Journal of Network and Computer Applications 69 (2016) 14–26

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

# Coordinated node and link mapping VNE using a new paths algebra strategy

Xavier Hesselbach [a,*], Josè Roberto Amazonas [b], Santi Villanueva [a], Juan Felipe Botero [c]

[a] Universitat Politècnica de Catalunya, Jordi Girona Street, 31, Barcelona, Spain
[b] Escola Politécnica of the University of São Paulo, SP, Brazil
[c] University of Antioquia, Street 67 53 - 10, Medellin, Colombia

ARTICLE INFO

*Article history:*
Received 23 June 2015
Received in revised form
10 February 2016
Accepted 17 February 2016
Available online 24 April 2016

*Keywords:*
Network virtualization
Virtual network embedding
Multi-constraint routing
Paths algebra

ABSTRACT

The main resource allocation research challenge in network virtualization is the Virtual Network Embedding (VNE) Problem. It consists of two stages, usually performed separately: node mapping and link mapping. A new mathematical multi-constraint routing framework for linear and non-linear metrics called "paths algebra" has already been proposed to solve the second stage, providing good results thanks to its flexibility. Unlike existing approaches, paths algebra is able to include any kind of network parameters (linear and non-linear) to solve VNE in reasonable runtime. While paths algebra had only been used to solve one stage (link mapping) of the VNE, this paper suggests an improvement to solve VNE using the paths algebra-based strategy by coordinating, in a single stage, the mapping of nodes and links based on a ranking made of the bi-directional pair of nodes of the substrate network, ordered by their available resources. Simulation results show that the New Paths Algebra-based strategy shows significant performance improvements when compared against the uncoordinated paths Algebra-based link mapping approach.



## 1. Introduction

Network virtualization (NV) was proposed as an enabling technology for the future Internet architecture as the deployment of new Internet services is increasingly difficult due to the lack of cooperation among stakeholders. NV allows multiple heterogeneous, logical networks to cohabit on a shared substrate network (SN). On top of the SN, several virtual networks run and offer end-to-end customized quality of service (QoS) to the end users (EUs).

NV needs a sustainable economic model to be implemented. Infrastructure as a service (IaaS) (Bhardwaj et al., 2010) is a business model where the hardware and associated software operating systems are delivered as a service. The introduction of the IaaS paradigm is changing the current infrastructure service providers (ISPs)-based Internet business model (Feamster et al., 2007). This new business model is well suited for the future dynamics in networking service requests as NV will be a fundamental enabler to provide end-to-end QoS guarantees in an IaaS based network architecture. It will be not just a technology to implement IaaS, but a component of the architecture itself (Botero Vega, 2013).

Fig. 1 shows a typical network virtualization environment. One of the main challenges of NV is the efficient allocation of virtual network elements on top of SN elements, commonly known as the virtual network embedding/mapping (VNE) problem (Fischer et al., 2013; Botero and Hesselbach, 2009; Haider et al., 2009). This can be divided into two stages: virtual node mapping (VNoM) and virtual link mapping (VLiM). In the VNoM stage virtual nodes have to be allocated in physical nodes; in the VLiM stage, virtual links connecting virtual nodes have to be mapped to paths connecting the corresponding nodes in the substrate network. The two stages of the VNE problem can be solved either in an uncoordinated or in a coordinated way.

This problem has been recently addressed in the works from Zhu and Wang (2014) using a different view. It was focused on processing multiple VN requests simultaneously, proposing a modified ant colony optimization (ACO) algorithm, where a set of candidate physical networks are first found for each VN request, and the VNE problem is converted to a multiple-choice knapsack problem. In this work, the optimization goal is to maximize revenue and to achieve optimal load-balancing characteristics, expecting to support more virtual network requests in the same substrate network. However, in our work, cost/revenue is considered as a primary target, and a different virtual network request

* Corresponding author.
*E-mail addresses:* xavierh@entel.upc.edu (X. Hesselbach), jra@lcs.poli.usp.br (J.R. Amazonas), santi.villanueva@alu-etsetb.upc.edu (S. Villanueva), juanf.botero@udea.edu.co (J.F. Botero).

http://dx.doi.org/10.1016/j.jnca.2016.02.025

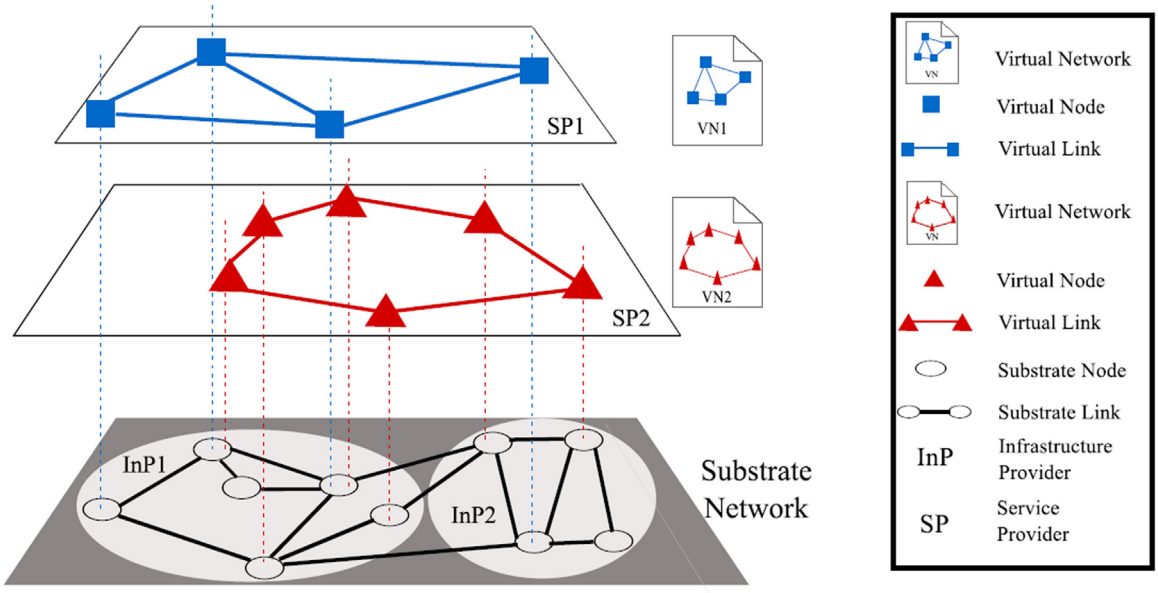


**Fig. 1.** Network virtualization environment.

(VNR) generation model is used. Therefore, only qualitative comparisons are possible between both works.

To solve the virtual link mapping stage of the VNE problem a novel paths algebra-based strategy was proposed in Botero et al. (2013a). This strategy introduces an unlimited number of linear and non-linear constraints and metrics to the VNE providing the necessary flexibility for better embeddings.

This new strategy can be improved by solving the mappings of nodes and links in a new coordinated way. It is proved in this paper that coordinated mapping provides excellent results by ranking the nodes in the node mapping stage. This new paths algebra-based strategy is called New Paths Algebra (NPA).

This work will present the improvements and better results provided by this NPA for scenarios of several sizes and loads.

The main contributions of this paper are:

1. Analyze the advantages of the coordinated link and node mapping strategy.
2. Define a paths algebra based strategy for the coordinated link and node mapping for linear and non-linear parameters.
3. Analyze and compare the performance of the New Paths Algebra algorithm simulating small and big scenarios.

Section 2 details the previous work in VNE and paths algebra, and highlights the shortcomings that motivate the introduction of the NPA VNE approach. Section 3 details the NPA VNE approach. Section 4 evaluates by simulation the proposed approach highlighting its improvements in acceptance ratio and revenue when compared with existing solutions. Section 5 presents the conclusions and future work.

## 2. Related work

This section presents the main approaches to solve the VNE problem and shows their inability to provide solutions when non-linear constraints are involved. It also describes the previous work in the "paths algebra" mathematical framework and how it can be used to solve the VNE problem.

### 2.1. Network virtualization and virtual network embedding

According to Botero et al. (2013a), Chowdhury (2009), Chowdhury and Boutaba (2010), and Fischer et al. (2013) one of the main challenges of network virtualization is the virtual network embedding problem (VNE), consisting on mapping virtual network demands in physical network resources.

Virtual networks (VNs) must be optimally allocated over the physical/substrate network (SN). A scenario will contain an SN and a group of VNs, called virtual network requests (VNRs), each one containing a set of resources required by the demanded service. The main problem of the VNE consists of the optimal allocation of this VNRs on top of the SN according to some predefined objective. In the virtual node mapping stage each virtual node of a VNR is mapped to one substrate node with enough capacity to meet the virtual node resource demand; in the virtual link mapping stage each virtual link is mapped to a directed path in the SN with enough resource capacity to meet the virtual link demand.

VNRs consume physical resources: CPU processing power on nodes and BW capacity on links of the substrate network. Physical resources are finite and represent the limit of VN assignments. Hidden hops must also be considered in the link mapping stage. Hidden hops, first introduced in Botero et al. (2013b), are the intermediate nodes of a directed path in the SN that is mapping a specific virtual link of a VNR. Hidden hops entail a resource demand for forwarding the traffic that will pass through it and will consume CPU resources of the SN. Fig. 2 illustrates the embedding of two VNRs on an SN. Both virtual nodes and virtual links share resources of the SN.

