A Path Generation Approach to Embedding of Virtual Networks

Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Raouf Boutaba

***Abstract*—As the virtualization of networks continues to attract attention from both industry and academia, the Virtual Network Embedding (VNE) problem remains a focus of researchers. This paper proposes a one-shot, unsplittable flow VNE solution based on column generation. We start by formulating the problem as a path-based mathematical program called the primal, for which we derive the corresponding dual problem. We then propose an initial solution which is used, first, by the dual problem and then by the primal problem to obtain a final solution. Unlike most approaches, our focus is not only on embedding accuracy but also on the scalability of the solution. In particular, the one-shot nature of our formulation ensures embedding accuracy, while the use of column generation is aimed at enhancing the computation time to make the approach more scalable. In order to assess the performance of the proposed solution, we compare it against four state of the art approaches as well as the optimal link-based formulation of the one-shot embedding problem. Experiments on a large mix of Virtual Network (VN) requests show that our solution is near optimal (achieving about 95% of the acceptance ratio of the optimal solution), with a clear improvement over existing approaches in terms of VN acceptance ratio and average Substrate Network (SN) resource utilization, and a considerable improvement (92% for a SN of 50 nodes) in time complexity compared to the optimal solution.**

***Index Terms*—Network virtualization, resource allocation, virtual network embedding, column generation, optimization.**

I. Introduction

The ever increasing requirements placed on the Internet are fueling its evolution to architectures which make a better and more efficient use of the available network resources, and promote service innovations. Service Providers (SPs) have to satisfy personalized needs for their customers and hence they are impelled to use different protocol stacks and provide customized services and network resources. Network virtualization [1] has been proposed as a feasible solution to achieve this goal. In network virtualization, Infrastructure Providers (InPs) divide their resources into chunks, called VNs, which are allocated to SPs. Thanks to virtualization, the resource chunks are isolated from each other so the service networks behave as if they were independent though they share the same substrate infrastructure.

However, the creation of VNs on top of a SN is not trivial. VN topologies composed of virtual nodes and virtual links have to be drawn to support traffic flows from sources to sinks. Virtual nodes and virtual links then have to be mapped onto the physical substrate in a way that satisfies user demands and optimizes the use of the available resources. This is the basis of the so called VNE problem [1], which in case of unsplittable flows, i.e. flows that have to be treated as a unit from source to sink, is NP hard [2]. Therefore, to simplify the problem, several existing solutions to VNE either assume that the SN supports the splitting of flows [3], or carry out the node and link embedding in two separate steps [4], which can lead to blocking or rejecting of resource requests at the link mapping stage and hence a sub-optimal substrate resource utilization.

In this paper, we propose a near optimal solution to the unsplittable flow VNE problem obtained by performing the embedding in one-shot (i.e. both virtual nodes and links are embedded in one step) using mathematical programming and path generation[1] [5]. The formulation of the embedding problem as being one-shot is motivated by the need to obtain an efficient embedding solution (which would ultimately lead to better resource utilization and hence better profitability for InPs), while the employment of path generation is aimed at ensuring that the resulting algorithm is more scalable compared to the optimal formulation.

To this end, we formulate two mathematical programs; one is a path-based formulation of the unsplittable flow one-shot VNE problem, also known as the primal problem, while the other is its corresponding dual problem. For given instances of the problem, both the primal and dual problems have approximately the same solution value. The proposed approach begins by obtaining an initial solution (composed of paths in an augmented SN) to the primal problem using a VNE approach that performs node and link mapping in two coordinated stages. The next step is to enhance the initial solution. This is achieved by using the initial solution as an input into the dual problem, hence resulting into *prices* for the SN links and nodes. Using Dijkstra's algorithm [6], these prices are utilized to determine an additional set of paths which can be added to enhance the solution. These paths, together with those obtained in the initial solution, are finally used to solve the primal problem to obtain a final embedding solution.

The main contributions of this paper are as follows:

- A near optimal unsplittable flow one-shot VNE approach that improves substrate resource utilization compared to existing heuristic and approximation solutions.
- A path generation-based approach for unsplittable flows that significantly improves the time complexity of the embedding compared to the optimal solution.

The rest of the paper is organized as follows: Section II

R. Mijumbi, J. Serrat and J.L. Gorricho are with the Network Engineering Department, Universitat Politècnica de Catalunya, 08034 Barcelona, Spain.

R. Boutaba is with the D.R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada.

[1]In this paper, we use the terms *path generation* and *column generation* interchangeably.

presents the description of the VNE problem. We present the related work in Section III. Sections IV and V respectively describe the mathematical formulation of the one-shot embedding problem and its solution based on path generation. Section IV presents the evaluation of our proposed solution and the discussion of the results. Finally, Section VII concludes this paper.

II. Problem Formulation

A. Substrate Network Capacity

We model the SN as an undirected graph denoted by $G_s = (N_s, L_s)$, where N_s and L_s represent the set of substrate nodes and links, respectively. Each substrate link $l_{uv} \in L_s$ connecting the nodes u and v has a bandwidth capacity C_{uv} while each substrate node $u \in N_s$ has computation capacity C_u and a location $Loc_u(x, y)$

B. Virtual Network Requests

In the same way, we model the VN as an undirected graph denoted by $G_v = (N_v, L_v)$, where N_v and L_v represent the set of virtual nodes and links respectively. Each virtual link $l_{ij} \in L_v$ connecting the nodes i and j has a bandwidth demand D_{ij} while each virtual node $i \in N_v$ has computation demand D_i, a location $Loc_i(x, y)$ as well a constraint on its location $Dev_i(\Delta x, \Delta y)$ which specifies the maximum allowed deviation for each of its x and y coordinates[2]. Constraints on the location of virtual nodes are aimed at giving SPs the flexibility to choose the geographical placement of given parts of their network topologies. This could be as a result of a given SP introducing specialized services for users in a given location, or a desire to ensure improved quality of service by restricting the distance (and hence latency) between a given pair of nodes.

C. Virtual Network Embedding

The embedding problem consists in the mapping of each virtual node $i \in N_v$ to one of the possible substrate nodes with in the set $\Upsilon(i)$. $\Upsilon(i)$ is defined as a set of all substrate nodes $u \in N_s$ which have enough *available capacity* (defined the difference between the total capacity of a resource and the amount already allocated) and are *located* within the maximum allowed deviation $Dev_i(\Delta x, \Delta y)$ of the virtual node. For a successful mapping, each virtual node must be mapped and any given substrate node can only map at most one virtual node from the same request. Similarly, all the virtual links have to be mapped to one or more substrate links connecting the nodes to which the virtual nodes at its ends have been mapped. Each of the substrate links must have enough capacity to support the virtual link(s) that go through it. A mapping is successful if all the virtual links are mapped.

In Fig. 1, we show an example of two VNs being mapped onto a SN. The resource requirements for each virtual node or link is also shown. The values in the SN are the total loading of any given physical node or link. As can be noted from Fig.

[2]The notation used in this paper is to represent virtual nodes with the letters i or j and substrate nodes with u or v.

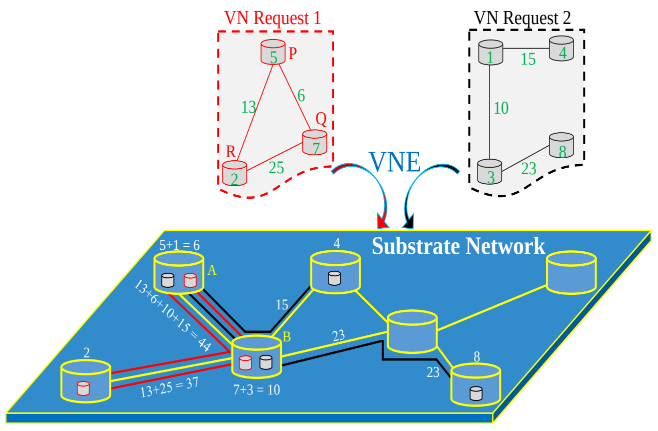


Fig. 1: Virtual Network Embedding: Two VNs mapped onto a SN

1, one substrate node can host more than one virtual node (e.g. node A). A substrate link can also host more than one virtual link (e.g. link AB), and a given virtual link can span more than one substrate link (e.g link RP).

In general, the objective in VNE is to map as many VNs as possible, hence leading to an efficient utilization of SN resources. For the online VNE problem, there is no knowledge on the requirements of future VN requests, and as such, one way of ensuring that as many requests are accepted is by balancing the overall loading of the SN [2] such that all substrate resources (nodes and links) are equally likely to accept resource requests. In the same way, it is worth noting that due to the lack of information about future virtual network requests, optimality as referred to in this paper is only based on the mapping of an arriving virtual network request to a substrate network, which could possibly already have other virtual networks already embedded, or for which other virtual networks may be embedded in the future. Therefore, this optimality does not represent an optimal embedding solution considering all possible virtual network requests.

III. Related Work

VNE is a well-studied problem. In what follows, we only discuss those approaches we consider more closely related to our proposal. An interested reader is referred to [1] for a detailed survey on VNE.

A. Two-step Embedding

Some approaches based on two stages, starting with node mapping and then link mapping, are proposed in [3] and [7]. These algorithms measure the resource of a node or link by its CPU capacity, or bandwidth without considering the topological structure of the VNs and the underlying substrate network. However, the topological attributes of nodes may have an impact on the success and efficiency of VNE. Cheng et al. [8] propose a topology-aware node mapping approach which uses the Markov Random Walk model to rank virtual and substrate network nodes based on their resource and topological attributes. The links are then mapped either using the shortest path (for unsplittable flows), or formulated as a commodity flow problem for splittable flows. Unlike our work, the above approaches don't consider location constraints on virtual nodes, assuming that they can be mapped at any location in the SN. A coordinated node and link mapping is proposed in [2]. Although the coordination here improves the solution space, the mapping is still performed in two separate stages, hence yielding sub-optimal embedding.

