Embedding Virtual Networks in Flexible Optical Networks with Sliceable Transponders

Juzi Zhao[1][0000−0001−6825−9666] and Suresh Subramaniam[2][0000−0003−1501−5953]

[1] San Jose State University, San Jose, CA 95192, USA
juzi.zhao@sjsu.edu
[2] The George Washington University, Washington, DC 20052, USA
suresh@gwu.edu

Abstract. Emerging inter-datacenter applications involving data transferred, processed, and analyzed at multiple data centers, such as virtual machine migrations, real-time data backup, remote desktop, and virtual data centers, can be modeled as virtual network requests that share computing and spectrum resources of a common substrate physical inter-datacenter network. Recent advances make flexible optical networks an ideal candidate for meeting the dynamic and heterogeneous connection demands between datacenters. In this paper, we address the static (offline) version of the virtual network embedding problem in flexible optical networks equipped with sliceable bandwidth variable transponders (SBVTs). The objective is to minimize the total number of required SBVTs in the network. An Integer Linear Programming (ILP) formulation is presented, lower bounds are derived, and four heuristics are proposed and compared. Simulation results are presented to show the effectiveness of the proposed approaches.

Keywords: Virtual networks · flexible optical networks · inter-datacenter networks · embedding · frequency slots · list scheduling.

1 Introduction

Flexible (or elastic) optical networks (FONs), with a fine-grained spectral spacing and variable channel center frequencies are promising for connecting geo-distributed datacenters due to their capability to transfer large amounts of data as well as efficiency and flexibility of utilizing spectrum resources [1, 2].

Network virtualization breaks the network's rigidly fixed pattern by enabling different virtual networks with various requirements to share the same substrate physical network. Each virtual network (VN) request is a logical topology including a set of virtual nodes interconnected by virtual links. These requests require not only communication resources on the optical links but also compute and storage resources at the physical nodes in the network. Emerging inter-datacenter applications, such as virtual machine migration, database replication/backup, and virtual data centers, can be modeled as VN requests. The problem of allocating route(s) as well as appropriate spectrum, computing, and storage resources to a request or set of requests is commonly known as virtual network mapping/embedding problem (VNM/VNE), which is proved to be NP-hard [3].

There are several papers on VNE in FONs. The authors of [4] consider both static and dynamic cases, with the objective of maximizing spectrum slot utilization in static case and minimizing blocking ratio in dynamic case. Traffic grooming is taken into account in VNE problem in [5] over both WDM networks and FONs. In [6], an ILP formulation and heuristics to minimize the total spectrum slot usage are proposed for both transparent and opaque VNs. Two ILP formulations for VNE problem are presented in [7]. The authors of [8] take fragmentation-awareness and load-balancing features into account for the VNE, with an ILP formulation proposed. The authors of [9] propose relaxed ILP and heuristics for static version of VNE based on re-optimized VCAT framework. An ILP formulation to maximize the number of successful embeddings over transparent FONs considering transmission distance limits is proposed in [10]. The authors of [11] develop an ILP and a heuristic algorithm to minimize the total network cost considering impairment constraints and single link failures. The authors of [12] construct an auxiliary graph and utilize spectrum partitioning approaches to solve the cost-effective survivable VNE problem. Energy efficient VNE schemes are proposed for dynamic transparent VN requests in [13].

Sliceable Bandwidth Variable Transponders (SBVT, also called Multi-Flow Transponders) have been recently adopted in FONs. Multiple lightpaths with different destinations can be launched by a single SBVT, and thus transponder resources are effectively utilized. Many research directions in SBVT-enabled FONs have been explored. The authors of [14] propose dynamic routing, spectrum and transponder assignment schemes for two different types of SBVTs. The authors of [15] devise a dynamic distance-adaptive routing, spectrum, and modulation format assignment algorithm to optimize the utilization of spectrum and SBVTs. A model based on auxiliary graph is proposed in [16] to investigate the dynamic routing and spectrum assignment problem with traffic grooming.

In this paper, we consider the static version of VNE problem in FONs with SBVT allocation that has not been addressed before, to the best of our knowledge. A set of VN requests is given, where each request consists of a set of virtual nodes and the amount of computation resources required by virtual nodes, and bit rate required by virtual links connecting virtual nodes. The objective is to minimize the total number of required SBVTs for all requests. We present an ILP formulation, two lower bounds, and four heuristics based on List Scheduling and Tabu Search to embed the VN requests onto the physical optical inter-datacenter network, and assign VMs, frequency slots, and SBVTs to the requests.