The VNE problem is $\mathcal{NP}$-hard. Exact approaches are not scalable and are only applicable in small network scenarios (Houidi et al., 2011, 2015; Botero et al., 2012; Botero and Hesselbach, 2013; Shanbhag et al., 2015; Mechtri et al., 2015; Dietrich et al., 2015). Current solutions are based on heuristics (Yu et al., 2008; Cheng et al., 2011; Till Beck et al., 2013; Xiao et al., 2014; Zhang et al., 2014; Guan et al., 2015; Jian et al., 2015; Beck et al., 2015) or metaheuristics (Fajjari et al., 2011; Zhang et al., 2012, 2015; Cheng et al., 2012; Su et al., 2014). Many VNE algorithms try to maximize the number of embedded VNRs by reducing embedding costs (Cheng et al., 2012; Fajjari et al., 2011; Chowdhury et al., 2012; Lischka and Karl, 2009). None of these algorithms consider non-linear constraints.

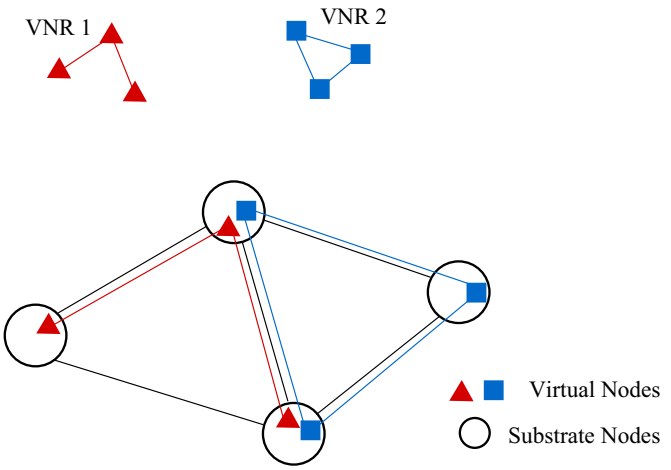


**Fig. 2.** Embedding of two Virtual Network Requests.

To deal with non-linear constraints, our previous work (Botero et al., 2013a) has proposed to solve the VLiM stage using the mathematical multi-constraint routing framework called "paths algebra". This strategy provides the flexibility to introduce an unlimited number of linear and non-linear constraints and metrics to the VNE, resulting in better and more flexible embeddings. However, this approach solves the VLiM separately and does not account for the VNoM stage.

To solve this problem, we propose a new coordinated approach that solves VLiM in a coordinated way with VNoM using the paths algebra framework.

### 2.2. Paths algebra

Paths algebra is a powerful and elegant mathematical framework for solving the multi-constraint routing problem using linear metrics as bandwidth, number of hops and delay, or non-linear metrics as availability and package loss rate; or even a combination of both. However, the main important metrics for the optimization of the VNE problem may be cost and revenue, as network virtualization is mainly a business strategy. These metrics can also be taken into account by the paths algebra framework.

Depending on the optimization goal of the link mapping stage, the metrics can be divided in terms of Quality of Service (QoS) (usage, throughput, delay, jitter, etc.), Quality of Network Economics (QoNE) (cost, revenue), Quality of Resilience (QoR) (number of backups, path redundancy, etc.) or others (runtime, number of messages, etc.). We will evaluate the performance of the resources assignment strategy based on the cost and revenue, and two other metrics derived from these two: Cost/Revenue and acceptance ratio.

Cost/Revenue describes the relationship between spent substrate resources and demanded virtual resources, while the acceptance ratio measures the number of virtual network requests that could be completely embedded by the embedding algorithm, divided by the total number of virtual network requests.

The paths algebra multi-constraint routing algorithm is implemented by the Loop Avoidance by the Destination Node – LADN software (de Paula Herman and de Almeida Amazonas, 2007) that finds all possible paths between each pair of nodes in the SN and orders them based on an unlimited number of constraints. The LADN consists of four stages:

- SEARCHPATH: discovers all paths between each pair of SN nodes;
- SORTPATH: selects only cycle-free paths;
- EVALUATEPATH: characterizes each path based on the defined link parameters;
- ORDERPATH: orders the paths according to the defined metrics and priorities.

The first stage (SEARCHPATH) can be an expensive procedure for highly connected networks, but it can be performed offline and the computing time can be neglected. The processing time can also be reduced using a filter that finds the minimum set of paths that ensure full network's connectivity (Botero et al., 2013a).

The VLiM stage can be performed in different ways to allocate as many VN requests to the SN as possible: FIFO, that is the procedure that emulates the online VNE; non-decreasing, or non-increasing, BW order; non-decreasing, or non-increasing, CPU order; non-decreasing, or non-increasing, (BW+CPU)[1] order, etc. Depending on the optimization criterion, the policy that provides the best results may change, so it is not possible to say what is the best policy in advance. In this paper, the policy used in the simulations was non-increasing (BW+CPU) order.

Once the VLiM stage is performed, a virtual link is assigned to the best path of the SN and its resources are updated according to the actual values consumed by the VNR. This procedure is repeated until all virtual links have been assigned to the SN.

If there are no available resources to map a virtual link to the SN, different strategies can be adopted: (i) the VN request can be dropped out; (ii) it can be tried to accommodate it with less resources than originally demanded; (iii) it may be possible to perform backtrack (re-assign a previous virtual link in order to accommodate another one). In the simulations done in this paper, VN requests were dropped out if they could not be mapped and the other possibilities will be analyzed in future work.

#### 2.2.1. Paths characterization

A network is represented by a directed graph $G = (V, A)$, where $V$ is the set of vertices and $A$ the set of arcs. Consider the simple path represented in Fig. 3.a. The set of vertices is given by $V = \{1, 2, 3, 4\}$ and the set of arcs is given by $A = \{a, b, c\}$. The source and destination nodes are $(s, d) = (1, 4)$. This path can be represented either as a succession of vertices $p_{1,4}$ or as a succession of arcs $p_{a,c}$.

Each arc in this example is characterized by a triple $(m_1(x), m_2(x), f[m_1(x), m_2(x)])$, where: $m_1(x)$ and $m_2(x)$ are the values of metrics $m_1$ and $m_2$ on the arc $x \in A$; $f[m_1(x), m_2(x)]$ is a function of combination of metrics applied to $m_1(x)$ and $m_2(x)$.

In general, the paths algebra uses $\mathbf{M}$ as the set of $m$ adopted routing metrics and $\mathbf{F}$ as the set of $k$ metrics combination functions.

The set of combined-metrics of all edges is given by:

$$\overline{C}(p_{a,c}) = \begin{bmatrix} \overline{C}_a \\ \overline{C}_b \\ \overline{C}_c \end{bmatrix} = \begin{bmatrix} m_1(a) & m_2(a) & f[m_1(a), m_2(a)] \\ m_1(b) & m_2(b) & f[m_1(b), m_2(b)] \\ m_1(c) & m_2(c) & f[m_1(c), m_2(c)] \end{bmatrix}$$

A synthesis $\overline{S}[\cdot]$ is a set of binary operations applied on the values of the links combined-metrics along a path to obtain a resulting value that characterizes this path as far as the constraint imposed by the combined-metric is concerned. So far, the syntheses are restricted to the following set: {add(), mult(), max(),min ()}).

If the routing algorithm is mono-constraint, only one value is

[1] BW and CPU, in the context of the VNE problem, are specified in terms of capacity units. For example, 100 CPU units=2.66 GHz and 100 BW units=1 Gbps. Using these scaled units, CPU and BW resources/demands can be added in spite of being variables of different nature.

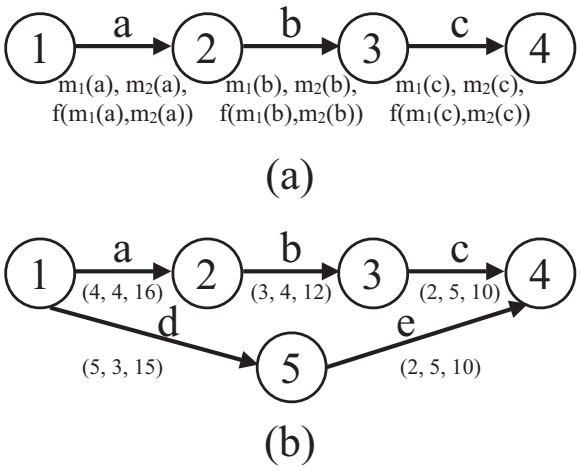


**Fig. 3.** (a) Example of a simple path (b) Example of two paths to be ordered.

obtained as the synthesis result and it is called weight-word. If the routing algorithm is multi-constraint, with $k$ constraints, then $k$ values are obtained. In this example, $\overline{S}[\cdot] = [S_1S_2S_3]^t$. The weight-word has as many letters as the path's number of arcs. The first letter corresponds to resulting value of the synthesis applied to the whole path; the second letter corresponds to resulting value of the synthesis applied to the subpath obtained by dropping out the last arc; the last letter corresponds to the resulting value of the synthesis applied to the subpath made of only the first arc. Any number of letters can be retained as the synthesis result and this is called an abbreviation: $\overline{b}_j(\overline{S}[\cdot])$ represents a $j$-letters abbreviation; $\overline{b}_\infty(\overline{S}[\cdot])$ represents no abbreviation, i. e., all letters are taken into account.