The works in [9]–[12] propose dynamic and distributed approaches to VN resource allocation, where the actual resources allocated to virtual nodes and links is scaled up and down based on actual resources utilization as well as resource availability. However, they do not consider the embedding stage, assuming that the VN is already mapped to a SN.

B. *One-shot Embedding*

A one-shot embedding solution based on a multi-agent system is proposed in [13]. However, this proposal assumes unbounded SN resources, and all VN requests to be known in advance. Also, messaging overhead exchanged between the agents can be detrimental to solution scalability. Zhu et al. [4] also propose a one-shot mapping solution, assuming infinite substrate resources, and no constraints on the locations of nodes. Authors in [14] - [15] propose different approaches to one-shot VNE assuming that all VN requests are known in advance (offline solutions), while those in [16] - [17] make simplifying assumptions with regard to the capacity of the SN and do not consider constraints on virtual nodes locations. While most VNE proposals use topologies to represent VN requests, [18] proposes the use of traffic matrices. However, the embedding is achieved by alleviating constraints on VN resources, such as node location. Houidi et al. [19] split any given VN request across multiple infrastructure providers and then uses max-flow and min-cut algorithms and linear programming to find one-shot solutions to the partial VN graphs. While the embeddings of the split graphs are solved in one-shot, they do not encompass the original VN request in its entirety.

Perhaps the the works most related our work are by Jarray et al. [20] and Hu et al. [21] both of whom apply column generation to VNE. Jarray et al. apply a column generation approach to one-shot VNE by assuming that the embedding of VN requests can be delayed by storing each arriving request to process them in batches using an auctioning mechanism. The proposal can therefore be considered to be an offline one. Hu et al. formulate a one-shot path-based VNE where the virtual links are represented as commodities. The formulated mathematical program is then relaxed so as to apply column generation. However, Hu et al. consider a scenario where the demand/commodity of any given virtual link may be split over more than one substrate path. This differs from the proposal in this paper which solves a harder problem where the flows are not splittable.

C. *Mathematical Programming*

Mathematical programming has been applied to a variety of problems in networking. Xie et al. [22] use mathematical programming for dynamic resource allocation in networks while Botero et al. [23] use an optimization technique for link mapping (assuming that the virtual nodes have already been mapped to substrate nodes). Unlike all these works, the mathematical programming formulation proposed in this paper does not only focus on unsplittable flows, but also combines both node and link mapping in one stage. The node mapping step is an important part of VNE since it determines the efficiency of the link mapping. This is why such approaches that coordinate these two steps have been shown to lead to better resource utilization efficiency [2]. Combining these two steps together even further enhances this efficiency, yet the resulting mathematical program is even harder to solve. Finally, path generation based formulations for multi-commodity flow based problems are proposed in a number of approaches such as [24]. In these formulations the source and end nodes for each flow are known a priori, which reduces the complexity of the problem, compared to the one-shot VNE that we solve in this paper.

D. *Summary*

To summarize, because of the NP hardness of the VNE problem, existing one-shot approaches either make simplifying assumptions such as considering the offline version of the problem, assuming infinite resources, or ignoring constraints on the virtual nodes and links, while other proposals solve the embedding problem in two stages, typically employing a greedy approach for node mapping and then try to optimize the link mapping. The approach proposed in this paper differs from previous work in many aspects. Most applications of mathematical programming and path generation to routing are concerned with simpler problems, in which either both the source and sink nodes are known, in which case the problem reduces to a load balancing problem, or only consider node mapping. While the link mapping *sub-problem* is still NP-hard for unsplittable flows, it is even harder in the case of VNE since the source and sink nodes should also be determined. To the best of our knowledge, this is the first path-based mathematical programming solution to the one-shot, unsplittable flow VNE problem.

IV. One-Shot Virtual Network Embedding

The one-shot VNE problem involves performing both node and link mapping at the same time. In this paper, we use mathematical programming to achieve this. Specifically, we consider that VNs arrive one at a time following a Poisson distribution and have exponentially distributed service times, and the formulated optimization problem involves the embedding of a single VN at any given time. This way, at every mapping step, the actual resource availability of all substrate links and nodes is taken into account when performing a mapping. For reference, the link-based formulation of the problem that obtains an optimal solution using mathematical programming is shown in the appendix. Here, we adopt a path-based formulation using column generation in order to solve the problem with much less time and storage requirements.

A. *Substrate Network Augmentation*

We start by creating an augmented network first introduced in [2], with each virtual node i connected to each of the

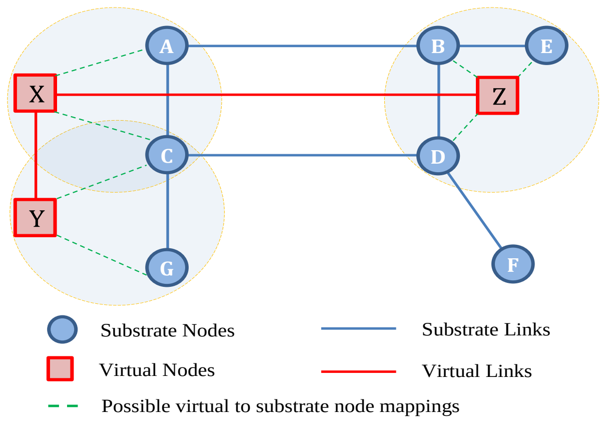


Fig. 2: VNE showing Virtual Node Mapping Constraints

substrate nodes in its possible node set $\Upsilon(i)$ by a *meta link* [2] $l_{iu} \in L_x$, where L_x is the set of all meta links. Then the aim is to establish a single path p_{uv}^{ij} from each virtual node i to all other virtual nodes j to which it is connected. The path p_{uv}^{ij} is made of two meta links, l_{iu} and l_{jv}, and a sub-path in the SN connecting the substrate nodes u and v. This sub-path may be made up of one or more SN links. In Fig. 2, we show a representation of an instance of the problem. In the figure, XYZ are nodes of a VN, while ABCDEFG are nodes of a SN. As an example, for virtual link XZ, one possible path could be XABEZ, and is represented as p_{ae}^{xz}. The path p_{ae}^{xz} is a sequence of links in the augmented network that start from one end of the virtual link to the other. Therefore, in order to embed the virtual link XZ, we need to determine the three components of the path, which – for this example – are the two meta links XA and EZ, and the SN path ABE composed of two links, AB and BE. The components XA and EZ can be determined from a virtual to substrate node mapping, while ABE from a link mapping approach such as shortest path. In particular, this path example would mean that the virtual node X is mapped onto substrate node A, the virtual node Z is mapped onto substrate node E and that the virtual link XZ is mapped onto the SN path ABE. One difficulty illustrated in this example comes from the fact that if, for example, we choose the path XABEZ for virtual link XZ, then the virtual link XY can only be mapped on a path that includes meta link XA and not XC. This would in turn require that Y be mapped onto C, otherwise we would have a sub-optimal solution in which the virtual link XY uses resources from two substrate links (AC & CG) instead of a single link (CG). Hence, the determination of these paths should not be carried sequentially and independently. As previously mentioned, our aim is to find the best possible path for each of the virtual links subject to the mapping requirements described in our problem formulation (see Section II).

B. ***LP–P****: Path based Formulation –Primal*

We formulate the VNE problem as a commodity flow problem [25], where virtual links are flows that should be carried by the SN. However, unlike most commodity flow formulations, in our case, the source node i and terminal node j for each flow also need to be determined.

Variable and Parameter definitions: In this formulation, we define a non-negative binary variable $f_{uv}^{ij} = [0, D_{ij}]$ which represents the unsplittable flow of a virtual link $l_{ij} \in L_v$ on a simple substrate path $p_{uv}^{ij} \in P$. The indices u, v, i and j define a path $(i-u-v-j)$ in the augmented SN. As described in IV (A), these paths are made up of three components: two meta-links iu and jv, and a SN path from u to v. The variable f_{uv}^{ij} is binary in that it can only take on values 0 and D_{ij}, where D_{ij} is the demand of virtual link $l_{ij} \in L_v$. We define P as a set of all the possible substrate paths, P_{uv} as the set of all paths that use the substrate link $l_{uv} \in L_s$ and P^{ij} as the set of all paths that can support the flow for virtual link $l_{ij} \in L_v$. We also define χ_u^i = [0,1] as a binary variable equal to 1 if the virtual node i is mapped onto the substrate node u and 0 otherwise. As mentioned in IV (A), it is important to note that variables χ_u^i and χ_v^j directly determine the existence or otherwise of meta links iu and jv for the path p_{uv}^{ij} since the meta links are dependent on the respective node mappings. For example, if $\chi_u^i == 0$ then the virtual node i is not mapped onto substrate node u, implying that the meta link from i to u is non existent, and so is the path p_{uv}^{ij}. Let A_{uv} be the available bandwidth capacity on the substrate link l_{uv}, and A_u be the available computation capacity on node u.