2 Model and Problem Statement

The physical inter-datacenter network topology is $G_p(V_p, E_p)$, where V_p is the set of nodes (each is a data center associated with an optical switch), and E_p is the set of links. Each physical node has H virtual machines (VMs). Each SBVT at one physical node has C carriers. There are two fibers on each physical link with opposite directions, each has S frequency slots. The bandwidth of each frequency slot is 12.5 GHz. A number of frequency slots are allocated to each lightpath according to its spectrum requirement. The frequency slots allocated to each lightpath have to be contiguous (spectrum contiguity constraint), and the same band of frequency slots has to be allocated on each link traversed by the lightpath (spectrum continuity constraint). A guardband of G consecutive frequency slots has to be assigned between adjacent frequency slot bands allocated to different lightpaths if they share common physical links, in order to avoid interference [17]. K shortest distance paths between each pair of nodes are precomputed. One of M possible modulation formats is assigned to each lightpath. Each modulation format m has a maximum transmission reach R_m associated with it. A lightpath assigned modulation format m must has path distance less than R_m. Depending on the path length, each precomputed path has a highest modulation level that can be used. The physical network is considered as transparent, therefore, no spectrum converters exist in the network.

A VN request i can be represented as a graph $G_l^i(V_l^i, E_l^i)$, where V_l^i is the set of virtual nodes, and E_l^i is the set of virtual links. For clarity, we drop subscript l which indicates a virtual or logical topology. There is a bit rate requirement Λ_k^i for each virtual link $k^i \in E^i$ for bidirectional communication between the two end virtual nodes. Each virtual node j^i can be mapped to one physical node of a candidate physical node set (N_j^i), and requires h_j^i VMs. According to the candidate physical node sets of two end virtual nodes, virtual link k^i has a candidate path set P_k^i. Path $p \in P_k^i$ has a frequency slot requirement $b_k^i(p)$, based on the path's highest modulation format m^p and Λ_k^i, i.e., $b_k^i(p) = \lceil \Lambda_k^i / (s_p {*} 12.5) \rceil$, where s_p is spectrum efficiency of m^p in Gbps/GHz. In this paper, we assume two virtual nodes of the same VN request can be mapped to one physical node as long as the VM resources permit. In this case, there are no frequency slots nor SBVTs allocated to the virtual link connecting these two virtual nodes. (We note that the proposed approaches can be easily modified if this assumption does not hold, by adding an additional constraint that no two virtual nodes of the same request can be mapped to a single physical node.)

In this paper, we adopt the multi-laser sliceable bandwidth variable transponder (SBVT) model [14]. Each SBVT has C carriers. There is a dedicated tunable laser associated with each carrier, and the bandwidth of each carrier is U GHz. A single lightpath can be assigned 1 to C carriers based on its bandwidth requirement. If more than one carriers are assigned to a single lightpath, a 2 GHz guardband will be allocated between two carriers due to the center-frequency instability issue with lasers. Note that if multiple lightpaths are assigned the same SBVT at a particular node, then the bandwidths allocated to these lightpaths cannot have any overlap. This leads to the idea of extra links in the physical network as shown in Fig. 1 (only three SBVTs at node A are shown in the figure for illustration purpose). The original network has 9 links. The links between SBVTs and physical nodes can be treated as extra links. All lightpaths assigned SBVT a at node A will transverse extra link 10. Recall that for FONs, a single

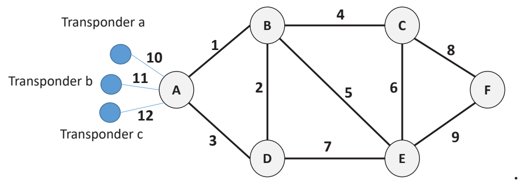

Fig. 1. Extra links introduced by transponders.

frequency slot on each link can be assigned to at most one lightpath (spectrum clash constraint). This constraint applies to both physical links and extra links.

In this paper, we consider static VNE problem where a set of requests are given *a priori*. Each virtual node in a request is mapped to a single physical node, with sufficient available VMs, selected from its candidate physical node set. Each virtual link is mapped to a physical path p, and a band of contiguous frequency slots is allocated to the virtual link based on its bit rate requirement. Two SBVTs on source and destination nodes, respectively, of path p have to be assigned to launch and terminate lightpath; and a corresponding number of carriers on the two assigned SBVTs are allocated to the virtual link. The objective is to minimize the total number of required SBVTs to map all requests.

3 Algorithms

In the following, we use pnode, vnode, plink, vlink, ppath, elink, and fslot to stand for physical node, virtual node, physical link, virtual link, physical path, extra physical link, and frequency slot, respectively. In addition, we use the letter i for virtual request index, j for vnode index, k for vlink index, n for pnode index, p for ppath index, and e for elink index.