*2.2.2. Paths ordering*

Consider the network represented in Fig. 3b where two paths connect the source node 1 to the destination node 4. These paths are $\alpha = (1, 2, 3, 4) = (a, b, c)$ and $\beta = (1, 5, 4) = (d, e)$. Each paths' arc is characterized by a triple $(m_1(x), m_2(x), f[m_1(x), m_2(x)])$, where $f[m_1(x), m_2(x)] = m_1(x) \times m_2(x)$. The syntheses to be used in this example are given by $\overline{S}[\cdot] = [\min()\max()\text{add}()]^t$.

The result of the synthesis is shown in Table 1. A path $\alpha$ is worse or less optimized than a path $\beta$, if $\overline{S}[\alpha] \preceq_{ML} \overline{S}[\beta]$, where $\preceq_{ML}$ stands for multidimensional lexical ordering. In the example $\preceq_{ML} = \{\geq, \leq, \geq\}$, that is translated by the following ordering relations:

- $S_1[\alpha] \preceq S_1[\beta] \Rightarrow S_1[\alpha] \geq S_1[\beta]$;
- $S_2[\alpha] \preceq S_2[\beta] \Rightarrow S_2[\alpha] \leq S_2[\beta]$;
- $S_3[\alpha] \preceq S_3[\beta] \Rightarrow S_3[\alpha] \geq S_3[\beta]$.

Different syntheses also have different priorities. In the example, $S_1$, $S_2$ and $S_3$ priorities go from the highest to the lowest.

Table 2 summarizes the results obtained for three different ordering criteria. It is important to realize that the syntheses letters are examined from the highest priority to the lowest priority

**Table 1**
Synthesis result of the network given in Fig. 3b.

| Path | $S_1$ min | $S_2$ max | $S_3$ add |
|---|---|---|---|
| $\alpha$ | 2; 3; 4 | 5; 4; 4 | 38; 28; 16 |
| $\beta$ | 2; 5 | 5; 3 | 25; 15 |

**Table 2**
Paths ordering of the network given in Fig. 3b.

| Abbreviation $\overline{b}_j(\overline{S}[\cdot])$ | Result |
|---|---|
| $\overline{b}_1[S_1]$ $\overline{b}_1[S_2]$ $\overline{b}_1[S_3]$ | $S_1 \Rightarrow \alpha \equiv \beta$<br>$S_2 \Rightarrow \alpha \equiv \beta$<br>$S_3 \Rightarrow \alpha \prec \beta$ |
| $\overline{b}_\infty[S_1]$ $\overline{b}_\infty[S_2]$ $\overline{b}_\infty[S_3]$ | $S_1 \Rightarrow$ 1st letters are equal $\Rightarrow \alpha \equiv \beta$<br>$S_1 \Rightarrow$ 2nd letters $\Rightarrow$ $3 < 5 \Rightarrow \beta \prec \alpha$ |
| $\overline{b}_1[S_1]$ $\overline{b}_\infty[S_2]$ $\overline{b}_1[S_3]$ | $S_1 \Rightarrow \alpha \equiv \beta$<br>$S_2 \Rightarrow$ 1st letters are equal $\Rightarrow \alpha \equiv \beta$<br>$S_2 \Rightarrow$ 2nd letters $\Rightarrow$ $4 > 3 \Rightarrow \beta \prec \alpha$ |

synthesis. When the paths are considered equivalent, then we will examine either the next letter of the same synthesis or will move to the next synthesis. This is determined by the adopted abbreviation.

## 3. New paths algebra proposal

The paths algebra has been introduced in Botero et al. (2013a) to solve the link mapping stage of the VNE. In that work the VNoM stage is performed first using the algorithms available in the ALEVIN framework. The obtained results show that the paths algebra approach was never worse than the results reported in the literature and were almost always significantly better. However, performing node and link mapping in two separate stages may not make the best use of CPU and BW available resources and leaves room for further improvement.

In the past, node and link mapping have been done in 2 stages: First node mapping, next link mapping. This paper proposes a coordinated mapping of nodes and links at the same time. Since decisions are taken altogether for nodes and links, it is expected a better acceptance ratio and a reduction in the cost/revenue. The algorithm to carry out the node mapping stage together with the already mentioned paths algebra strategy for the link mapping stage is called New Paths Algebra (NPA). The New Paths Algebra study is based on a rank metrics based on the total resources of a paths scenario: the highest ranked nodes will be the destination of the mapped nodes from the VNRs. Resources from both the SN and the VNRs include nodes CPU and links bandwidth (BW). In this section, the proposal will be presented, and an example will numerically illustrate the mechanism in Section 3.1.

This development started with a basic idea that was implemented but, against our expectations, did not produce the expected results. To achieve the final results, we improved the original idea introducing modifications, evaluating the results and identifying the weaknesses. This cycle had to be repeated several times. This section will show the proposal and then introduce the improvements that were necessary to achieve good results. In this way, the interested reader that may want to provide further modifications to the algorithm may avoid the pitfalls that have already been faced. It has been a learning process worth to be described.

The algorithm is based on a rank metrics related to the total resources of a paths scenario. Using this metrics, the highest ranked nodes would be the destination of the mapping of nodes from the VNRs. Resources from both the SN and the VNRs include nodes CPU and links BW.

The algorithm also borrows the concept of *availability* of a communication channel. Availability is defined as the amount of

time a channel is available for transmission or, alternatively, as the probability that a channel is available in a certain instant of time.

In this work, the availability for a path $k$ between a pair of nodes $(s,d)$ is a measure of the path's available resources, i.e., a measure of the probability that the path has enough resources to be used by the VNR. This measure is defined in:

$$a_{(s,d)}(k) = \frac{CPU(s)}{t_{CPU}} \times \frac{\min_{BW(s,d)}(k)}{t_{BW}} \times \frac{CPU(d)}{t_{CPU}} \times x_{(s,d)}(k), \quad (1)$$

where:

- $t_{CPU}$ is the total network's available CPU resource;
- $t_{BW}$ is the total network's available BW resource;
- $CPU(n)$ is the CPU associated to node $n$;
- $\min_{BW(s,d)}(k)$ is the minimum available BW for a path $k$ between the pair of nodes $(s, d)$;
- $x_{(s,d)}(k)$ is the weighting factor to be applied to path $k$ between the pair of nodes $(s, d)$ to penalize the longer paths as they increase the cost of the embedding solution.

Eq. (1) has a multiplicative form because the probability of having enough resources to accommodate a VNR depends on the probability of having CPU and BW enough resources. As these resources are independent, the total probability is given by the multiplication of their respective probabilities.

The weighting factor is defined by Eq. (2):

$$x_{(s,d)}(k) = \frac{1}{b^{(l_{(s,d)}(k) - l_{(s,d)}(sh))}}, \quad (2)$$

where:

- $l_{(s,d)}(k)$ is the length of path $k$, measured in the number of links, between the pair of nodes $(s, d)$;
- $l_{(s,d)}(sh)$ is the length of the shortest path $sh$, measured in number of links, between the pair of nodes $(s, d)$;
- $b \geq 1$ represents the intensity of the penalty. For $b=1$, longer paths are not penalized. In this paper, we set $b=2$.

For the shortest path $sh$ between any pair of nodes $(s, d)$, $x_{(s,d)}(sh) = 1$, independently of its length. Then, the availability between a pair of nodes $(s,d)$ is obtained summing all availabilities for all existing paths $K$ between the pair of nodes $(s,d)$:

$$A_{(s,d)} = \sum_{k \in K} a_{(s,d)}(k). \quad (3)$$

The last step is to sum both availabilities of the pair of nodes $(s, d)$ and $(d,s)$ to get the equivalent of the one-way availability of a transmission channel. It is designated as bi-directional availability.:

$$A_{(s,d),(d,s)} = A_{(s,d)} + A_{(d,s)}. \quad (4)$$

Therefore, the first proposed algorithm is:

1. Given an SN, using the paths algebra, enumerate all paths between all pairs of nodes. In this step filtering can be used to guarantee connectivity and limit the size of the problem (Botero et al., 2013a).
2. Evaluate the availability of each pair of nodes using the availability equations (1), (3) and (4).
3. Rank all pairs of nodes according to their decreasing order of availability.
4. Given a VNR, order the virtual demands according to the decreasing minimum required availability evaluated by Eq. (5):

$$(VNR)req_{av}(v_s, v_d) = CPU(v_s) \times BW(v_s, v_d) \times CPU(v_d), \quad (5)$$

where:

- $CPU(v_s)$ and $CPU(v_d)$ are the required normalized CPU[2] by the source and destination virtual nodes;
- $BW(v_s, v_d)$ is the required normalized BW[3] between the virtual nodes.

5. Select the first pair of virtual nodes and check the minimum and maximum demanded CPU resource.
6. Create a new list of ranked pair of nodes by:
   - deleting the pair of nodes that have nodes with available CPU less than the minimum required by the VNR;
   - deleting the pair of nodes in which neither of them has available CPU at least equal to the maximum required by the VNR.
7. Map the first pair of virtual nodes to the highest ranked pair of nodes in the SN;
8. Map the virtual link to a SN path selected by the paths algebra ordering using a chosen set of combined metrics.