Objective: The objective of the mathematical formulation (1)–(7) is to balance the resource usage of the SN, by favoring the selection of those resources with comparatively higher available capacity. Balancing the loading of the SN has two advantages; first, it distributes the mapping of a given VN request over multiple SN resources which avoids a single VN being majorly affected by single or regional failures in SN, hence ensuring better VN survivability. In addition, since the problem we consider in this paper is online, we do not know in advance the required node locations for VN requests. Balancing the loading of the SN ensures that at any given point, each substrate node/link has the same capacity on average. This avoids situations where a VN request would be rejected due to one or more of its nodes not being able to be mapped because substrate nodes in their respective possible node sets $\Upsilon(i)$ have less resources than other parts of the SN. As was shown by [2], load balancing leads to a better acceptance ratio of VNs, which would directly translate in higher incomes for InPs.

$$\text{minimize} \sum_{l_{ij}\in L_v} \sum_{p_{uv}^{ij}\in P} \frac{1}{A_{uv}} f_{uv}^{ij} + \sum_{i\in N_v} \sum_{u\in\Upsilon(i)} \frac{1}{A_u}\chi_u^i \tag{1}$$

subject to

$$\sum_{u\in\Upsilon(i)} \chi_u^i = 1 \qquad \forall i \in N_v \tag{2}$$

$$\sum_{i\in N_v} \chi_u^i \le 1 \qquad \forall u \in N_s \tag{3}$$

$$\sum_{p_{uv}^{ij}\in P^{ij}} f_{uv}^{ij} = D_{ij} \qquad \forall l_{ij} \in L_v \tag{4}$$

$$\sum_{p_{uv}^{ij} \in P_{uv}} f_{uv}^{ij} \leq A_{uv} \qquad \forall l_{uv} \in L_s \tag{5}$$

$$f_{uv}^{ij} - D_{ij}\chi_u^i \leq 0 \qquad \forall p_{uv}^{ij} \in P \tag{6}$$

$$f_{uv}^{ij} - D_{ij}\chi_v^j \leq 0 \qquad \forall p_{uv}^{ij} \in P \tag{7}$$

$$f_{uv}^{ij} = [0, D_{ij}] \qquad \forall p_{uv}^{ij} \in P$$

$$\chi_u^i = [0,1] \qquad \forall i \in N_v, \forall u \in N_s$$

The first term in the objective (1) is for link mapping, while the second term is for node mapping. Each of these terms are divided by the respective capacities to ensure that the substrate resources with more free resources are preferred. Constraint (2) ensures that each virtual node is mapped to a substrate node, while (3) ensures that any substrate node may be used at most once for a given mapping request. Constraints (4) and (5) represent the virtual link demand requirements and substrate link capacity constraints respectively. Specifically, (4) states that the flow f_{uv}^{ij} on path p_{uv}^{ij} should carry the total demand of the virtual link ij, while (5) states that the flow f_{uv}^{ij} on path p_{uv}^{ij} should be at most equal to the capacity of each substrate link on that path. From constraint (6), if $\chi_u^i == 0$ then $f_{uv}^{ij} = 0$. If $\chi_u^i == 1$ then $f_{uv}^{ij} = [0, D_{ij}]$. This is also true for (7). These constraints ensure that virtual links and virtual nodes are mapped at the same time, i.e., a flow f_{uv}^{ij} $-$ using the path p_{uv}^{ij} starting with meta link iu and ending with meta link jv $-$ is only non-zero if the virtual node i is mapped onto substrate node u and j is mapped onto v. Together, (6) and (7) ensure that a flow f_{uv}^{ij} is only non zero if both the two end links iu AND jv exist.

The formulation in (1)$-$(7) is intractable for two reasons; first, the restrictions that variables χ_u^i and f_{uv}^{ij} only take on binary values, and then the fact that the number of possible paths p_{uv}^{ij} (and hence the number of variables f_{uv}^{ij}) is very large (exponential) even for moderately sized networks. Therefore, solving the problem in its current form is impractical. There are three possibilities to solving the problem;

1) a relaxation to the constraints on variables χ_u^i and f_{uv}^{ij} to take on continuous values,
2) restricting the number of input variables f_{uv}^{ij} (by restricting the number of paths p_{uv}^{ij}).
3) a combination of both the first two approaches.

For the VNE problem as formulated in (1)$-$(7), a relaxation would require careful consideration to avoid violating the requirements that both nodes and links are mapped in one-shot (since the variables χ_u^i would no longer be able to restrict the mapping of virtual nodes to particular substrate nodes), as well splitting the flows of the virtual links across multiple links. Therefore, we take the second approach, and employ path generation, which allows for the use of only a sufficiently meaningful number of paths, and adding more paths as needed until a final solution is obtained.

V. Path Generation

Path generation is a method that solves mathematical programs with a large number of variables efficiently. The main idea is to solve a restricted version of the program (the restricted primal problem [26]) - which contains only a subset of the variables, and then (through the use of the dual problem [26]) add more variables as needed. Usually, path generation involves creating an *initial solution* (*restricted* set of variables) which are used in the solution for the *restricted* primal problem. Then, solving pricing problems (which are determined from the dual problem), allows for adding more variables to improve the initial solution, until either a final optimal solution is found, or a stopping condition is reached.

The path generation approach taken in this paper is as follows: we start by creating an initial set of paths using a two stage node and link mapping. We then use these paths to solve a dual problem, and use the pricing problems to determine a set of paths to add to the initial solution. These paths are then used to solve a restricted primal problem to obtain a final solution. Therefore, our proposal avoids the usual iteration required in a path generation approach where the primal and dual problems are solved sequentially, many times, instead preferring only to perform a single iteration. In the next subsections, we propose a method for determining the initial set of paths, derive the pricing problems, and then describe the overall algorithm proposed in this paper.

A. Initial Solution

An initial solution (Init$-$Sol) is determined as a set of paths P' in the augmented SN, with each path $p_{uv}^{ij} \in P'$ able to support the flow f_{uv}^{ij} of virtual link l_{ij}. Each of these paths must be able to meet the VN mapping conditions as formulated in the primal problem. Considering the example in Fig. 2, since we have two virtual links, an initial solution would have two paths, one for each virtual link. Examples of these paths could be XABEZ and XACY for virtual links XZ and XY respectively. In order to determine such a path, say for virtual link XZ, the approach in this paper is as follows: we start by performing a node mapping, which for this example, would map virtual nodes X and Z onto substrate nodes A and E respectively. This step gives us the meta links XA and EZ. In this subsection, we propose a novel node mapping solution LP$-$N for determining XA and EZ. The next step involves determining the path ABE in the SN. This is done by using Dijkstra's algorithm, with the constraint that each link on the path should have enough capacity to support the virtual link under consideration. The complete path is determined by joining meta links XA and EZ to the respective ends of ABE.

LP$-$N: Node Mapping: LP$-$N is based on mathematical programming. It is formulated in such a way that mapping of any given virtual node is relatively biased towards each substrate node by a weight. The determination and use of the weights is discussed in what follows.

Objective: It is noteworthy that, essentially, LP$-$N is

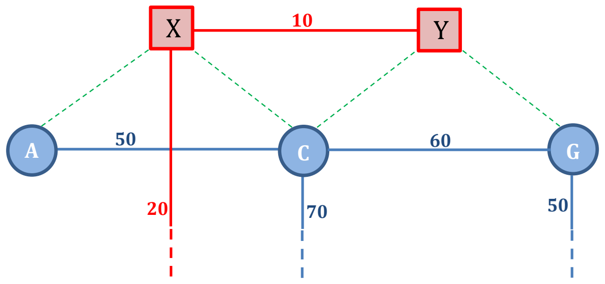


Fig. 3: Node-Link Weighted Averages

aimed at achieving an initial node and link mapping. As such, many other state-of-art two-step approaches [1] could be used for this purpose. However, the authors could not find a previous two-stage mapping approach that simultaneously achieves both objectives considered in our formulation: The first objective is to keep the computation time of the initial solution as low as possible by including only the possible virtual node to substrate node combinations. Secondly, as explained later in this section, we minimize the possibility of failure at the link mapping stage, by making the node mapping *aware* of the link mapping stage through the use of weights W_i and W_u.

Variable definition: As before, χ_u^i is a binary variable equal to 1 when the virtual node i is mapped onto substrate node u and 0 otherwise.

$$\text{minimize} \sum_{i \in N_v} \sum_{u \in \Upsilon(i)} \frac{W_i}{W_u} \chi_u^i \tag{8}$$

subject to:

$$\sum_{u \in \Upsilon(i)} \chi_u^i = 1 \qquad \forall i \in N_v \tag{9}$$

$$\sum_{i \in N_v} \chi_u^i \leq 1 \qquad \forall u \in N_s \tag{10}$$

$$\chi_u^i = [0, 1] \qquad \forall i \in N_v, \forall u \in N_s$$

Constraints (9) and (10) are the same as (2) and (3). The weights W_i and W_u are dynamically determined for each virtual and substrate node respectively. The motivation to use such weights is from the need bias or coordinate the mapping of the nodes to the following link mapping step. This has been shown by related works to improve the mapping efficiency [2] by avoiding the use of a high amount of resources for the link mapping phase. In our proposal, this is particularly important to avoid the possibility that we fail to obtain an initial solution due to unavailable SN resources. Therefore, W_u is defined as the weighted average of the available capacities of all the substrate links connected to u. Similarly, W_i is defined as the weighted average of the demand of all the virtual links connected to i. To illustrate the idea behind these weighted averages, consider Fig. 3, which is a subset of the topology represented in Fig. 2. The values beside each link represent the available link bandwidths and link demands respectively. As an example, considering the virtual node X,

$$W_X = 20 \times \left(\frac{20}{20 + 10}\right) + 10 \times \left(\frac{10}{20 + 10}\right) = 16.67.$$

In the same way for substrate node C,

$$W_C = 50 \times \left(\frac{50}{50 + 60 + 70}\right) + 60 \times \left(\frac{60}{50 + 60 + 70}\right) + 70 \times \left(\frac{70}{50 + 60 + 70}\right) = 61.11$$

The reason for using this ratio as a weight is to ensure that those substrate nodes that are connected to many substrate links with higher available resources are usually preferred, and that in case two or more virtual nodes have a given substrate node in their possible node set (such as X and Y in Fig. 2), then the substrate node would always be allocated to that virtual node with the highest weighted average link demand. This achieves some level of coordination between the node mapping and link mapping phases and thereby reduces the probability of rejecting link mapping requests.