3.1 ILP Formulation

Table 1. Notations

Symbol	**Meaning**
x_{ijn}	$x_{ijn} = 1$ if vnode j in request i is mapped to pnode n
y_{ikp}	$y_{ikp} = 1$ if vlink k in i is mapped to ppath p, where p could be a dummy path with source pnode and destination pnode being a same pnode
u_{ikq}	$u_{ikq} = 1$ if k in i uses SBVT q on its source pnode
v_{ikq}	$v_{ikq} = 1$ if k in i uses SBVT q on its destination pnode
τ_{iks}	$\tau_{iks} = 1$ if the starting fslot for vlink k in i is fslot s
g_{iks}	$g_{iks} = 1$ if vlink k in request i uses fslot s
z_{ike}	$z_{ike} = 1$ if vlink k in request i uses elink e
ϕ_{ikls}	$\phi_{ikls} = 1$ if vlink k in request i uses fslot s on plink l
γ_{ikes}	$\gamma_{ikes} = 1$ if vlink k in request i uses fslot s on elink e
θ_{iknq}	the number of carriers required by k in i at SBVT q of n
m_e	$m_e = 1$ if elink e is used by some request

There are several input parameters used in the ILP: a_{ijn} (binary), $a_{ijn} = 1$ if pnode n is in the candidate set (N_j^i) of pnodes to map to for vnode j in request

i. h_j^i (integer), the number of VMs required by j in i. H_n (integer), the number of VMs on n. $S(p)$ (integer), source pnode of ppath p. $D(p)$ (integer), destination pnode of p. $S(k^i)$ (integer), source vnode of vlink k in request i. $D(k^i)$ (integer), destination vnode of k in i. $b_k^i(p)$ (integer), the number of fslots (including guardband) required by k in i if it is mapped to ppath p. $T(e)$ (integer), the SBVT with elink e. $B(e)$ (integer), the pnode with e. $c_k^i(p)$ (integer), the number of carriers required by k in i if it is mapped to p. η_{pl} (binary), $\eta_{pl} = 1$ if p transverses plink l. The variables in ILP formulation are shown in Table 1.

From Fig. 1, it can be seen that each elink is a SBVT, so minimizing total number of required SBVTs is minimizing the total number of used elinks.

Objective: Minimize $\sum_e m_e$

Constraints:
(1) $\sum_n x_{ijn} = 1$ for all i, j; (2) $\sum_n a_{ijn} x_{ijn} = 1$ for all i, j
(3) $\sum_{ij} h_j^i x_{ijn} \leq H_n$ for all n; (4) $\sum_p y_{ikp} S(p) = \sum n x_{iS(k^i)n}$ for all i, k
(5) $\sum_p y_{ikp} D(p) = \sum n x_{iD(k^i)n}$ for all i, k; (6) $\sum_p y_{ikp} = 1$ for all i, k
(7) $\tau_{iks_0} = g_{iks_0}$ for all i, k; (8) $\tau_{iks} \geq g_{iks} - g_{iks-1}$ for all i, k, s
(9) $\sum_s g_{iks} = \sum_p b_k^i(p) y_{ikp}$ for all i, k ; (10) $\sum_s \tau_{iks} \leq 1$ for all i, k
(11) $\sum_s g_{iks} \leq S \sum_s \tau_{iks}$ for all i, k; (12) $\sum_{ik} \theta_{iknq} \leq C$ for all n, q
(13) $\sum_q u_{ikq} = \sum_s \tau_{iks}$ for all i, k; (14) $\sum_q v_{ikq} = \sum_s \tau_{iks}$ for all i, k
(15) $z_{ike} \geq x_{iS(k^i)B(e)} + u_{ikT(e)} - 1$ for all i, k, e
(16) $z_{ike} \geq x_{iD(k^i)B(e)} + v_{ikT(e)} - 1$ for all i, k, e; (17) $m_e \geq z_{ike}$ for all i, k, e;
(18) $\phi_{ikls} \geq \sum_p y_{ikp} \eta_{pl} + g_{iks} - 1$ for all i, k, l, s
(19) $\gamma_{ikes} \geq z_{ike} + g_{iks} - 1$ for all i, k, e, s
(20) $\sum_{ik} \phi_{ikls} \leq 1$ for all l, s; (21) $\sum_{ik} \gamma_{ikes} \leq 1$ for all e, s
(22) $\theta_{iknq} \geq \sum_p c_k^i(p) y_{ikp} + C(x_{iS(k^i)n} - 1) + C(u_{ikq} - 1)$ for all i, k, n, q
(23) $\theta_{iknq} \geq \sum_p c_k^i(p) y_{ikp} + C(x_{iD(k^i)n} - 1) + C(v_{ikq} - 1)$ for all i, k, n, q