In the case of a tie at point 3, they should be ranked according to their associated BWs. Given a node $n$, it can be associated to $BW_{\text{in}}(n)$, $BW_{\text{out}}(n)$ and $BW_{\text{total}}(n) = BW_{\text{in}}(n) + BW_{\text{out}}(n)$. A decision will be made based on a voting criterion, for instance, a node $n_1$ will be considered to have more BW than a node $n_2$ if, at least, two of its associated BWs are greater than those associated to the other node. If the three BW values are the same, then the nodes are equivalent.

### 3.1. Example scenario

The scenario shown in Fig. 4 is used to illustrate how the NPA algorithm works. In the figure we see the SN and the VNRs.

After step 3, the initial bi-directional rank for the initial state of the SN can be seen in Table 3. Table 4 shows the bi-directional rank for the VNR #1 at the conclusion of step 4.

Steps 5 and 6 check the minimum and maximum demanded CPU of the first pair of virtual nodes and create a new list of ranked pair of substrate nodes deleting those pairs that do not fulfill the required demands. As the minimum CPU of the pair of nodes (cd, dc) is 20 (from node c) and the maximum CPU is 70 (from node d), and all CPU resources of the SN nodes are 100, the list of ranked pair of SN nodes remains exactly the same.

As the VNR #1 (cd, dc) demand requires more resources than the (ac, ca) demand, it will be mapped to the first ranked pair of SN nodes (BD, DB).

The indication that node (c) is less demanding than node (d) is expressed by $c \preceq d$. In the same way, $B \preceq D$ indicates that B offers less resources than D. In fact, both nodes offer the same amount of CPU, but, as it will be seen, $BW(B) \preceq BW(D)$ according to the voting criterion.

Table 5 shows the BW resources of all substrate nodes while Table 6 shows the nodes ordering according to the offered BW resources and for the three criteria: inbound, outbound and total.

Then:

- according to the $BW_{in}(n)$: $BW(B) \preceq BW(D)$;
- according to the $BW_{out}(n)$: $BW(D) \preceq BW(B)$;
- according to the $BW_{total}(n)$: $BW(B) \preceq BW(D)$.
- Voting decision: $B \preceq D$.

Therefore, the result of the node mapping for the nodes (c,d) of the VNR #1 (step 7) is:

[2] Normalization is made dividing the actual CPU VNR by $t_{CPU}$.
[3] Normalization is made dividing the actual BW VNR by $t_{BW}$.

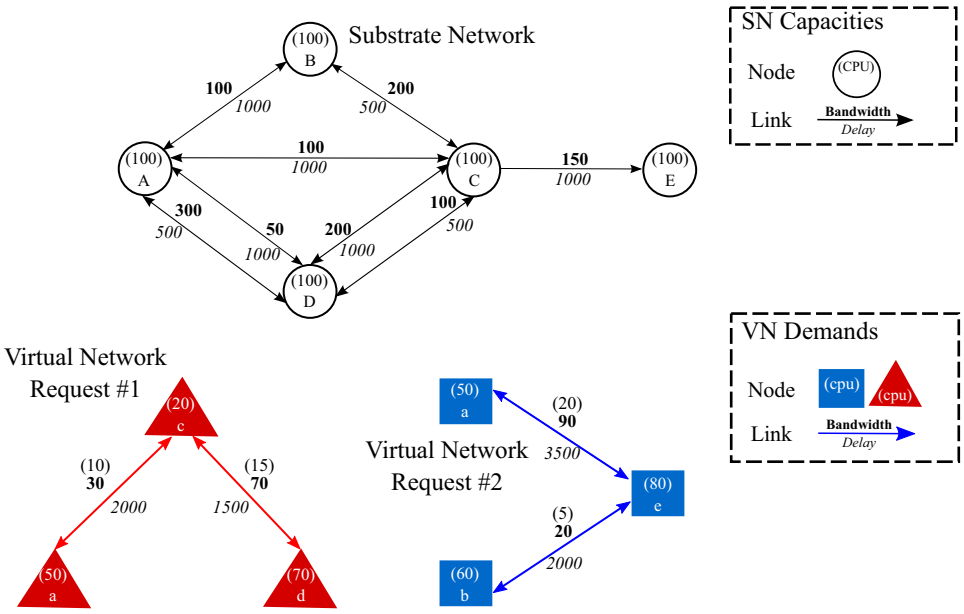


**Fig. 4.** Substrate network and virtual network requests.

**Table 3**
Bi-directional rank for the initial state of the SN.

| Rank | Pair(s) of nodes |
|---|---|
| 1 | BD, DB |
| 2 | BC, CB |
| 3 | AD, DA |
| 4 | BE, EB |
| 5 | AC, CA |
| 6 | CD, DC |
| 7 | AE, EA |
| 8 | AB, BA; DE, ED |
| 9 | CE, EC |

**Table 4**
Bi-directional rank for the VNR #1.

| Rank | Pair(s) of nodes |
|---|---|
| 1 | cd, dc |
| 2 | ac, ca |

**Table 5**
BW resources before the VNR #1 assignment.

| Node | Inbound BW $BW_{in}(n)$ | Outbound BW $BW_{out}(n)$ | Total BW $BW_{total}(n)$ |
|---|---|---|---|
| A | 250 | 500 | 750 |
| B | 300 | 300 | 600 |
| C | 650 | 550 | 1200 |
| D | 400 | 250 | 650 |
| E | 150 | 150 | 300 |

- Node mapping: **c** →**B**; **d** →**D**.

In step 8, the link mapping stage chooses the best path according to the metrics **M**={Hops, BW, CPU,} the corresponding syntheses $\overline{S}[\cdot] = [\text{add}()\min()\min()]^t$, and ordering relation $\preceq_{ML} = \{ \geq , \leq , \leq \}$. From B to D, both B-A-D and B-C-D have 2 hops and offer $BW_{min}$ = 100 and $CPU_{max}$ = 100. Both paths are equivalent and for this example the chosen path is B-C-D. From D to B, both

**Table 6**
Nodes ordering according to the offered BW resources before the VNR #1 assignment.

| Order | Inbound BW $BW_{in}(n)$ | Outbound BW $BW_{out}(n)$ | Total BW $BW_{total}(n)$ |
|---|---|---|---|
| 1 | C | C | C |
| 2 | D | A | A |
| 3 | B | B | D |
| 4 | A | D | B |
| 5 | E | E | E |

D-A-B and D-C-B have 2 hops, but $BW_{min}$(D-A-B)=50 < $BW_{min}$(D-C-B)=200, and, consequently, D-A-B ≺D-C-B. So the chosen path is D-C-B.

- Link mapping: (**c**, **d**) → (**B** – **C** – **D**); (**d**, **c**) → (**D** – **C** – **B**).

Then, the virtual nodes (a) and (c) have to be mapped to highest ranked pair of substrate nodes where one of the nodes is B (c was already mapped to B). The second pair shown in Table 3, (BC, CB) satisfies the condition and the result of the node and link mapping is:

- Node mapping: **a** →**C**;
- Link mapping: (**a**, **c**)→(**C**-**B**); (**c**, **a**)→(**B**-**C**).

The resources consumed for the mapping of VNR #1 can be seen in Table 7.

After the mapping of the VNR #1, nodes C and D had their

**Table 7**
Resources consumed after the VNR #1 mapping.

| Pair(s) of nodes | Node | CPU consumed | From node | BW consumed | From link |
|---|---|---|---|---|---|
| (cd, dc) | (c) | 20 | B | 70 | B-C |
| | (d) | 70 | D | 70 | C-D |
| | Hidden hop | 15 | C | 70 | D-C |
| | | | 70 | C-B | |
| (ac, ca) | (a) | 50 | C | 30 | C-B |
| | | | 30 | B-C | |

**Table 8**
Bi-directional rank of the SN after the VNR #1 mapping.

| Rank | Pair(s) of nodes |
|---|---|
| 1 | AE, EA |
| 2 | AB, BA; BE, EB |
| 3 | AC, CA |
| 4 | AD, DA |
| 5 | CE, EC |
| 6 | BD, DB |
| 7 | BC, CB |
| 8 | DE, ED |
| 9 | CD, DC |

**Table 9**
Bi-directional rank for the VNR #2.

| Rank | Pair(s) of nodes |
|---|---|
| 1 | ae, ea |
| 2 | be, eb |

**Table 10**
BW resources before the VNR #2 assignment.

| Node | Inbound BW $BW_{in}(n)$ | Outbound BW $BW_{out}(n)$ | Total BW $BW_{total}(n)$ |
|---|---|---|---|
| A | 250 | 500 | 750 |
| B | 200 | 200 | 400 |
| C | 480 | 380 | 860 |
| D | 330 | 180 | 510 |
| E | 150 | 150 | 300 |

**Table 11**
Nodes ordering according to the offered BW resources before the VNR #2 assignment.

| Order | Inbound BW $BW_{in}(n)$ | Outbound BW $BW_{out}(n)$ | Total BW $BW_{total}(n)$ |
|---|---|---|---|
| 1 | C | A | C |
| 2 | D | C | A |
| 3 | A | B | D |
| 4 | B | D | B |
| 5 | E | E | E |

resources drastically reduced (and, to a lesser extent, node B). In the new rank, shown on Table 8, all pairs of nodes with one of these nodes are ranked after the pairs of nodes in which they do not appear. This fact suggests that the proposed metrics is reflecting well the available resources.