We note that there could be instances where the weighted averages lead to selecting substrate nodes with less good links, especially when the links have widely differing residual capacities. For example, a node connected to two links with residual capacities 80 and 10 respectively will have a $W_1 = 72$, while a node connected to two links with residual capacities 60 and 70 respectively will have a $W_2 = 65$. In this case, the first node will be selected yet the second node *could* be a better choice. One simple solution to handle such scenario is to use the sum of two averages: the weighted average and a simple average. However, it is worth mentioning that in our approach network embedding is done in such a way that the average loads of SN nodes and links are balanced, this way, avoiding scenarios where some node and/or links have widely differing residual capacity. The procedure, **Init-Sol**, for determining the initial solution is shown in Algorithm 1.

B. Pricing Problem

To determine which paths should be added to the initial set so as to improve the solution, we need to solve the pricing problems for LP$-$P. In order to identify the pricing problems we first formulate the dual problem LP$-$D for the primal problem LP$-$P. The formulation of a dual problem from a primal can be obtained in five steps [27].

1) Creating a dual variable for every primal constraint,
2) Creating a dual constraint for every primal variable,
3) The right-hand sides of primal constraints become coefficients for the dual objective,
4) The coefficients of the primal become right-hand sides of the dual constraints,
5) If the primal problem is a maximisation problem, the dual is a minimisation problem.

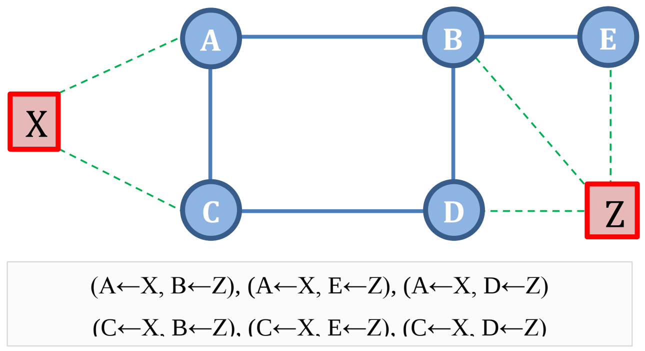
In Table I, we summarize these steps, giving the bounds for the resulting dual constraints and variables for all possible cases of primal variables and constraints respectively. These

Algorithm 1 Init−Sol $(G_v(N_v, L_v), G_s(N_s, L_s))$

```
1: for $i \in N_v$ do
2:     $Determine\ Candidate\ Node\ Set,\ \Upsilon(i)$
3:     if $\Upsilon(i) = \emptyset$ then
4:         $Reject\ Request$
5:         end
6:     end if
7:     $Calculate\ \ W_i$
8: end for
9: for $u \in N_s$ do
10:    $Calculate\ \ W_u$
11: end for
12: $Solve$ : LP-N
13: for $l_{ij} \in L_v$ do
14:     for $u \in \Upsilon(i)$ do
15:         if $\chi_u^i = 1$ then
16:             Meta Link 1: $l_1 = iu$
17:             Start Node, $s = u$
18:         end if
19:     end for
20:     for $v \in \Upsilon(j)$ do
21:         if $\chi_v^j = 1$ then
22:             Meta Link 2: $l_2 = jv$
23:             End Node, $t = v$
24:         end if
25:     end for
26:     LinkMapping: $p_s = Dijkstra\Big(s, t, G_s(N_s, L_s)\Big)$
27:     Create Path: $p_{uv}^{ij} = l_1 + p_s + l_2$
28:     Add $p_{uv}^{ij}$ to $P'$
29: end for
```

TABLE I: Relationship between dual and primal problems

Primal	Dual
Objective Function	
Maximisation	Minimisation
Variable bounds	**Constraint bounds**
$-\infty \leq i \geq +\infty$	$i =$
$i \geq 0$	$i \geq$
$i \leq 0$	$i \leq$
Constraint bounds	**Variable bounds**
$j =$	$-\infty \leq j \geq +\infty$
$j \geq$	$j \leq 0$
$j \leq$	$j \geq 0$



(A←X, B←Z), (A←X, E←Z), (A←X, D←Z)

(C←X, B←Z), (C←X, E←Z), (C←X, D←Z)

We use A←X to mean that node virtual node X is mapped onto substrate node A

Fig. 4: Possible substrate node combinations for virtual link XZ

conventions reflect the interpretation of the dual variables as shadow prices of the primal problem. A less-than-or-equal-to constraint, normally representing a scarce resource, has a positive shadow price, since the expansion of that resource generates additional profits. On the other hand, a greater-than-or-equal-to constraint usually represents an external requirement (e.g., demand for a given resource). If that requirement increases, the problem becomes more constrained; this produces a decrease in the objective function and thus the corresponding constraint has a negative shadow price. Finally, changes in the right hand side of an equality constraint might produce either negative or positive changes in the value of the objective function. This explains the unrestricted nature of the corresponding dual variable.

Dual Variables definitions: To determine the dual program, we start by relaxing the bounds of the variables χ_u^i and f_{uv}^{ij} such that $\chi_u^i \geq 0$ and $f_{uv}^{ij} \geq 0$. Then, following the five steps stated above, we define six dual variables as follows: λ_i for the virtual node constraints (2), μ_{ij} for the virtual links demand constraints in (4), $\eta_u >= 0$ substrate node constraints in (3), $\gamma_{uv} >= 0$ substrate links available capacity constraints in (5), $\sigma_{iu} >= 0$ for simultaneous node and link mapping constraint constraint (6) and $\tau_{jv} >= 0$ for constraint (7). Since most results of duality for linear programs do extend to integer programming [28], the dual formulation in this paper is based on [27].

The **objective** of the dual formulation (11)−(13) is to obtain a mathematical program that produces a maximized value as close as possible to that of its original primal program for any instance of the variables. Therefore, the dual of the primal formulation in (1)−(7) is:

$$\text{maximize} \sum_{i \in N_v} \lambda_i + \sum_{l_{ij} \in L_v} D_{ij}\mu_{ij} - \sum_{u \in N_s} \eta_u - \sum_{l_{uv} \in L_s} A_{uv}\gamma_{uv} \tag{11}$$

subject to

$$\lambda_i + \sum_{p_{uv}^{ij} \in P} (\sigma_{iu} + \tau_{jv}) - \eta_u \leq \frac{1}{A_u} \qquad \forall l_{iu} \in L_x \tag{12}$$

$$\mu_{ij} - \sigma_{iu} - \sum_{l_{uv} \in p_{uv}^{ij}} \gamma_{uv} - \tau_{jv} \leq \sum_{l_{uv}, l_{iu} \in p_{uv}^{ij}} \frac{1}{A_{uv}} \qquad \forall p_{uv}^{ij} \in P \tag{13}$$

The pricing problems are shown in (12) and (13). From (12), the pricing condition for substrate nodes can be determined as:

$$\lambda_i + \sum_{p_{uv}^{ij} \in P} (\sigma_{iu} + \tau_{jv}) > \frac{1}{A_u} + \eta_u$$

However, since the variables χ_u^i are much fewer compared to f_{uv}^{ij}, we include all the possible substrate nodes for each virtual node in the restricted primal problem. This eliminates the need for node pricing and we are left to deal with only the link pricing problem (13):

$$\mu_{ij} > \sum_{l_{uv}, l_{iu} \in p_{uv}^{ij}} \frac{1}{A_{uv}} + (\sigma_{iu} + \sum_{l_{uv} \in p_{uv}^{ij}} \gamma_{uv} + \tau_{jv})$$

This pricing problem can be solved using the shortest path algorithm. Any path $p_{uv}^{ij} = S_{iu} + (l_{uv} \in P_{uv}) + T_{jv}$ in the augmented SN whose length with respect to the dual variables (this means that the costs of the substrate links $l_{uv} \in P_{uv}$ are γ_{uv}, those of meta links S_{iu} are σ_{iu} and those of T_{jv} are τ_{jv}) is smaller than μ_{ij} satisfies the inequality above, and *has the potential* to improve the solution. However, a change in path for any given virtual link could necessitate a change in the mapping of one of its end nodes, which would change the prices and feasibility of mappings for other virtual links connected to it. For example in Fig. 2, if the virtual node X is mapped onto substrate node C, all the paths for both links XZ and XY go through C. If the path for say XZ is changed to go through A, it would either mean that the path for XY should also be changed to go through A, otherwise this path cannot be used to give a feasible and improved solution. Therefore, addition of paths individually for each virtual link does not guarantee that each of the added paths would still lead to a feasible solution, and for as long as the added path cannot yield a feasible solution, this path cannot lead to improvement in the solution of the restricted primal problem. In this case, there would be no guarantee that the pricing problems can be solved in polynomial time, as it could require quite a number of iterations before enough paths are added to actually improve the solution.

In this paper, instead of adding individual paths for each virtual link in each iteration of the path generation algorithm, we include all the possible shortest path combinations after solving the formulation (11) $-$ (13). We use Fig. 4, which is extracted from Fig. 2, to illustrate this for the case of virtual link XZ. Since the node X has two possible substrate nodes and virtual node Z has three possible substrate nodes, then the possible combinations for these nodes are 6. In our pricing solution, we determine the shortest path $-$ based on the weights in (14) for each of these 6 possible end node combinations.

$$\sum_{l_{uv}, l_{iu} \in p_{uv}^{ij}} \frac{1}{A_{uv}} + \left(\sigma_{iu} + \sum_{l_{uv} \in p_{uv}^{ij}} \gamma_{uv} + \tau_{jv}\right) \quad (14)$$

This is done for all the virtual links, and all the corresponding paths are added to the restricted primal problem. However, the number of paths added for each pricing iteration would be too big to handle if many iterations are carried out. Even the Dijkstra algorithm takes quite some time to find the shortest paths. For this reason, we perform only one round for the substrate paths and use the resulting shortest paths based on the dual problem to solve LP$-$P to obtain the final solution. As we show in the simulation results, the solution obtained is near optimal.

The proposed approach, **Final$-$Sol**, for determining the final solution is shown in Algorithm 2.