Constraints (1) and (2) ensure that each vnode is mapped to one of its candidate pnodes. Constraint (3) ensures that each pnode's VM capacity is not exceeded. Constraints (4) and (5) ensure that two end pnodes ($S(p)$ and $D(p)$) of ppath p assigned to vlink k must match the two pnodes assigned to two end vnodes ($S(k^i)$ and $D(k^i)$) of k. Constraint (6) ensures that each vlink is mapped to a ppath. Constraints (7) and (8) ensure spectrum contiguity constraint, where s_0 is the first fslot index. Constraint (9) ensures that the bit rate required by each vlink is satisfied. Note that if source pnode and destination pnode of p are the same pnode, then $b_k^i(p) = 0$, which makes $\sum_s g_{iks} = 0$ since no fslot is allocated. Constraints (10) and (11) ensure that each vlink has one starting fslot if its two end vnodes are not mapped to the same pnode ($\sum_s g_{iks} > 0$), where S is the number of fslots on each plink. Constraint (12) ensures that each SBVT's carrier capacity is not exceeded. Constraints (13) and (14) ensure that each vlink uses at most one SBVT each at source and destination pnodes. Note that a vlink does not need SBVTs if its two end vnodes are mapped to the same pnode. Constraints (15) and (16) ensure that if a vlink uses the corresponding pnode and SBVT with elink e, then the vlink uses e. Constraint (17) ensures that $m_e = 1$ if at least one vlink uses elink e. Constraint (18) ensures $\phi_{ikls} = 1$ if k uses plink l and fslot s. Constraint (19) ensures $\gamma_{ikes} = 1$ if k uses elink

e and fslot s. Constraints (20) and (21) ensure that each fslot on each (extra) physical link can be assigned to at most one vlink. Constraints (22) and (23) ensure that the number of carriers of SBVT q on pnode n allocated to k (i.e., θ_{iknq}) depends on the number of required carriers (i.e., $\sum_p c_k^i(p) y_{ikp}$). If source vnode and destination vnode of k are not mapped to n (i.e., $x_{iS(k^i)n} = 0$ and $x_{iD(k^i)n} = 0$) or SBVT q is not assigned to k (i.e., $u_{ikq} = 0$ and $v_{ikq} = 0$), then θ_{iknq} can be set to 0. If there is a constraint that no two virtual vnodes of the same request can be mapped to a single pnode, then we need another constraint (24) $\sum_j x_{ijn} \leq 1$ for all i, n.

3.2 Lower Bounds

LB1 This bound is based on (1) each vlink's lightpath requires 0 or more carriers of SBVTs at the source and destination pnodes to which the end vnodes are mapped, (2) each SBVT has C carriers, and (3) there are S fslots on each plink.

For each vlink k of request i, according to the distance of each candidate ppath p_k^i in set P_k^i (in turn, the modulation format) and the bit rate requirement Λ_k^i, we can calculate the number of required fslots as $b_k^i(p)$ (including guardband) and the number of carriers $c_k^i(p)$ that need to be allocated to k of i if it is mapped to ppath p. ($b_k^i(p)$ and $c_k^i(p)$ are 0 if p is a dummy ppath with same end pnodes.) For vlink k of i, we can find the minimum number of carriers that have to be allocated as $\hat{c}_{ik} = \min_p c_k^i(p)$. $b_k^i(q)$ fslots will be allocated on every plink l along the ppath q_{ik} that achieves $\hat{c}_{ik}$.

For each plink l, there are several ppaths using fslots on it (according to the above minimum carrier allocation). Each ppath q_{ik} using l requires the minimum fslot $b_k^i(q)$. For each ppath q_{ik}, among all candidate ppaths for k, find the minimum number of required carriers g_{ikl} if plink l is *not* used by k. Let $v_q = g_{ikl} - \hat{c}_{ik}$ (i.e., if k does not use plink l, v_q more carriers have to be allocated). We now formulate a 0/1 knapsack problem (which can be solved easily) to obtain $LB1$. Let the capacity of knapsack be $S + G$ (G is guardband); there are several items with weight $b_k^i(q)$ and value v_q. The objective of 0/1 knapsack is to maximize total value of knapsack with weight constraint. Suppose the result is Z_l; this means that $y_l = \sum_q v_q - Z_l$ more carriers are required to satisfy all requests. Let $Y = \sum_{ik} \hat{c}_{ik} + \sum_l y_l$. Then $LB1 = \lceil Y/C \rceil$. The time complexity to compute $LB1$ is $O(IKV^2M + LIK(S+G))$, where I is number of requests, K is maximum number of vlinks per request, V is number of candidate pnodes per vnode, M is number of ppaths for each pair of pnodes, and L is number of plinks.

LB2 $LB2$ is obtained by observing that (1) vlinks that share an end vnode must be allocated carriers at a single pnode (the pnode to which the end vnode is mapped) and (2) capacity constraint of each pnode (H VMs).

Consider a vnode j which has several candidate pnodes $n_1, n_2, n_3, \ldots$. There are several vlinks $k_1, k_2, k_3, \ldots$ that share j as one end vnode. If j is mapped to n_1, then some carriers of SBVTs at pnode n_1 will be allocated to $k_1, k_2, k_3, \ldots$. We can find the minimum number of required carriers for these vlinks if j is mapped to n_1 based on the candidate ppaths of $k_1, k_2, k_3, \ldots$. Denote $Q_j(n_1)$ as the value. Similarly, $Q_j(n_2), Q_j(n_3), ...$ can be calculated. Therefore, we can find the best pnode mapping for vnode j in terms of the number of required carriers $\bar{c}_j = \min(Q_j(n_1), Q_j(n_2), Q_j(n_3), ...)$.