Table 9 shows the bi-directional rank for the VNR #2.

Following the same procedure as before, VNR #2 (ae, ea) demand requires more resources than the (be, eb) demand. Node (a) is less demanding than node (e) and, consequently, $a \preceq e$. $E \preceq A$ as E offers less resources than A. In fact, both nodes offer the same amount of CPU, but, as it will be seen, $BW(E) \preceq BW(A)$ according to the voting criterion, which can be observed in Tables 10 and 11. Then:

- according to the $BW_{in}(n)$: $BW(E) \preceq BW(A)$;
- according to the $BW_{out}(n)$: $BW(E) \preceq BW(A)$;
- according to the $BW_{total}(n)$: $BW(E) \preceq BW(A)$.
- Voting decision: $E \preceq A$.

Proceeding in the same way as before:

**Table 12**
Resources consumed after the VNR #2 mapping.

| Pair(s) of nodes | Node | CPU consumed | From node | BW consumed | From link |
|---|---|---|---|---|---|
| (ae, ea) | (a) | 50 | E | 90 | E-C |
| | (e) | 80 | A | 90 | C-A |
| | Hidden hop | 20 | C | 90 | A-C |
| | | | 90 | C-E | |
| (eb, be) | (b) | 60 | B | 20 | A-B |
| | | | 20 | B-A | |

- Node mapping: **a** →**E**; **e** →**A**;
- Link mapping: (**a**, **e**)→(**E**-**C**-**A**); (**e**, **a**)→(**A**-**C**-**E**).

Virtual nodes (b) and (e) have to be mapped to the highest ranked pair of substrate nodes where one of the nodes is A (e was already mapped to A). The second pair shown in Table 8, (AB, BA) satisfies the condition and the result of the node and link mapping is:

- Node mapping: **b** →**B**;
- Link mapping: (**e**, **b**)→(**A**-**B**); (**b**, **e**)→(**B**-**A**).

The resources consumed for the mapping of VNR #2 can be seen in Table 12.

It is important to check if the mappings have been successful by observing the available resources of the SN after the last VNR has been assigned. In this example, neither any of the nodes have finished with negative CPU resources nor any of the links with negative BW resources. The mapping has been completely successful.

### 3.2. Improvements introduced to the new paths algebra algorithm

Despite the good result obtained in the example scenario described in Section 3.1, the New Paths Algebra algorithm has not always produced the expected results. Many times, even for small examples, the algorithm was able to map a first VNR and to be without resources to map a second VNR. In these cases, it was easy to verify that a different mapping of the first VNR would leave enough resources for the second one. For this reason, a series of improvements were developed and they are described in this section.

#### 3.2.1. Degree penalty

Let $d_{\text{in}}$ and $d_{\text{out}}$ represent the in-degree and out-degree, respectively, of an SN node. The idea is to introduce a penalty for those nodes with $d_{\text{in}} \neq d_{\text{out}}$ because they can easily produce bottlenecks. The penalty should be larger as the difference increases. Given a substrate network with $n$ nodes, let $i$, $1 \le i \le n$ represent a SN node. The degree penalty $P_i$ is given by:

$$P_i = 1 - \left[ (1 - P_{min}) \times \frac{d_i}{d_{\max}} \right], \tag{6}$$

where

- $d_i = |d_{\text{in}}(i) - d_{\text{out}}(i)|$, $1 \le i \le n$,
- $d_{\max} = \max(d_i)$,
- $P_{min}$ $(0 < P_{min} < 1)$ is the penalty to be applied to the nodes with $d_i = d_{\max}$.

The penalty to be applied to a pair of nodes $(s, d)$ is finally given by:

$$P_d(s, d) = P_s \times P_d. \quad (7)$$

The hardest part of this modification is the choice of a suitable $P_{min}$ value. It has to be a value that penalizes the difficult nodes just enough to achieve good values for both Cost/Revenue and acceptance ratio. Although each scenario may require an optimal value of $P_{min}$ it was observed that in most of the cases $P_{min} = 0.5$ and/or $P_{min} = 0.4$ produce good results.

### 3.2.2. Last ranked pair

The new idea consists of selecting the last ranked pair of nodes with enough available resources to assign a pair of nodes of a virtual network to a pair of nodes of the substrate network, instead of selecting the pair of nodes with the most available resources. The rationale behind this idea is to use the minimum required resources in order to save resources for the next mappings.

To know if a pair of SN nodes has enough available resources to host the virtual nodes, the concept of availability is used: the value of required availability of the pair of nodes of the VNR $(VNR)req_{av}(v_i, v_j)$ and the availability of the pair of SN nodes $A(s, d)$ are compared. A mapping is possible if $(VNR)req_{av}(v_i, v_j) \le A(s, d)$.

### 3.2.3. Two-way availability

Eq. (4) is the equivalent of the one-way availability of a transmission channel. However, in general, a VNR demand consists of asking for resources in both directions of an SN link. To have a measure of resources in both directions it is necessary to use the concept of two-way availability, to indicate the probability of having enough resources from source to destination and from destination to source. Considering that a resource in one direction is independent from the resource in the other direction, the two-way availability is given by Equation (8):

$$A_{(s,d),(d,s)} = A_{(s,d)} \times A_{(d,s)}. \quad (8)$$

### 3.2.4. BW penalty

The idea is to introduce a penalty for the nodes having lower outbound than inbound BW because they may not be able to forward all the traffic they receive. This penalty is given by:

$$P_{BW} = \begin{cases} \dfrac{\max(BW_{out})}{\max(BW_{in})} & \text{if } \dfrac{\max(BW_{out})}{\max(BW_{in})} \le 1, \\ 1 & \text{if } \dfrac{\max(BW_{out})}{\max(BW_{in})} > 1. \end{cases} \quad (9)$$

### 3.2.5. Single path availability

So far the availability of a pair of nodes has been evaluated by Eq. (3). This equation is a good measure of availability in the case of multi-path routing for which the total BW demand can be split among several paths. For the single path routing case the pair of nodes should be ranked based on their minimum availability as given by Eq. (10):

$$(A_{min})_{(s,d)} = \min\{a_{(s,d)}(k), \quad k \in K\} \quad (10)$$

### 3.2.6. Ranking evaluation

Taking into account all the proposed improvements the final ranking is obtained applying Eq. (11):

$$R_{(s,d)(d,s)} = (A_{min})_{(s,d)} \times (A_{min})_{(d,s)} \times P_d(s, d) \times P_{BW}(s) \times P_{BW}(d). \quad (11)$$

In Eq. (11) the factors $(A_{min})_{(s,d)}$, $(A_{min})_{(d,s)}$, $P_{BW}(s)$ and $P_{BW}(d)$ are dynamic in the sense that they take into account the available resources in the SN which change after each successful mapping of a VNR. The factor $P_d(s, d)$ is static as it depends only on the SN topology.

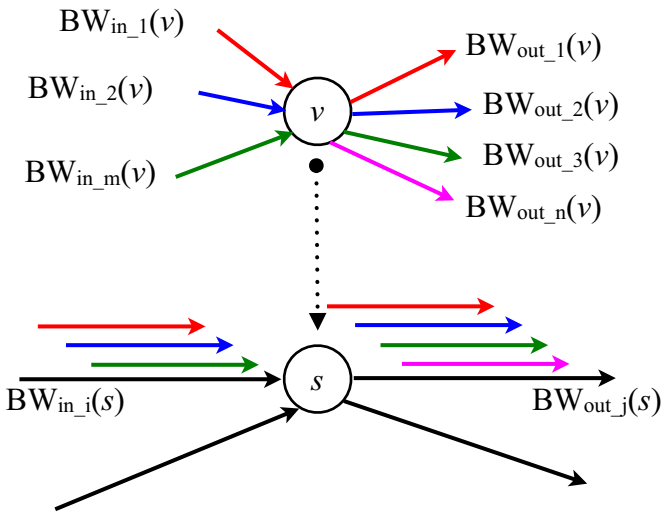


**Fig. 5.** BW checking illustration.

### 3.2.7. BW checking

The availability concept proposed in Section 3.2.2 along with Eq. (11) are useful to rank the SN nodes according to their available resources but it does not guarantee that the SN node has in fact enough resources to receive the mapping of a VNR demand. This happens because the proposed availability concept is based on the combination of CPU and BW resources, and a penalty imposed to the longer paths by the weight $x(k)$.

It may happen to have high ranked pairs of SN nodes with low BW resources but high CPU resources. This distortion is enhanced after the first VNR mapping when low connected nodes have their BW resources used and continue to be in a high ranked position.

To avoid the selection of high ranked pair of nodes without enough available resources it has been introduced a specific check of BW: the total required BW of the VNR demand must be less than both the maximum outbound and inbound BW of the selected SN node.