Algorithm 2 Final$-$Sol$(G_v(N_v, L_v), G_s(N_s, L_s))$

```
1:  Create Augmented Substrate Network
2:  Initial Paths Set: $P' \leftarrow Solve$ **Init−Sol**
3:  $Solve$ **LP−D($P'$)**
4:  **for** $l_{ij} \in L_v$ **do**
5:      **for** $u \in \Upsilon(i)$ **do**
6:          **for** $v \in \Upsilon(j)$ **do**
7:              $P' \leftarrow (P' + GetShortestPath(i, u, v, j))$
8:          **end for**
9:      **end for**
10: **end for**
11: $Solve$ **LP−P($P'$)**
```

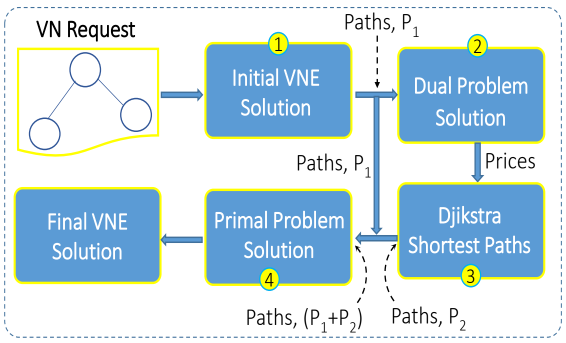
Example: To illustrate the details of **Final$-$Sol** in algorithm 2, we use a simple running example based on Fig. 2 as well as the flow diagram in Fig. 5a. The aim of the example is to illustrate the sequence of the proposed algorithm rather than its effectiveness, which is evaluated in the next section. As such, we keep it simple by avoiding the details of how the actual mathematical programs are solved. In Fig. 5b, we show a possible initial solution (black dotted lines) where virtual nodes X, Y and Z have been mapped to substrate nodes A, G and E respectively. The virtual links XY and XZ have been mapped onto substrate paths ACG and ABE respectively. Therefore, based on the discussion in section IV(A), the initial solution is made up of two paths XACGY and XABEZ in the augmented substrate network. In Fig. 5a, these two paths make up P_1. With these paths, the dual problem (LP-D(P_1)) is solved. The values of σ_{iu}, γ_{uv} and τ_{jv} along each link in Fig. 5b represent hypothetical values that could result from solving LP-D. As explained above, and illustrated in Fig. 4, the next step is then, for each virtual link, to find the shortest path in the substrate network for all the possible virtual-to-substrate node mappings. Using the values in Fig. 5b, the these shortest paths are determined using Dijkstra's algorithm as shown in Table II[3] for the 6 combinations (see Fig. 4) of the virtual link XZ. Using a similar process, the paths corresponding to the virtual link XY are determine. These paths (excluding those which were already in the initial solution such as XABEZ) constitute P_2 in Fig. 5a. The combined paths $(P_1 + P_2)$ are then used as inputs to solve a restricted primal problem to obtain a final solution.

VI. Performance Evaluation

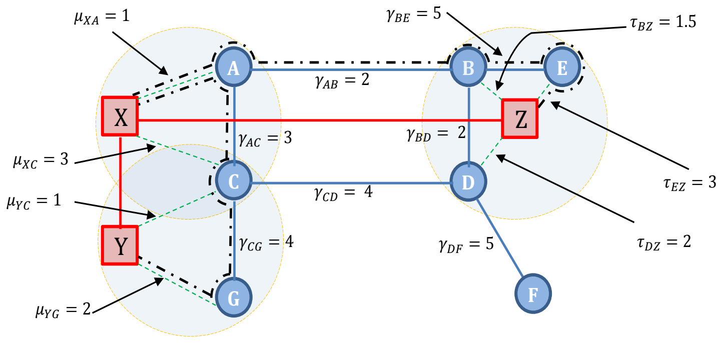
A. Simulation Setup

To evaluate the performance of our proposed approach, we implemented a discrete event simulator in Java, which uses the tool Brite [29] to generate substrate and VN topologies. We used the tool ILOG CPLEX 12.4 [30] to solve the mathematical programs. Simulations were run on Windows 8

[3]The reader should note that for representation simplicity, the terms $1/A_{uv}$ in (14) are not included in the shortest path summations in Table II. These terms represent the reciprocal of the available bandwidth on each link along the shortest path in the augmented substrate network.



(a) Summary of Path generation-based VNE Approach



(b) Initial Solution and Dual Pricing of Links

Fig. 5: Running Example

TABLE II: Shortest Paths for Virtual Link XZ

Path	Values Along Path	Total Path Length
XABEZ	$1+2+5+3$	11
XABZ	$1+2+1.5$	4.5
XABDZ	$1+2+2+2$	7
XCZBEZ	$3+3+2+5+3$	16
XCABZ	$3+3+2+1.5$	9.5
XCDZ	$3+4+2$	9

TABLE III: Brite Network Topology Generation Parameters

Parameter	Substrate Network	Virtual Network
Name (Model)	Router Waxman	Router Waxman
Number of nodes (N)	100 and 20	[15-25] and [3-10]
Size of main plane (HS)	500	500
Size of inner plane (LS)	500	500
Node Placement	Random	Random
GrowthType	Incremental	Incremental
Neighbouring Nodes	3	2
alpha (Waxman Parameter)	0.15	0.15
beta (Waxman Parameter)	0.2	0.2
BWDist	Uniform	Uniform

TABLE IV: Performance Quality Evaluation Algorithms

Code	Mapping Method
GNMSP	Greedy Node Mapping and Shortest Path (SP) for Links [3]
CNMMCF	Coordinated Node and MCF for Link Mapping [2]
VNA-1	One-Shot Mapping [4]
TANMSP	Topology-aware Node Mapping and SP for Links [8]
PaGeViNE	Path Generation based Virtual Network Embedding
ViNEOPT	Link based Optimal Virtual Network Embedding

Pro running on a 4.00GB RAM, 3.00GHz Processor Machine. Both substrate and VNs were generated on a 500×500 grid. The CPU and bandwidth capacities of substrate nodes and links are uniformly distributed between 50 and 100 units respectively. The CPU demand for VN nodes is uniformly distributed between 2 and 10 units while the bandwidth demand of the links is uniformly distributed between 10 and 20 units. The parameters used in Brite to generate network topologies are shown in Table III. The parameters α and β are Waxman-specific exponents, such that, $0 < \alpha \leq 1, 0 < \beta \leq 1, (\alpha, \beta) \in \mathbb{R}$. α represents the maximal link probability while β is used to control the length of the edges. High values of alpha lead to graphs with higher edge densities while high values of beta lead to a higher ratio of long edges to short ones. The values used in this paper are the default values in the Brite router Waxman model used in [29]. Each virtual node is allowed to be located within a uniformly distributed distance between 100 and 150 units of its requested location. For embedding quality evaluations, two possible sets of network sizes have been used. One involves a SN with 100 nodes and VNs with number of nodes varied uniformly between 15 and 25, while the other has a SN with 20 nodes and VNs with number of nodes varied uniformly between 3 and 10. The need for different network sizes will be explained in a later subsection. For these simulations, we assumed Poisson arrivals at an average rate of 1 per 3 time units. The average service time of the requests is 60 time units and assumed to follow a negative exponential distribution. The experiments are performed for 1500 arrivals. For the time complexity evaluation, the number of nodes for the SN is gradually increased from 20 to 100, and each simulation setup is repeated 20 times and average values determined.

B. *Performance Metrics*

1) Solution Quality: Three performance indicators – Acceptance ratio, Node utilization and Link utilization – are used for quality evaluation. The acceptance ratio gives a measure of the number of VN requests accepted compared to the total requests. We define the average node utilization as the average proportion of the total substrate node capacity that is under use at any given time. In the same way, we define average link utilization as the average proportion of the total substrate link capacity that is under use at any given time.

2) Solution Complexity: We define the time complexity of a given solution as the average time to complete the computation.

3) Embedding Cost and Revenue: We define the costs and revenue from embedding a given VN the same way as a related