For each pnode n, there are several vnodes j using VMs on it (according to the above allocation). Each vnode has a VM requirement h_j. For all vlinks with end vnode j, find the minimum number of required carriers g_{jn} for these vlinks if j is *not* mapped to n. Let $v_j = g_{jn} - \bar{c}_j$ (i.e., if j is not mapped to n, v_j more carriers have to be allocated). We again have a 0/1 knapsack problem for n. The capacity of the knapsack is H (the VM capacity), there are several items, with item j having weight h_j and value v_j. The objective is to maximize total value of knapsack with weight constraint. Suppose the result is Z_n, which means $y'_n = \sum_j v_j - Z_n$ more carriers are required to satisfy all requests. Let $Y' = \sum_j \bar{c}_j + \sum_n y'_n$. Then $LB2 = \lceil Y'/C \rceil$. The time complexity to compute $LB2$ is $O(IKV^2M + NIJH)$, where N is the number of pnodes, J is the maximum number of vnodes per request, and H is the number of VMs per pnode.

3.3 Proposed Heuristics

Although ILP can provide an optimal solution, it only works for small instances due to high time complexity. For larger instances, we propose two heuristics based on list scheduling, and then propose two improved meta-heuristics based on Tabu Search. Note that since VNE problem is NP-hard, the heuristics may occasionally fail to produce a result even though a valid mapping exists.

LL Heuristic This heuristic first assigns a weight W_i to each request i based on its requirement, then requests are sorted by decreasing order of weights and mapped one by one (Steps 1 and 4 in Algorithm 1). Regarding a particular i under consideration, a weight ω_{ik} is assigned to each vlink k^i, and the vlinks are sorted by decreasing order of weights and are mapped following this order (Steps 4.a and 4.b). For vlink k^i being mapped, every possible candidate (as a combination of a ppath, a band of contiguous fslots and two SBVTs at source and destination pnodes respectively) is assigned a weight, and the combination with minimum weight is allocated to k^i (Steps 4.b(1)-4.b(3)).

Weight W_i is calculated as $W_i = \max\left(\frac{\hat{h}_i}{\bar{h}\mu}, \frac{\hat{\Lambda}_i}{\bar{\Lambda}\nu}\right)$, where $\hat{h}_i = \sum_j h^i_j$ is the total number of VMs required by i, $\hat{\Lambda}_i = \sum_k \Lambda^i_k$, is the total bit rate requirement of i, $\bar{h} = \sum_i \hat{h}_i$, $\bar{\Lambda} = \sum_i \hat{\Lambda}_i$, $\mu = \sum_{ij} f^i_j$ where f^i_j is the number of pnodes the vnode j can be mapped to, and $\nu = \sum_{ik} r^i_k$ where r^i_k is the number of candidate ppaths the vlink k can be mapped to. Decreasing order of request weights gives higher priority to resource-intensive requests.

Before mapping the requests, we check (in Step 2 *CheckTrivialRequests*) whether any request i can be mapped to a pnode n without affecting vnode mapping of other requests (i.e., all other vnodes can still be mapped to n if n is one of their candidate pnodes). It is best to map an entire request to a single pnode if possible since this mapping won't use any fslots nor SBVTs. There is another check made before mapping VN requests and after every vlink mapping

```
1  Step 1: Assign weight $W_i$ to each VN request $i$; and sort the requests in
     decreasing order of weights
2  Step 2: TrivialRequestCheck()
3  Step 3: CheckOneCandidate()
4  Step 4: Schedule ordered unmapped requests one by one
5  foreach sorted request $i$ do
6      Step 4.a: Assign weight $\omega_{ik}$ to each unmapped vlink $k$ of $i$, and sort them
         in decreasing order of weights
7      Step 4.b: Schedule ordered unmapped vlinks of $i$ one by one
8      foreach unmapped vlink $k$ of request $i$ do
9          foreach allocation-combination $x$ (including candidate ppath $p$, fslot
             band $f$ on $p$, SBVT $t_s(p)$ at source pnode of $p$, SBVT $t_d(p)$ at
             destination pnode of $p$) do
10             Step 4.b(1): Calculate $\mathit{\Delta}_x$ the weight of $x$
11         end
12         Step 4.b(2): Map vlink $k$ to allocation-combination $\bar{x}$ that achieves
             $\min_x \mathit{\Delta}_x$
13         Step 4.b(3): CheckOneCandidate()
14     end
15 end
```

Algorithm 1: LL algorithm

(Steps 3 and 4.b(3)). There is a VM limit at every pnode and a fslot limit on each plink; the purpose of *CheckOneCandidate* is to share these limited resources fairly by first mapping vnodes and vlinks of all requests that have only one pnode and ppath to map to.

