimize the backup path length. During a failure and the consequent service migration, the path to the backup node could be optimized step-by-step. In such case, a new primary-backup site pair will be formed, or the failed primary site node can become the backup site node after its recovery and an unnecessary migration back to it can be avoided. We will address this in our future work.

4) Physical Network Scale Influence: The topology of a PN will also influence the VNE performance to a certain extent. To review the PN scale influence, we apply a core transport network with a grid topology. The number of switching nodes is set to $5 \times 5$, $6 \times 6$, $7 \times 7$ and $8 \times 8$ in different simulation scenarios. The number of site nodes is fixed to 20 and they are randomly attached to the switching nodes. The results shown in Figure 5 are the mean value after 100 iterations in order to average the randomness effect. The number of VNO requests is set to 50 in each run of the simulation. To obtain the VNO acceptance ratio, we divide the number of accepted requests by the total number of requests. As shown in Figure 5 (a), our proposed algorithm has a much higher VNO request acceptance ratio than the conventional approach for VNE. When the network size is increased, the acceptance ratios drop in both cases, which is more obvious by using our proposed algorithm than the conventional algorithm. There are two reasons for VNO request drop. The number of site nodes within the PN is fixed. Once the core transport network becomes larger, the average distance between site nodes is also increased. Hence, the first reason for VNO request drop is due to the fact that the path length between the selected site nodes cannot satisfy the delay constraint specified by the simulation. This is the direct influence from the PN topology. The second reason for VNO request drop is simply because the BW within the PN is not sufficient to host the VNO requests. When we further investigate the reasons of the VNO request drop by using these two algorithms, we find out that the VNO request drop for our algorithm is mainly due to the first reason and for the conventional one is mainly due to the second reason. As shown in Figure 5 (b), we depict the average path length for the embedded VNO requests with different PN sizes. As we discussed above, due to the distribution of site nodes, the path length tends to be longer in our proposed algorithm.

VI. CONCLUSION

In this paper, we have proposed and evaluated a heuristic algorithm which realizes VNE for one-to-one site protection. We do not relax the problem by using flow-splitting as that is unrealistic in operational networks. Our algorithm achieves sound correlation between the node and link embeddings and consumes less bandwidth than the conventional approach. The evaluation results also show that our algorithm performance in finding out a VNE solution is very close to the theoretical ceiling. In our future work, we will address VNO node request mapping over multiple sites for further site resource optimization, and backup path optimization for further reduction of link resources consumption.

REFERENCES