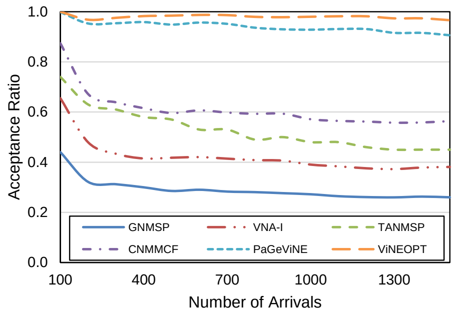


Fig. 6: Average Acceptance Ratio - 20 SN Nodes

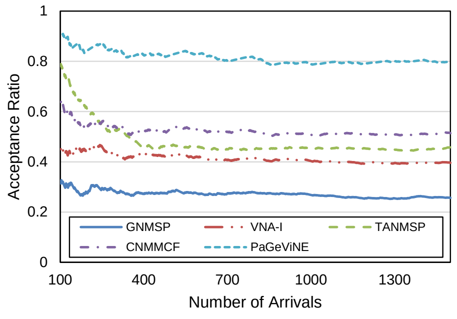


Fig. 7: Average Acceptance Ratio - 100 SN Nodes

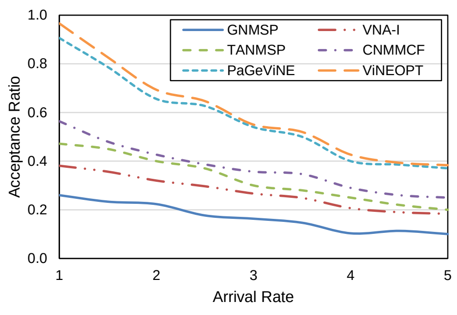


Fig. 8: Effect of VN Arrival Rate on Acceptance Ratio

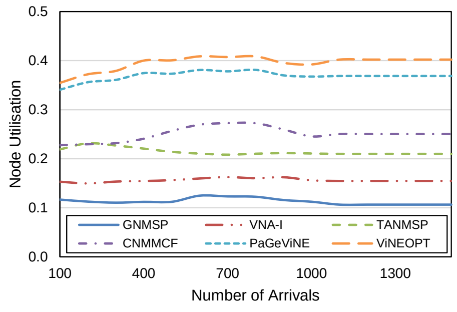


Fig. 9: Average Node Utilisation

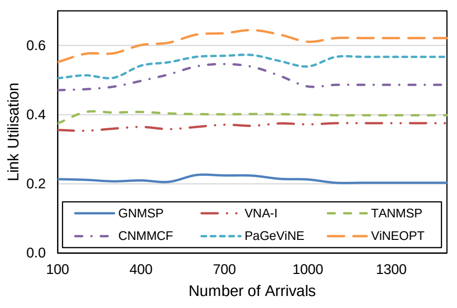


Fig. 10: Average Link Utilisation

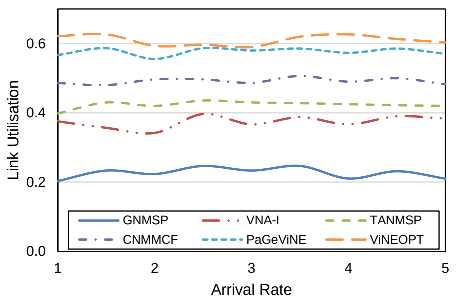


Fig. 11: Effect of VN Arrival Rate on Link Utilization

work [2]. In particular, we define revenue, $R\Big(G_v(N_v,L_v)\Big)$ as the benefit to the SN for accepting the VN request $G_v(N_v,L_v)$. As formulated in (15), it is the weighted sum of the link and node demands for the VN.

$$R\Big(G_v(N_v,L_v)\Big) = \sum_{i\in N_v} D_i + \sum_{l_{uv}\in L_v} D_{ij} \tag{15}$$

Similarly, in (16), we define an embedding cost $C\Big(G_v(N_v,L_v)\Big)$ as the sum of total substrate resources that are allocated to the VN $G_v(N_v,L_v)$. κ_u and ξ_{uv} are parameters that represent the relative unit costs of substrate nodes and links respectively, where the virtual nodes and links are mapped.

$$C\Big(G_v(N_v,L_v)\Big) = \sum_{i\in N_v} \kappa_u D_i + \sum_{l_{ij}\in L_v}\sum_{l_{uv}\in L_s} \xi_{uv} f_{uv}^{ij} \tag{16}$$

C. Comparisons

We compare the performance of our solution with closely related solutions. In particular, four representative solutions from the literature are chosen. We name and describe the compared solutions in table IV. These solutions were slightly modified to fit into our formulation of the problem. Specifically, unsplittable flows, constraints on SN capacities and constraints on virtual node locations were applied. We also implemented a baseline formulation of the optimal one-shot mapping (see Appendix).

Since ViNEOPT requires a very long time (in excess of 1 hour for a single embedding involving a SN of 60 nodes and a VN of 10 nodes) to perform an embedding, simulations evaluating this algorithm have been restricted to SNs with 20 nodes and VNs with nodes from 3 – 10. However, an extra simulation for acceptance ratio using larger sized networks has been performed so as to reflect more practical network sizes. This simulation excludes ViNEOPT.

D. Results

1) Solution Quality: From the graphs in Fig. 6 it is evident that PaGeViNE achieves an average acceptance ratio close to that obtained by the optimal solution ViNEOPT. In addition Fig. 6 and Fig. 7 show that PaGeViNE outperforms state-of-the-art solutions in terms of average acceptance ratio. These two figures also confirm that the embedding efficiency of PaGeViNE is not affected by increasing the size of substrate and VNs. In addition, it can be observed from Fig. 8 that even as the arrival rate of VNs increases, PaGeViNE continues to perform comparable to ViNEOPT and better than the four other approaches. The fact that CNMMCF is under-performing PaGeViNE with respect to the average acceptance ratio and resource utilization can be attributed to the fact that CNMMCF is using more resources at the link mapping stage since it performs node and link mappings separately. For VNA-1, while the node and link mapping is done in one shot, they are carried out sequentially, considering specific clusters of the SN each time. It is therefore expected that the results would not be as good as those achieved by a global solution based on mathematical programming. It can also be observed that TANMSP, which uses the topology information to determine node mapping performs better than VNA-1 and GNMSP. However, since it still falls short of CNMMCF which determines the node mapping from a mathematical program.

It is also evident from the graphs in figures 9 and 10 that PaGeViNE achieves a better utilization ratio for substrate node and link resources compared to other solutions. However,

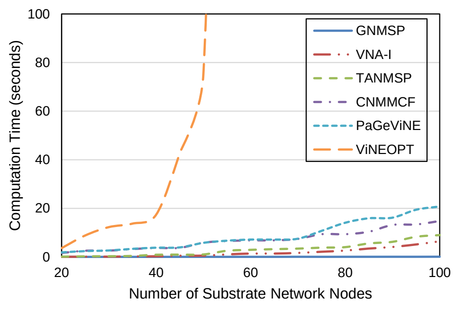


Fig. 12: Average Computation Time

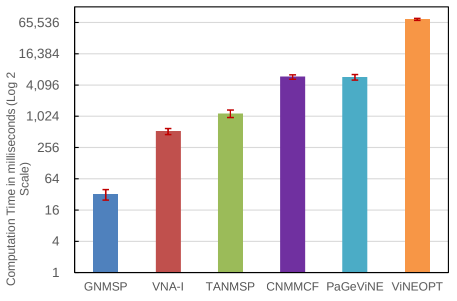


Fig. 13: 95% Confidence Interval Error Bars

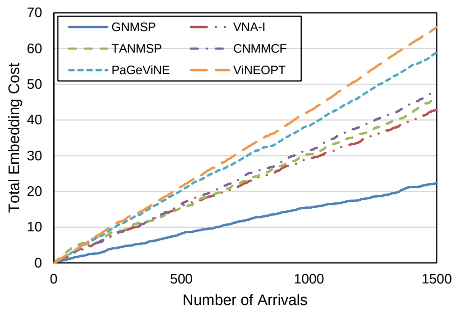


Fig. 14: Cummulative Embedding Cost

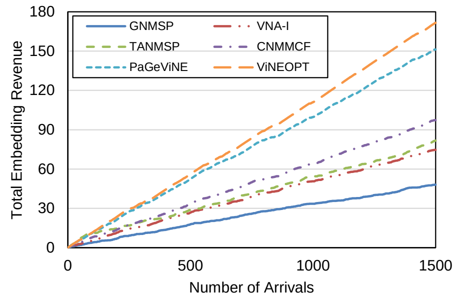


Fig. 15: Cummulative Embedding Revenue

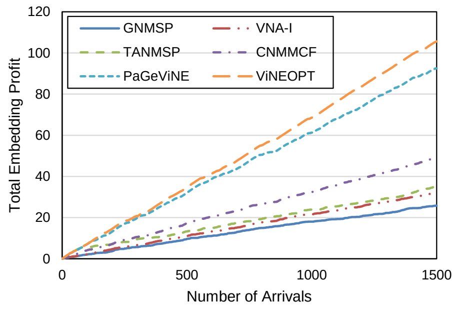


Fig. 16: Cummulative Embedding Profit

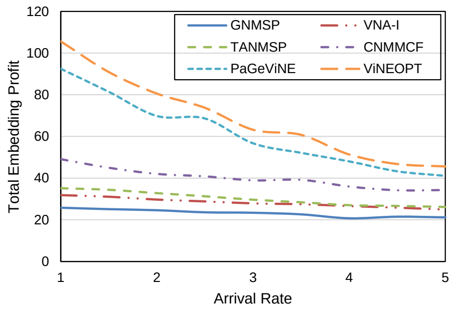


Fig. 17: Effect Arrival Rate on Profit

we note that CNMMCF has a link utilization ratio that is comparatively close to that of PaGeViNE. Finally, it is evident from Fig. 11 that the utilisation of the resources is almost un affected by the arrival rate. This confirms the fact that the rejection of VN requests is not caused by depletion of resources but rather by inefficient embedding which either fails due to bottleneck nodes. This is why mathematical programming-based algorithms which have global knowledge of the embedding perform better.

2) Solution Complexity: With respect to time complexity, the graphs in Fig. 12 show that the running times of GNMSP, VNA-1 and TANMSP are comparatively lower than those of PaGeViNE. Once again, this can be explained by the fact that these two solutions do not solve a mathematical program as PaGeViNE does. We also note that the computation time of PaGeViNE is slightly higher than that of CNMMCF. This can be attributed to the fact that PaGeViNE solves three mathematical programs, while CNMMCF solves only two. Moreover, it is expected that solving the problem in one-shot requires more computation than solving it in two stages, since some of the mathematical programs solved in PaGeViNE are binary. With regard to ViNEOPT we see that the computation time quickly grows exponentially. In fact, ViNEOPT could not find a solution even after 1 hour for 60 substrate nodes[4]. We therefore note a significant improvement in time complexity of PaGeViNE compared to ViNEOPT. These simulations were each repeated 20 times, and the average time determined in each case. In Fig. 13, we show the 95% confidence intervals of the computation time for a SN with 50 nodes. The small error values in each of these graphs further confirms the profile in Fig. 12.

[4]Once again, this is why the simulations for acceptance ratio were split into one with 20 SN nodes and another with 100 substrate nodes.