For a particular request i under consideration, each of its unmapped vlinks k is assigned a weight ω_{ik} (Step 4.a) as $\bar{b}^i_k/m^k_i$, where m^k_i is the number of possible candidate ppaths for k, and $\bar{b}^i_k$ is the current minimum number of required fslots for k among the m^k_i candidate ppaths. The vlinks of i are sorted in decreasing order of weights to first map vlinks requiring more fslots and having fewer candidate ppaths. Now we describe the vlink mapping steps (Steps 4.b(1)-4.b(3)). For each unmapped vlink k (with source vnode $s(k)$ and destination vnode $d(k)$) mapping, all of its currently available candidate x, as a combination of ppath p (with source pnode $s(p)$ and destination pnode $d(p)$), a band of fslots f on p, SBVT $t_s(p)$ at $s(p)$, and SBVT $t_d(p)$ at $d(p)$, will be assigned a weight $\mathit{\Delta}_x = h'_x + t'_x + b_k(p)/S + c_k(p)/C$, where $h'_x = \max(m_{s(k)}h_{s(k)}/H'_{s(p)}, m_{d(k)}h_{d(k)}/H'_{d(p)})$ if $s(p) \neq d(p)$, otherwise $h'_x = (m_{s(k)}h_{s(k)} + m_{d(k)}h_{d(k)})/H'_{s(p)}$, where $m_{s(k)} = 0$ ($m_{d(k)} = 0$) if $s(k)$ ($d(k)$) has already been mapped, otherwise they are set as 1; $h_{s(k)}$ ($h_{d(k)}$) is the number of required VMs for $s(k)$ ($d(k)$), and $H'_{s(p)}$ ($H'_{d(p)}$) is the number of currently available VMs at $s(p)$ ($d(p)$); t'_x is the number of newly allocated SBVTs (which have not been allocated so far) to k; $b_k(p)$ and $c_k(p)$ are the numbers of required fslots and required carriers, respectively, if k is mapped to p; S is the fslots capacity of each plink; and C is the number of carriers per SBVT. Vlink k is mapped to the candidate achieving the minimum value $\mathit{\Delta} = \min_x \mathit{\Delta}_x$. After that, VM, SBVTs (with carriers), and fslot resources are updated for the corresponding plinks and pnodes (allocated to vlink k and its end vnodes). At last, *CheckOneCandidate* function is called since the mapping of

k might result in changes of other vnodes'/vlinks' available mapping candidates. This procedure is repeated until all vlinks of i are mapped. The time complexity of LL is $O(I^3J^2NK + V^2MSI^3K^3)$.

Note that if no two vnodes of the same request can be mapped to a single pnode, then there is no TrivialRequestCheck() function. Dummy paths (with the same end pnodes) will not be considered as candidate ppaths in all steps.

ALL Heuristic ALL heuristic is similar to LL heuristic with the difference that LL first sorts VN requests, then does the vlink mapping for these requests one by one. In ALL heuristic, all vlinks of all requests are sorted by decreasing order of their weights ω_{ik} (defined in LL heuristic), and the vlinks are then mapped one by one. The time complexity of ALL is the same as LL.

Tabu Search We utilize Tabu Search meta-heuristic to improve the LL and ALL heuristics performance, called LL-TS and ALL-TS, respectively. The main idea of Tabu search is to utilize a neighborhood search procedure to iteratively generate a better solution from one potential solution by exploring the neighborhood of current potential solution, until an attempt limit. In LL-TS algorithm (ALL-TS algorithm), the Tabu search method is utilized to iteratively generate different ordered sets of VN requests (virtual links), each order is treated as a Tabu Search solution. Given this order, the remaining steps in LL (ALL) are kept the same to find the number of required SBVTs. The initial request (vlink) order is the one used in LL (ALL). To create a new ordered set based on current order, two VN requests (vlinks) in current order are randomly selected and swapped. To improve performance, Θ different new ordered sets are created for each current order (according to different swaps). The new order that results in minimum number of required SBVTs will be used as current order for the next iteration. The procedure is repeated for Φ iterations. The larger the Θ and Φ values, the better the performance, but the longer simulation time. The time complexity is $O(\Theta\Phi I^3J^2NK + \Theta\Phi V^2MSI^3K^3)$.

4 Simulation Results

We present results for two network topologies, the small 6-node network, shown in Fig. 1 and Google data center network, shown in Fig. 2.

There are 3 modulation levels used in the simulations: DP-QPSK, DP-8QAM and DP-16QAM, and their corresponding transmission reach limits are 3000 km, 1000 km, and 650 km, respectively [15]. The spectrum efficiencies (in Gbps/GHz) of the three modulation formats are: 4, 6, and 8. The guardband is set as $G = 1$ frequency slot. The bandwidth of each carrier in SBVT is 25 GHz. There are 3 precomputed paths for each pair of physical nodes. Each virtual network request has either 3 or 4 virtual nodes (selected randomly). Each virtual node has randomly 1–3 physical node candidates. The probability that there is a virtual link between each pair of virtual nodes is set as 0.8 for small 6-node network and 0.6 for Google Network. The bit rate requirement for each virtual link is randomly set in the range 100 - 500 Gbps.