Fig. 5 shows a virtual node $v$ with VNR BW input and output demands designated respectively as $BW_{in\_i}(v)$, $1 \le i \le m$ and $BW_{out\_j}(v)$, $1 \le j \le n$. This virtual node is mapped to the SN node $s$ and all input/output virtual links are mapped to a single input / output physical link with available BW resources $BW_{in\_i}(s)$ and $BW_{out\_j}(s)$. This mapping is possible if and only if:

$$\sum_{i=1}^{m} BW_{in\_i}(v) \le BW_{in\_i}(s), \quad (12)$$

and

$$\sum_{j=1}^{n} BW_{out\_j}(v) \le BW_{out\_j}(s). \quad (13)$$

Being $v$ the node from the VNR, $s$ the node from the SN, $v_{in}/v_{out}$ the input/output links to/from node $v$, $s_{in}/s_{out}$ the input/output links to/from node $s$, and $V_{in}/V_{out}$ the total set of input/output links of node $v$ and $S_{in}/S_{out}$ the total set of input/output links of node $s$, the BW checking consists in verifying both Eqs. (14) and (15):

$$\sum_{v_{in} \in V_{in}} BW_{in\_v_{in}}(v) < \max\{BW_{in\_s_{in}}(s), s_{in} \in S_{in}\}, \quad (14)$$

$$\sum_{v_{out} \in V_{out}} BW_{out\_v_{out}}(v) < \max\{BW_{out\_s_{out}}(s), s_{out} \in S_{out}\}. \quad (15)$$

## 4. Evaluation scenarios and experimental results

This section describes the network topology generation, simulations scenarios and presents the experimental results.

### 4.1. Network topology generation

The following network generators were studied and the characteristics of the produced topologies were compared:

- Barabási–Albert: this model generates a random scale-free network, whose degree distribution follows a power law. It incorporates two important general concepts: growth (the number of nodes in the network increases over time) and preferential attachment (the more connected a node is, the more likely it is to receive new links).
- Erdős–Rényi: in this model every possible edge is created with the same constant probability $p$.
- Random regular: this network topology generator creates a random graph where each vertex has the same degree $k$.
- Scale-free: this model generates a scale-free, non-growing random graph with an expected power-law degree distribution.
- Watts–Strogatz: is a random graph generation model that produces graphs with small-world properties, including short average path lengths and high clustering.
- Waxman: this random network topology generator is a geographic model where the nodes are uniformly distributed in the plane and edges are added according to probabilities that depend on the distances between the nodes.

Table 13 summarizes the characteristics of the studied topologies.

The Waxman generator is one of the most used topology generators in the VNE literature (Yu et al., 2008; Lischka and Karl, 2009; Chowdhury et al., 2009, 2012; Cheng et al., 2011; Till Beck et al., 2013; Botero et al., 2013a; Beck et al., 2015). Therefore, to implement a comparable VNE approach and taking advantage of its availability in the ALEVIN framework (Fischer et al., 2011) and our previous experience with it Botero et al. (2013a), Waxman topology generator was chosen to be used in this work.

The algorithm presented in this paper will be tested with the remaining studied topologies in future work.

### 4.2. Simulation environment and scenarios

To deal with the computational complexity of the paths algebra-based multi-constraint routing algorithm, we use the filtering pre-processing strategy SEARCHPATH and SORTPATH that comes from our previous work (Botero et al., 2013a).

For the comparison of former VNE algorithms (Botero et al., 2013a) with the paths algebra algorithm, the following methodology is used (this methodology has been previously used and described by the authors in Botero et al., 2012, 2013a; Botero and Hesselbach, 2013). The scenarios are parameterized by:

- topology creation;
- resource and demand deployment;
- simulation conditions.

**Table 13**
Comparison between the studied topologies.

| Topology | Degree distribution | Growth/ pref. attachment | Average path length | Clustering coefficient | Small-world network |
|---|---|---|---|---|---|
| Barabási–Albert | Power law | Yes | Short | Low | No |
| Erdős–Rényi | Binomial | No | Short | Medium | No |
| Random regular | Regular | No | Large | High | No |
| Scale-Free | Power Law | No | Medium | Low | No |
| Watts–Strogatz | Dirac Delta | No | Short | High | Yes |
| Waxman | Geometric | No | Medium | Medium | No |

#### 4.2.1. Topology creation

For creating both the substrate networks and the virtual networks as well, we use the Waxman algorithm. To that end, we uniformly distribute the coordinates of the nodes in an area. The Waxman generator takes two parameters, $\alpha$ and $\beta$ that determine the probability of an edge connecting two nodes, as given by Eq. (16):

$$P(u, v) = \alpha e^{-\frac{d(u,v)}{\beta \times L}}, \tag{16}$$

in which $0 < \alpha, \beta \leq 1$, $d$ is the Euclidean distance between nodes $u$ and $v$, and $L$ is the maximum Euclidean distance between any two nodes. An increase in the parameter $\alpha$ increases the probability of edges between any nodes in the graph. An increase in the parameter $\beta$ yields a larger ratio of long edges to short edges.

This topology generation procedure is used for the substrate network as well as for all $k^{\max}$ virtual networks.

#### 4.2.2. Resource and demand deployment

A load targeted resource and demand deployment is available in ALEVIN and consists of two steps:

1. The substrate network will be equipped with resources by uniformly distributing each node resource **X** in a given interval $(0, \mathrm{NR}_X^{\max})$ for every substrate node (the interval for CPU resources is (0,100]) as well as each link resource **Y** in a given interval $(0, \mathrm{NR}_Y^{\max})$ for every substrate link (the interval for BW resources is (0,100]), as usually done in literature (Yu et al., 2008; Lischka and Karl, 2009; Chowdhury et al., 2009, 2012).
2. Consider the generation of demands in all virtual networks and the goal is to achieve a certain average load of every resource. The creation of node demands and link demands that fulfill these load requirements comprises different challenges. As the number of nodes is fixed in the Waxman topology generation, we can easily calculate the mean resource on a substrate node as

$$E[\mathrm{NR}_X] = \frac{0 + \mathrm{NR}_X^{\max}}{2} = \frac{\mathrm{NR}_X^{\max}}{2} \tag{17}$$

as well as the mean demand on a virtual node for a given overall load $\rho$ as

$$E[\mathrm{ND}_X] = \rho \cdot \mathrm{E}[\mathrm{NR}_X] \cdot \frac{|V|}{|V^k| \cdot k^{\max}}, \tag{18}$$

in which
- $|V|$ = number of substrate nodes;
- $k$ = number of virtual networks; and
- $|V^k|$ = number of virtual nodes per virtual network.

Eq. (17) results in a maximum resources demand

$$\mathrm{ND}_X^{\max} = 2 \cdot \mathrm{E}[\mathrm{ND}_X] \tag{19}$$

due to the uniform distribution of demands within $(0, \mathrm{ND}_X^{\max}]$.

As the Waxman topology generation is probabilistic regarding link creation, the number of links in a Waxman-network is not fixed but can only be given by probabilities. To achieve the targeted load $\rho$ of link resource as well, we consider the average number of edges in a Waxman-network by estimating the mean probability $E[p]$ of creating an edge between any two nodes. Thus, the average number of edges $E[|A|]$ in a directed graph is given by

$$E[|A|] = E[p]\cdot|V|\cdot(|V| - 1). \tag{20}$$

Therefore, the average link resource in a network is calculated by

$$E[\mathrm{LR}_Y] = \frac{0 + \mathrm{LR}_Y^{\max}}{2} = \frac{\mathrm{LR}_Y^{\max}}{2} \tag{21}$$

and the average link demand is given by

$$E[\mathrm{LD}_Y] = \rho\cdot \mathrm{E}[\mathrm{LR}_Y]\cdot\frac{\mathrm{E}[|A|]}{\mathrm{E}[|A^k|]\cdot k^{\max}} \tag{22}$$

which results in

$$\mathrm{LD}_Y^{\max} = 2\cdot \mathrm{E}[\mathrm{LD}_Y]. \tag{23}$$

However, it is necessary to ensure that $\mathrm{LD}_Y^{\max} \le \mathrm{LR}_Y^{\max}$ holds. In particular, it is necessary to ensure that $E[|A|] < E[|A^k|]\cdot k^{\max}$ in Eq. (22) holds for $0 \le \rho \le 1$. To that end, we have to enforce the constraint

$$|V|^2 < k^{\max}.|V^k|^2. \tag{24}$$

If the constraint in Eq. (24) is violated, the approximation provided above will achieve a higher load value than the targeted load $\rho$ for the link resources and even result in $\mathrm{LD}_Y^{\max} > \mathrm{LY}_Y^{\max}$ which can never be fulfilled.

#### 4.2.3. Scenarios

As described in Section 4.2.1, we use the Waxman topology generation. For evaluation, we used $\alpha = \beta = 0.5$ and distributed the coordinates of the nodes uniformly in an $1 \times 1$ square area. Empirical studies for these parameters have provided an average distance of any two nodes $E[d] \approx 0.5$ and a maximum distance of $L = \sqrt{2}$. Thus, according to Eq. ((16) the average probability for creating a link between any two nodes is $E[p] \approx 1/4$.

In this work, we consider CPU cycles as a node resource, denoted by $\mathrm{NR}_{\mathrm{CPU}}$, and bandwidth as a link resource, denoted by $\mathrm{LR}_{\mathrm{BW}}$ in the substrate network. For the uniform distribution of these values we have chosen the maxima $\mathrm{NR}_{\max} = 100$ and $\mathrm{NR}_{\max} = 100$.

The scenarios were generated using the ALEVIN environment. All of them consist of a substrate network (SN) with 20 nodes and 10 virtual networks (VN) with 10 nodes each. The substrate metrics BW and CPU are different in each scenario and the resources demands in the VNRs change according to the desired load $\rho$. Table 14 summarizes the simulation scenarios parameters.