3) Embedding Cost, Revenue and Profit: Figs. 14, 15 and 16 show the cumulative embedding costs, revenue and profit. The profit is the difference between the revenue and cost of embedding a given VN. We note that PaGeViNE achieves a profitability close that of ViNEOPT, which is considerably higher than that of the compared state-of-art approaches. We also note CNMMCF achieves a higher profitability than VNA-1, TANMSP and GNMSP. It is worth noting that these profiles are similar to those obtained from the acceptance ratios of the three approaches. This means that the superiority of our approach is not based on accepting VNs with less resources requirements which would be less profitable for the physical resource providers. The fact that VNA-1, TANMSP, GNMSP and CNMCMF obtained much lower embedding costs is due to rejecting most of the VN requests, which is further confirmed by the revenue obtained, and hence profitability. In Fig. 17, we evaluated the effect of the arrival rate on profitability, noting that as the arrival rate is increased, the profitability reduces. This is not surprising since an increase in the arrival rate ensures that most of the arriving VN requests in the simulation time are not accepted due to lack of resources. This profile is consistent with that of the acceptance ratio is Fig. 8.

Effect of Initial Solution: In Fig. 18, we evaluate the proposed initial solution. In particular, the effect of the initial solution on the computation time for a substrate network of 20 nodes, and the acceptance ratio after the arrival of 1,500 VN requests are shown. It can be observed that the proposed initial solution achieves the balance between time complexity and embedding quality. While it performs worse than GNMSP, VNA-1 and TANMSP in terms of computation time, it outperforms them on solution quality.

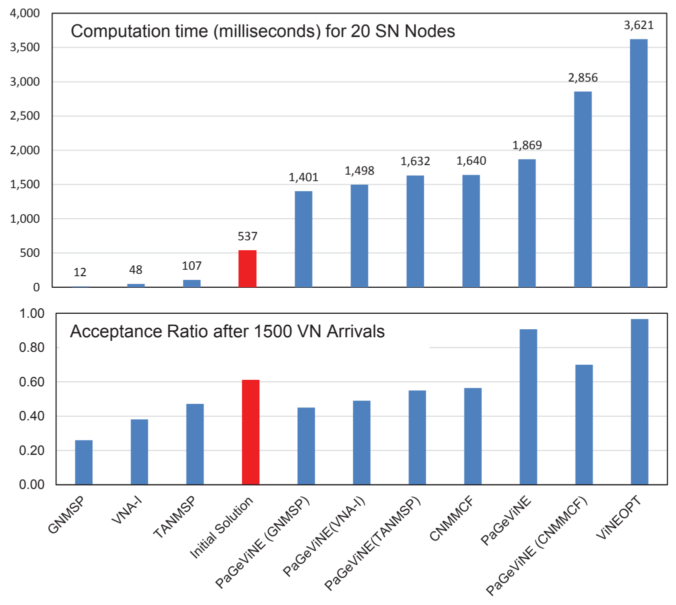


Fig. 18: Evaluation of the Initial Solution

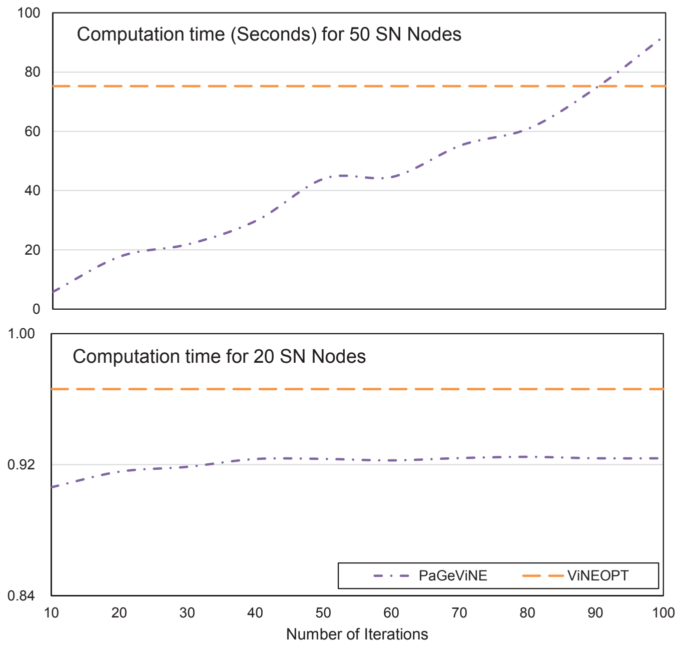


Fig. 19: Effect of Number of Iterations

Even more, it outperforms CNMMCF both on computation time as well as solution quality. The reason for this slightly better performance can be attributed to the fact that in CNMMCF node mapping is finalized by mapping each virtual node individually, which could sometimes lead to failures in embedding especially if more than one virtual node compete for a given substrate node. We also evaluated the performance of PaGeViNE in case the initial solution is changed. For example, PaGeViNE(GNMSP) means that GNMSP is used to determine the initial solution before applying path generation. These results show that PaGeViNE is dependent on an initial solution for both solution quality as well as computation time. With regard to the computation time, this dependence can be explained by the fact that the initial solution as well as the main PaGeViNE mathematical programs are solved sequentially. This means that if the computation of the initial solution takes longer, the overall solution will take longer. Similarly, since we do not allow the algorithm to run to completion, the quality of the initial solution will determine that of the final solution in two ways (1) in some cases, the initial solution just fails to even find a start solution, or (2) if the obtained initial solution is not good enough, the improvement in one iteration is not as good as it could be. These aspects are all confirmed by Fig. 18.

Effect of Number of Iterations: Finally, Fig. 19 is aimed at justifying our decision to perform a single iteration rather than having an iterative approach. From the Fig. we can observe that as the number of iterations is increased, the computation time increases more rapidly than does the solution quality.

E. Limitations

The mathematical formulation (1)−(7) involves solving a binary program. This problem is NP-hard in the general case, and only exponential algorithms are known to solve it in practice [31]. Our approach is to reduce the number of input variables to the program using path generation. While a significant improvement in computation time is achieved compared to the optimal solution, more work can be done for instance seeking a relaxation to the program which permits to solve it in polynomial time. We however note that in practice there are *high performance* tools [30] for solving binary programs. In particular, we have noted that the initial solution also contributes significantly to the overall computation complexity, and hence a more efficient heuristic for the same purpose could possible further enhance the results obtained in this paper. In addition, it would be interesting to make a mathematical analysis on the bounds of the computation time savings achieved in this paper.

VII. Conclusion

In this paper we have proposed a VNE solution which differs from previous solutions by performing node and link mappings in one shot using optimization theory and path generation. Our path generation based approach first obtains an initial solution by coordinating the node and link mapping stages, and then enhances this solution by carrying out only one round of pricing for the dual variables to obtain the final solution. Through extensive simulations, we have shown that our approach has a comparative advantage over previous approaches in terms of solution quality, achieving a comparatively superior acceptance ratio as well as VNE revenue, which directly leads to higher profitability for SN providers. The acceptance ratio is atleast 95% of that obtained by the optimal solution. In addition, our approach significantly reduces solution computation time compared to the optimal one (achieving a 92% saving in computation time for SNs of

50 nodes), and that this time complexity is comparable to that of related works.

Looking at the future, there are several possible research avenues. With regard to time complexity, it would also be interesting to propose relaxations to the mathematical programs in order to ensure polynomial time convergence. For this purpose, we are currently investigating the feasibility of using a combination of Tabu Search and path relinking to further improve the solution time. In addition, to optimize resource allocations over time, we are exploring possibilities of modeling the substrate network state as a markovian decision process [32], and by assigning state probabilities and transition rewards be able to bias the mapping of virtual resources to more appropriate resources. Finally, we intend to extend our proposed solution to a multi-domain VNE scenario [33] and to consider failures in the SN [34], [35].

Acknowledgement

The authors are grateful to the editors as well as anonymous reviewers whose insightful comments and suggestions led to a significant improvement in this paper. This work has been supported in part by FLAMINGO, a Network of Excellence project (318488) supported by the European Commission under its Seventh Framework Programme and project TEC2012-38574-C02-02 from Ministerio de Economia y Competitividad.

Appendix
ViNEOPT

This is the link based formulation of the one-shot optimal VNE problem. We define f_{uv}^{ij} as the flow of a virtual link $l_{ij} \in L_v$ on the link $l_{uv} \in (L_s \cup L_x)$. L_x is the set of all meta links in the augmented SN.

$$\text{minimize} \sum_{l_{ij} \in L_v} \sum_{l_{uv} \in (L_s \cup L_x)} \frac{1}{A_{uv}} f_{uv}^{ij} + \sum_{n_v \in N_v} \sum_{n_s \in N_s} \frac{1}{A_{n_s}} \chi_{n_s}^{n_v}$$

subject to

Node Mapping Constraints

$$\sum_{n_s \in N_s} \chi_{n_s}^{n_v} = 1 \qquad \forall n_v \in N_v$$

$$\sum_{n_v \in N_v} \chi_{n_s}^{n_v} \leq 1 \qquad \forall n_s \in N_s$$

$$f_{uv}^{ij} - D_{ij}\chi_u^i \leq 0 \qquad \forall uv \in L_x, \forall l_{ij} \in L_v$$

$$f_{uv}^{ij} - D_{ij}\chi_v^j \leq 0 \qquad \forall uv \in L_x, \forall l_{ij} \in L_v$$

Capacity Constraints

$$\sum_{ij \in L_v} f_{uv}^{ij} \leq A_{uv} \qquad \forall l_{uv} \in (L_s \cup L_x)$$

$$\sum_{uv \in L_v} f_{uv}^{ij} = D_{ij} \qquad \forall l_{ij} \in L_v$$

Flow Conservation Constraints

Source Nodes

$$\sum_{k \in N_s} f_{ik}^{ij} - \sum_{k \in N_s} f_{ki}^{ij} = D_{ij} \qquad \forall l_{ij} \in L_v$$

Sink Nodes

$$\sum_{k \in N_s} f_{jk}^{ij} - \sum_{k \in N_s} f_{kj}^{ij} = -D_{ij} \qquad \forall l_{ij} \in L_v$$

Intermediate Nodes

$$\sum_{u \in N_s} f_{uv}^{ij} - \sum_{u \in N_s} f_{uv}^{ij} = 0 \qquad \forall l_{ij} \in L_v, \forall v \in N_s$$

Domain Constraints

$$f_{uv}^{ij} = [0, D_{ij}] \qquad \forall l_{ij} \in L_v, \forall l_{uv} \in (L_s \cup L_x)$$

$$\chi_u^i = [0, 1] \qquad \forall i \in N_v, \forall u \in N_s$$

References

[1] A. Fischer, J. Botero, M. Till Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 4, pp. 1888–1906, Fourth 2013.