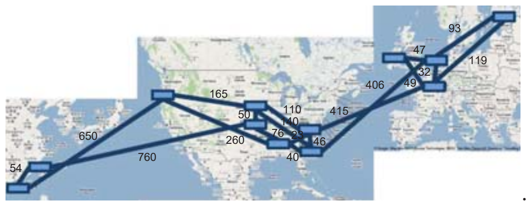

Fig. 2. Google Data Center Network with link distance in km.

4.1 Numerical Results for 6-node Network

In this section, we compare the results of ILP, lower bounds, and four heuristics for small 6-node network topology. We assume that each physical node has 30 VMs and there are 30 frequency slots on each physical link. Each virtual node requires a random number between 1 and 5 VMs. The number of carriers per SBVT is set as $C = 5$. Let I denote the number of VN requests. The value of Θ is set as $I(I-1)/2$ and the value of Φ is set as 500 for Tabu search algorithms. Sample results for 1 - 14 virtual network requests are shown in Table 2 (15 requests turn out to be infeasible due to node capacity), where LB is the minimum of $LB1$ and $LB2$; and *inf* means a feasible result was not found.

It can be seen that ALL algorithm outperforms LL algorithm, but both produce infeasible solutions for some cases. Tabu Search can improve heuristic results, suggesting that the order of requests (or virtual links) can affect performance especially for the cases that resources requirements are close to resources capacity (i.e., 13 and 14 request cases). The LB values are not far away from optimal ILP results.

Table 2. Number of Required SBVTs in 6-node network

I	ILP	LB	LL	ALL	LL-TS	ALL-TS
5	6	4	6	6	6	6
6	8	6	8	8	8	8
7	9	7	11	9	9	9
8	10	8	14	12	11	10
9	10	8	18	14	12	10
10	10	8	20	14	12	12
11	12	10	*inf*	19	15	14
12	14	12	*inf*	21	18	16
13	16	13	*inf*	*inf*	20	20
14	18	14	*inf*	*inf*	22	22

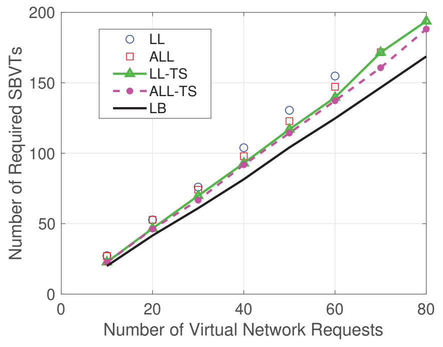

Fig. 3. Number of Required SBVTs vs. Number of Requests.

4.2 Numerical Results for Google Network

For the larger Google network topology, we show the results for lower bounds, LL, ALL, LL-TS, and ALL-TS, as the average of 10 seeds simulation results. We assume that each physical node has 1000 VMs, and there are 320 frequency slots on each link (as optical C-band is 4000 GHz and each frequency slot is 12.5

GHz). The number of required VMs for each virtual node is randomly selected in $\{10, 20, 30, 40, 50\}$. The value of Θ is set as 20 and value of Φ is set as 25. Fig. 3 shows the number of required SBVTs as a function of number of requests when there are 5 carriers per SBVT. Fig. 4 shows number of required SBVTs for 50 requests versus number of carriers per SBVT. Fig. 5 shows simulation results for mixed virtual links' bit rate requirements of 50 requests, where α percentage of virtual links have bit rate requirement in range $100-200$ Gbps, $1-\alpha$ percentage of virtual links have bit rate requirement in range $400-500$ Gbps (when there are 5 carriers per SBVT). These results confirmed that performance of ALL is better than the performance of LL. Tabu Search produces feasible solutions when LL and ALL fail as shown in Fig 3. The differences between Tabu Search and lower bound are on average 10%, 17%, and 11.2% in Figs 3, 4, and 5 respectively.

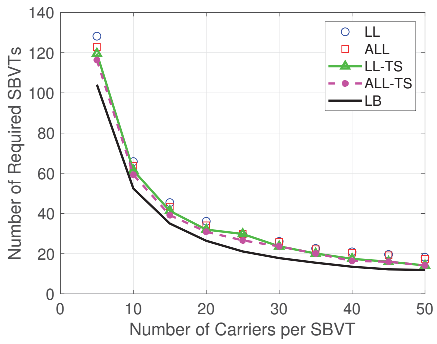

Fig. 4. Number of Required SBVTs vs. Number of Carriers per SBVT.

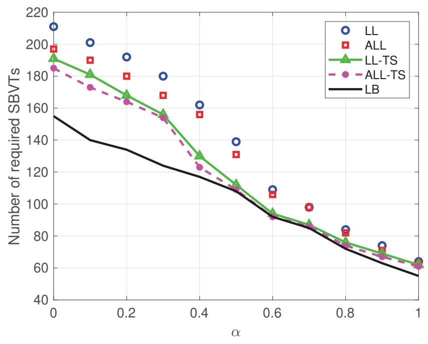

Fig. 5. Number of Required SBVTs vs. parameter α.