The results are the average values obtained for 5 simulations runs for each load $\rho$.

The cost (**C**) is calculated adding the resources consumed for the mapping of the VNRs, while the revenue (**R**) is the sum of the required resources of the mapped VNRs. Consider a VNR with $v_n$ nodes and $v_l$ links that is mapped to the SN using $s_n$ nodes and $s_l$ links. In the optimal case, i.e, all mapped virtual nodes are connected by direct links, $v_n = s_n$ and $v_l = s_l$. In the general case, in which the virtual links are mapped onto substrate paths that traverse hidden nodes, $v_n \le s_n$ and $v_l \le s_l$. Eqs. (25) and (26) are used to evaluate respectively the cost and revenue of a successful mapping.

**Table 14**
Parameters chosen for the simulation scenarios.

| Parameter description | Chosen values |
|---|---|
| Number of substrate nodes ($V$) | 20 |
| Number of VNRs ($k$) | 10 |
| Number of virtual nodes per virtual network ($V^k$) | 10 |
| Set of loads ($\rho$) | {0.2, 0.3, 0.4, 0.5, 0.6, 0.7} |

$$C = \sum_{i=1}^{s_l} BW(i) + \sum_{i=1}^{s_n} CPU(i) \tag{25}$$

$$R = \sum_{j=1}^{v_l} BW(j) + \sum_{j=1}^{v_n} CPU(j) \tag{26}$$

The VNRs were processed according to the most consuming first (MCF) ordering, while the metrics used by the paths algebra to order the paths was **M**={Hops, BW, CPU}. The chosen metrics optimizes the cost of the mapping.

### 4.3. Experimental results

To evaluate the performance of the proposed algorithms, compared to previous published works, for each scenario the following simulations were done:

- Original paths algebra (OPA): refers to the original paths algebra-based VNE algorithm (Botero et al., 2013a) that performs the mapping in two uncoordinated stages—node mapping followed by link mapping.
- New paths algebra (NPA): refers to the proposed algorithm in this paper that performs the mapping in two coordinated stages in which the rank is evaluated by Eq. (4).
- Improved new paths algebra (I-NPA): refers to the proposed algorithm in this paper that performs the mapping in two coordinated stages in which the rank is evaluated by Eq. (11) followed by the BW checking.

In the OPA case, the ALEVIN environment is used to generate topology and VNRs, and node mapping. The paths algebra-based link mapping stage was implemented in MATLAB.

In the NPA and I-NPA cases, the ALEVIN environment is used to generate topology and VNRs, while the coordinated nodes and links mappings were implemented in MATLAB.

The algorithms are evaluated by means of the following metrics:

- Cost/revenue.
- Accepted VNR percentage, considering a VNR as a whole unit.
- Mapped revenue ratio, that is the ratio between the actual revenue provided by the successful mappings and the total possible revenue associated to the VNRs.

Table 15 shows the results obtained by the OPA algorithm. These results are taken as the reference to compare with the NPA and I-NPA algorithms.

Table 16 shows the results obtained with the NPA, while Figs. 6 and 7 compare the accepted VNR percentage and the Cost/Revenue obtained with the OPA and NPA algorithms. Figs. 8 and 9 compare the Cost and Revenue results for the same algorithms.

Figs. 6–9 show that the OPA and NPA algorithms have similar

**Table 15**
Results obtained with the OPA algorithm.

| OPA | | | | | |
|---|---|---|---|---|---|
| Load ($\rho$) | Cost | Revenue | Cost/ Revenue | Accepted VNR (%) | Mapped revenue ratio (%) |
| 0.2 | 3135.0 | 2143.8 | 1.46 | 100 | 100.0 |
| 0.3 | 4047.4 | 2627.7 | 1.54 | 78 | 80.6 |
| 0.4 | 3941.4 | 2582.3 | 1.55 | 56 | 59.1 |
| 0.5 | 4198.3 | 2602.5 | 1.60 | 44 | 47.4 |
| 0.6 | 3839.3 | 2338.6 | 1.62 | 34 | 37.2 |
| 0.7 | 3516.7 | 2156.9 | 1.63 | 28 | 30.0 |

**Table 16**
Results obtained with the NPA algorithm.

| NPA | | | | |
|---|---|---|---|---|
| Load ($\rho$) | Cost | Revenue | Cost/Revenue | Accepted VNR (%) |
| 0.2 | 2975.8 | 1958.7 | 1.51 | 90.0 |
| 0.3 | 3881.5 | 2437.3 | 1.60 | 72.0 |
| 0.4 | 4092.3 | 2528.0 | 1.62 | 54.0 |
| 0.5 | 3787.9 | 2282.5 | 1.67 | 38.0 |
| 0.6 | 4324.2 | 2591.6 | 1.68 | 38.0 |

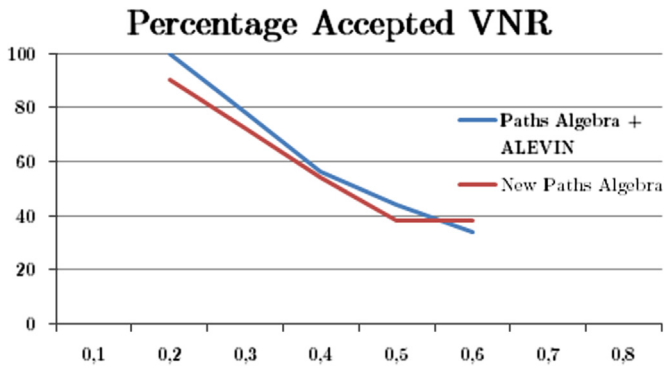


**Fig. 6.** Accepted VNR percentage: OPA *vs.* NPA.

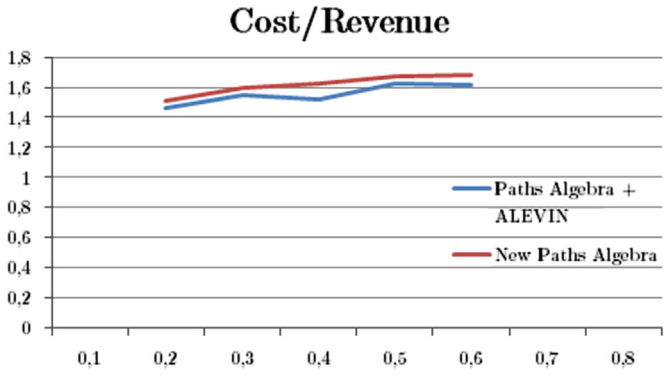


**Fig. 7.** Cost/Revenue comparison: OPA *vs.* NPA.

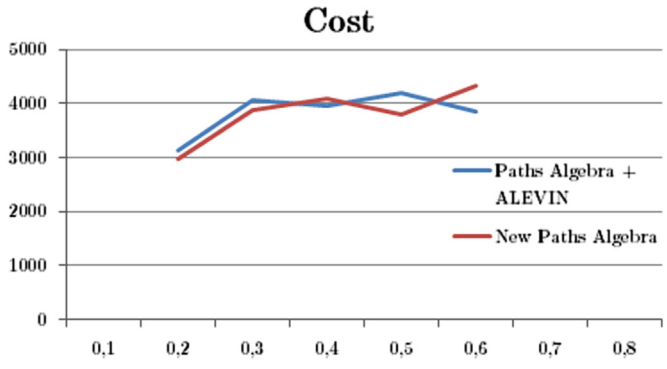


**Fig. 8.** Cost comparison: OPA *vs.* NPA.

behavior. This is an indication that the NPA algorithm was correctly implemented. However, its performance is worse than the OPA's. This is an unexpected result as the coordinated node and link mapping should provide better results. The reason for this is Eq. (4) that does not model adequately the SN available resources. In spite of not being good results they were important to guide the introduction of the improvements described in Section 3.2 that led

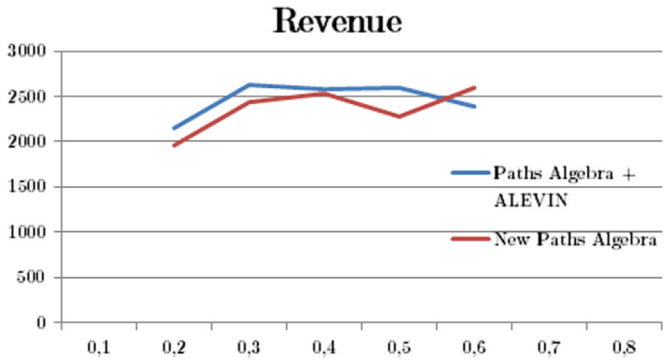


**Fig. 9.** Revenue comparison: OPA *vs.* NPA.

**Table 17**
Results obtained with the I-NPA algorithm.

| I-NPA | | | | | |
|---|---|---|---|---|---|
| Load ($\rho$) | Cost | Revenue | Cost/ revenue | Accepted VNR Perc. (%) | Mapped revenue ratio (%) |
| 0.2 | 3189.8 | 2143.8 | 1.49 | 100.0 | 100.0 |
| 0.3 | 5027.9 | 3256.6 | 1.54 | 100.0 | 100.0 |
| 0.4 | 5605.3 | 3652.3 | 1.53 | 82.0 | 83.4 |
| 0.5 | 5386.0 | 3422.3 | 1.60 | 60.0 | 62.3 |
| 0.6 | 4825.5 | 3016.4 | 1.61 | 44.0 | 47.1 |
| 0.7 | 4058.6 | 2431.7 | 1.67 | 30.0 | 33.8 |

to the development of the I-NPA algorithm.