[2] M. Chowdhury, M. Rahman, and R. Boutaba, "Vineyard: Virtual network embedding algorithms with coordinated node and link mapping," *Networking, IEEE/ACM Transactions on*, vol. 20, no. 1, pp. 206 –219, feb. 2012.

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

[4] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, 2006, pp. 1–12.

[5] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance, "Branch-and-price: Column generation for solving huge integer programs," *Operations Research*, vol. 46, no. 3, pp. pp. 316–329, 1998.

[6] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*, 2nd ed. McGraw-Hill Higher Education, 2001.

[7] J. Lu and J. Turner, "Efficient Mapping of Virtual Networks onto a Shared Substrate," Washington University in St. Louis, Tech. Rep., 2006.

[8] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology-aware node ranking," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, pp. 38–47, Apr. 2011. [Online]. Available: http://doi.acm.org/10.1145/1971162.1971168

[9] J. He, R. Zhang-shen, Y. Li, C. yen Lee, J. Rexford, and M. Chiang, "Davinci: Dynamically adaptive virtual networks for a customized internet," in *in Proc. CoNEXT*, 2008.

[10] R. Mijumbi, J.-L. Gorricho, J. Serrat, M. Claeys, F. De Turck, and S. Latre, "Design and evaluation of learning algorithms for dynamic resource management in virtual networks," in *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS)*, ser. NOMS2014. IEEE, 2014.

[11] R. Mijumbi, J.-L. Gorricho, J. Serrat, M. Claeys, F. De Turck, and J. Famaey, "Neural network-based autonomous allocation of resources in virtual networks," in *Proceedings of the European Conference on Networks and Communications (EuCNC)*, ser. EuCNC2014, June 2014.

[12] R. Mijumbi, J.-L. Gorricho, J. Serrat, K. Xu, M. Shen, and K. Yang, "A neuro-fuzzy approach to self-management of virtual network resources," *Journal of Expert Systems With Applications*, Feb 2015.

[13] I. Houidi, W. Louati, and D. Zeghlache, "A distributed virtual network mapping algorithm." in *ICC*. IEEE, 2008, pp. 5634–5640.

[14] J. Infhr and G. R. Raidl, "Introducing the virtual network mapping problem with delay, routing and location constraints." in *INOC*, ser. Lecture Notes in Computer Science, J. Pahl, T. Reiners, and S. Vo, Eds., vol. 6701. Springer, 2011, pp. 105–117.

[15] G. Schaffrath, S. Schmid, and A. Feldmann, "Generalized and resource-efficient vnet embeddings with migrations," *CoRR*, vol. abs/1012.4066, 2010.

[16] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proceedings of the 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures*, ser. VISA '09. New York, NY, USA: ACM, 2009, pp. 81–88.

[17] H. Yu, C. Qiao, V. Anand, X. Liu, H. Di, and G. Sun, "Survivable virtual infrastructure mapping in a federated computing and networking system under single regional failures." in *Global Telecommunications Conference (GLOBECOM)*. IEEE, 2010, pp. 1–6.

[18] C. Wang and T. Wolf, "Virtual network mapping with traffic matrices," in *Architectures for Networking and Communications Systems (ANCS), 2011 Seventh ACM/IEEE Symposium on*, Oct 2011, pp. 225–226.

[19] I. Houidi, W. Louati, W. Ben Ameur, and D. Zeghlache, "Virtual network provisioning across multiple substrate networks," *Comput. Netw.*, vol. 55, no. 4, pp. 1011–1023, mar 2011.

[20] A. Jarray and A. Karmouch, "Decomposition approaches for virtual network embedding with one-shot node and link mapping," *Networking, IEEE/ACM Transactions on*, vol. PP, no. 99, pp. 1–1, 2014.

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

[22] R. Xie, F. R. Yu, and H. Ji, "Dynamic resource allocation for heterogeneous services in cognitive radio networks with imperfect channel sensing." *IEEE T. Vehicular Technology*, vol. 61, no. 2, pp. 770–780, 2012.

[23] J. Botero, X. Hesselbach, M. Duelli, D. Schlosser, A. Fischer, and H. De Meer, "Energy efficient virtual network embedding," *Communications Letters, IEEE*, vol. 16, no. 5, pp. 756–759, May 2012.

[24] D. Santos, A. de Sousa, F. Alvelos, and M. Pioro, "Link load balancing optimization of telecommunication networks: A column generation based heuristic approach," in *Telecommunications Network Strategy and Planning Symposium (NETWORKS), 2010 14th International*, Sept 2010, pp. 1–6.

[25] R. B. M. Pfetsch and C. Liebchen, "Lecture notes on multi-commodity flows and column generation," February 2006.

[26] M. X. Goemans and D. P. Williamson, "Approximation algorithms for np-hard problems," D. S. Hochbaum, Ed. Boston, MA, USA: PWS Publishing Co., 1997, ch. The Primal-dual Method for Approximation Algorithms and Its Application to Network Design Problems, pp. 144–191. [Online]. Available: http://dl.acm.org/citation.cfm?id=241938.241942

[27] S. Lahaie, "How to take the Dual of a Linear Program," www.cs.columbia.edu/coms6998-3/lpprimer.pdf?, 2008, Accessed: 2014-02-17.

[28] M. Güzelsoy and T. K. Ralphs, "Integer programming duality," in *Encyclopedia of Operations Research and Management Science*, J. Cochran, Ed. Wiley, 2010. [Online]. Available: http://coral.ie.lehigh.edu/~ted/files/papers/Duality-EOR10.pdf

[29] A. Medina, A. Lakhina, I. Matta, and J. Byers, "Brite: An approach to universal topology generation," in *Proceedings of the Ninth International Symposium in Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, ser. MASCOTS '01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 346–353.

[30] "IBM ILOG CPLEX Optimizer," http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/about/, Accessed: 2014-02-17.

[31] G. J. Woeginger, "Combinatorial optimization - eureka, you shrink!" M. Jünger, G. Reinelt, and G. Rinaldi, Eds. New York, NY, USA: Springer-Verlag New York, Inc., 2003, ch. Exact Algorithms for NP-hard Problems: A Survey, pp. 185–207. [Online]. Available: http://dl.acm.org/citation.cfm?id=885909.885927

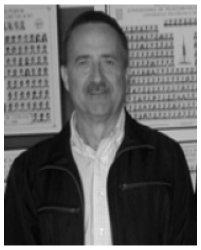
[32] K. W. Ross, *Multiservice Loss Models for Broadband Telecommunication Networks*, P. J. Hancock, Ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1995.

[33] F. Samuel, M. Chowdhury, and R. Boutaba, "Polyvine: policy-based virtual network embedding across multiple domains," *Journal of Internet Services and Applications*, vol. 4, no. 1, 2013. [Online]. Available: http://dx.doi.org/10.1186/1869-0238-4-6

[34] R. Mijumbi, J.-L. Gorricho, J. Serrat, J. Rubio-Loyola, and R. Aguero, "Survivability-oriented negotiation algorithms for multi-domain virtual networks," in *Network and Service Management (CNSM), 2014 10th International Conference on*, Nov 2014, pp. 276–279.

[35] Z. Ye, A. Patel, P. Ji, and C. Qiao, "Survivable virtual infrastructure mapping over transport software-defined networks (t-sdn)," in *Optical Fiber Communications Conference and Exhibition (OFC), 2014*, March 2014, pp. 1–3.

Rashid Mijumbi obtained a Bachelors of Science Degree in Electrical Engineering from Makerere University (Kampala, Uganda) in 2009, and a PhD in Telecommunications Engineering from the Universitat Politècnica de Catalunya (UPC) (Barcelona, Spain) in 2014. He is currently a Postdoctoral Researcher in the Network Engineering Department at UPC. His research interests are in management of networks and services for the future Internet. Current focus is on resource management in virtualized networks and functions, software defined networks and cloud computing.



Joan Serrat received a degree of telecommunication engineering in 1977 and a PhD in the same field in 1983, both from UPC. Currently, he is a full professor at UPC where he has been involved in several collaborative projects with different European research groups, both through bilateral agreements or through participation in European funded projects. His topics of interest are in the field of autonomic networking and service and network management. He is the contact point of the TM Forum at UPC.

Juan-Luis Gorricho received a telecommunication engineering degree in 1993, and a Ph.D. degree in 1998, both of them from the Technical University of Catalonia (UPC). Since 1994 he joined the Department of Network Engineering at the UPC as an assistant professor, and as associate professor since 2001. His most recent research interests have been focused on applying artificial intelligence to the research fields of ubiquitous computing and network management; with special interest on using smartphones to achieve the recognition of user activities and locations; and applying linear programming and reinforcement learning to solve the network embedding problem and the resource allocation problem, targeting the implementation of the network virtualization and the future network function virtualization.

Raouf Boutaba received the MSc and PhD degrees in computer science from the Université de Pierre et Marie Curie, Paris, France, in 1990 and 1994, respectively. He is currently a full professor of computer science at the University of Waterloo, Waterloo, ON, Canada, and a distinguished visiting professor at the Pohang University of Science and Technology (POSTECH), Korea. His research interests include network, resource and service management in wired and wireless networks. He has received several best paper awards and other recognitions such as the Premier's Research Excellence Award, the IEEE Hal Sobol Award in 2007, the Fred W. Ellersick Prize in 2008, the Joe LociCero and the Dan Stokesbury awards in 2009, and the Salah Aidarous Award in 2012. He is a fellow of the IEEE and the Engineering Institute of Canada.