5 Conclusions

A virtual network embedding problem is investigated in flexible optical inter-datacenter networks equipped with sliceable transponders. The objective is to minimize the total number of required sliceable transponders. We proposed an Integer Linear Programming formulation, two lower bounds, two heuristics based on list scheduling, and two meta-heuristics based on Tabu Search. Simulation results show that the Tabu Search variant of two proposed heuristics achieve quite good performance compared to the derived lower bounds.

Acknowledgment

This work was supported in part by NSF grants CNS-1813617 and CNS-1813772.

References

1. Tomkos, I., Azodolmolky, S., Sole-Pareta, J., Careglio, D., Palkopoulou, E.: A tutorial on the flexible optical networking paradigm: State of the art, trends, and research challenges. Proceedings of the IEEE **102**(9), 1317–1337 (2014)

2. Jinno, M., Kozicki, B., Takara, H., Watanabe, A., Sone, Y., Tanaka, T., Hirano, A.: Distance-adaptive spectrum resource allocation in spectrum-sliced elastic optical path network. IEEE Communications Magazine **48**(8), 138 – 145 (2010)
3. Chowdhury, N. M. M. K., Rahman, M. R., Boutaba, R.: Virtual network embedding with coordinated node and link mapping. In: 28th Conference on Computer Communications (INFOCOM), pp. 783–791. IEEE, Rio de Janeiro, Brazil (2009)
4. Zhao, J., Subramaniam, S., Brandt-Pearce, M.: Virtual topology mapping in elastic optical networks. In: International Conference on Communications (ICC), pp. 3904–3908. IEEE, Budapest, Hungary (2013)
5. Zhang, S., Shi, L., Vadrevu, C.S.K., Mukherjee, B.: Network virtualization over WDM and flexible-grid optical networks. Optical Switching and Networking **10**(4), 291–300 (2013)
6. Gong, L., Zhu, Z.: Virtual optical network embedding (VONE) over elastic optical networks. Journal of Lightwave Technology **32**(3), 450–460 (2014)
7. Wang, Y., McNulty, Z., Nguyen, H.: Network virtualization in spectrum sliced elastic optical path networks. Journal of Lightwave Technology **35**(10), 1962 – 1970 (2017)
8. Madani, F.M., Mokhtari, S.: Fragmentation-aware load-balancing virtual optical network embedding (VONE) over elastic optical networks. In: The Sixth International Conference on Cloud Computing, GRIDs, and Virtualization, pp. 27–32. Nice, France (2015)
9. Yu, C., Guo, L., Hou, W.: Novel elastic optical network embedding using re-optimized VCAT framework accompanied by hitless PPSM function. Journal of Lightwave Technology **34**(22), 5199 – 5213 (2016)
10. Soto, P., Botero, J.F., Hesselbach, X.: Optimal occupancy mapping of virtual networks over elastic optical infrastructures. In: 19th International Conference on Transparent Optical Networks (ICTON), pp. 1–7. Girona, Spain (2017)
11. Xie, W., Jue, J., Zhang, Q., Wang, X., She, Q., Palacharla, P., Sekiya, M.: Survivable virtual optical network mapping in flexible-grid optical networks. In: International Conference on Computing, Networking and Communications (ICNC), pp. 221–225. Honolulu, HI, USA (2014)
12. Chen, B., Zhang, J., Xie, W., Jue, J., Zhao, Y., Shen, G.: Cost-effective survivable virtual optical network mapping in flexible bandwidth optical networks. Journal of Lightwave Technology **34**(10), 2398–2412 (2016)
13. Zhu, M., Gao, P., Zhang, J., Zeng, X., Zhang, S.: Energy efficient dynamic virtual optical network embedding in sliceable-transponder-equipped EONs. In: IEEE Global Communications Conference, pp. 1–6. Singapore (2017)
14. Dallaglio, M., Giorgetti, A., Sambo, N., Velasco, L., Castoldi, P.: Routing, spectrum, and transponder assignment in elastic optical networks. Journal of Lightwave Technology **33**(22), 4648–4658 (2015)
15. Martinez, R., Casellas, R., Vilalta, R., Munoz, R.: GMPLS/PCE-controlled multi-flow optical transponders in elastic optical networks. Journal of Optical Communications and Networking **7**(11), B71 - B80 (2015)
16. Zhang, J., Ji, Y., Song, M., Zhao, Y., Yu, X., Zhang, J., Mukherjee, B.: Dynamic traffic grooming in sliceable bandwidth-variable transponder-enabled elastic optical networks. Journal of Lightwave Technology **33**(1), 183–191 (2015)
17. Christodoulopoulos, K., Tomkos, I., Varvarigos, E. A.: Elastic bandwidth allocation in flexible OFDM-based optical networks. Journal of Lightwave Technology **29**(9), 1354–1366 (2011)