Table 17 shows the results obtained with the I-NPA algorithm while Fig. 10 compares the accepted VNR percentage between the OPA and the I-NPA algorithms for all simulation runs. The results shown in Fig. 10 are quite impressive: the I-NPA performance is never worse than the OPA's and produced up to 60% enhancement of the accepted VNR percentage.

Fig. 11 compares the mapped revenue ratio obtained by the OPA and I-NPA algorithms. The I-NPA performance is much better than the OPA's for all workloads. Fig. 12 shows that the Cost/Revenue relationship is quite similar for both algorithms. As network virtualization is essentially a technique to implement a business model, the results provided by the I-NPA algorithm can be directly translated into an increase of the service provider's profit as a better mapped revenue ratio is obtained at a similar Cost/Revenue value.

Fig. 13 shows the mapping cost for the OPA and I-NPA algorithms. When the load is low both algorithms give the same result; when the load is in the mid-range the I-NPA cost is much higher than the OPA's; and when the load is high the costs are closer again. Fig. 14 shows the mapping revenue for the OPA and I-NPA algorithms and the observed behaviors are similar to the cost behavior. When the load is low the SN has enough resources to accommodate the mapping and both algorithms behave in a similar way looking for the shortest paths. When the load increases the OPA tends to revisit the same nodes and map the links to the possible shortest paths. The I-NPA penalizes the nodes that have already been used and distribute the traffic more evenly across the whole SN. This implies that longer paths are selected making the cost to increase but the VNR acceptance percentage also increases producing a higher revenue. When the load is high there are not enough resources to accommodate many VNRs and the revenue drops. Those that can be mapped can use shorter paths decreasing the cost for both algorithms.

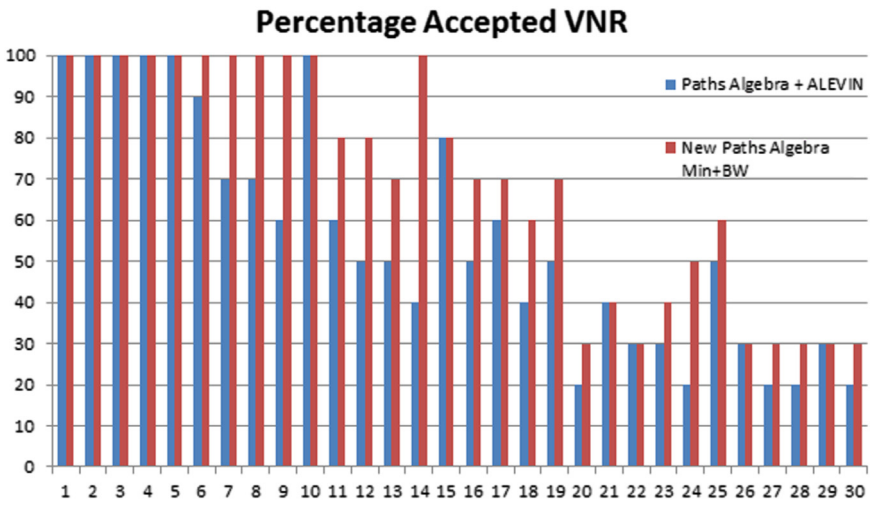


**Fig. 10.** Accepted VNR percentage: OPA *vs.* I-NPA for all simulation runs.

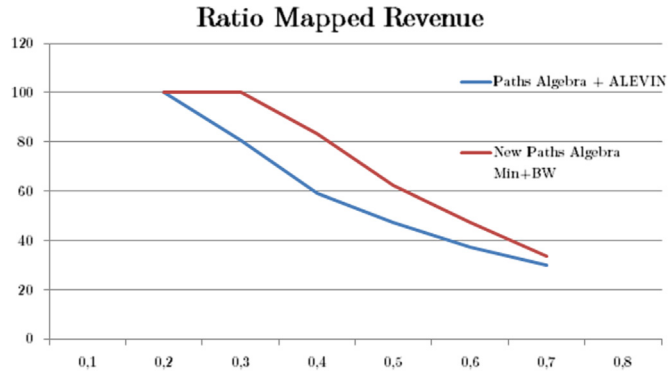


**Fig. 11.** Mapped revenue ratio: OPA *vs.* I-NPA.

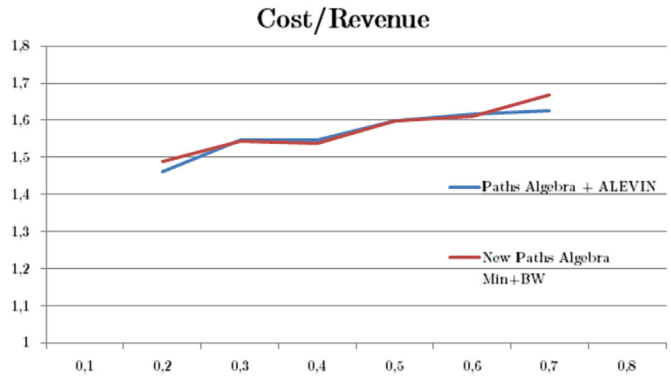


**Fig. 12.** Cost/Revenue: OPA *vs.* I-NPA.

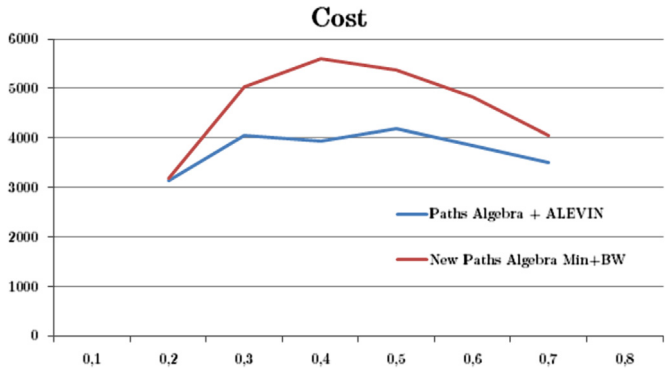


**Fig. 13.** Cost: OPA *vs.* I-NPA.

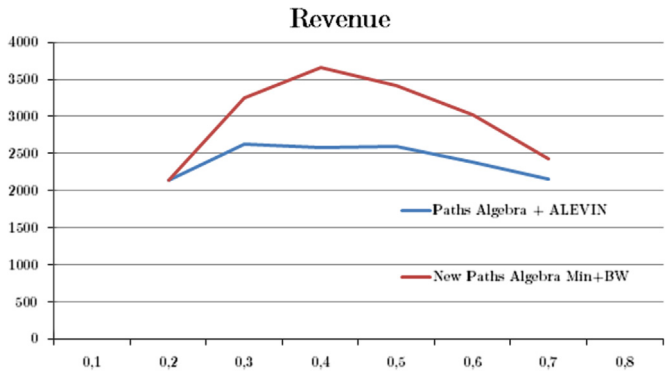


**Fig. 14.** Revenue: OPA *vs.* I-NPA.

### *4.4. System impact*

The numerical results presented in Section 4.3 show that when the I-NPA is compared to the OPA the results are impressive. The average VNR acceptance percentage increased for all loads except for $\rho = 0.2$ for which both algorithms map all VNRs. The maximum improvement is of 26% for $\rho = 0.4$. For high loads, $\rho = 0.7$, the I-NPA is still slightly better showing an improvement of 2%. The mapped revenue ratio was also increased by the I-NPA showing an average improvement of 14.42% for a maximum of 24.27%. Last but not least, the OPA and the I-NPA provide similar Cost/Revenue values. This means that the increased VNR acceptance percentage of the I-NPA can translate directly into system performance and higher profits for the service provider.

The I-NPA can be used to implement differentiated services. The services provider's clients could be offered, for example, three different subscriptions: gold, silver and bronze. The better the subscription is, the SP could offer more resources charging a higher price. The target price can be easily incorporated as a metrics into the paths algebra framework and used along the other metrics to solve the VNE problem.

## 5. Conclusions and future work

The advantages of the coordinated link and node mapping were analyzed, defining an optimization strategy based in the paths algebra mathematical framework including linear and non-

linear parameters. Two new algorithms were proposed to solve the VNE problem in a coordinated way of the node and link mapping stages, as an enhancement from previous already published works. Both algorithms are also based on the paths algebra, originally developed to implement multi-constraint routing algorithms and that were successfully applied by the authors to solve the VNE problem in an uncoordinated way. This algorithm is designated as Original Paths Algebra—OPA. A metric to evaluate the resources available in the SN and rank its nodes was proposed Such a metric borrows the concept of availability of a transmission channel.

The first algorithm, designated as new paths algebra (NPA), was explained in detail and applied to a small scenario example providing good results. However, when applied to larger scenarios the results were initially quite disappointing but indicated the improvements to be introduced.

This second algorithm was designated improved new paths algebra (I-NPA). The I-NPA was compared to the OPA and the results are quite impressive.

As future works we intend to change the ranking equation. The current equation has factors that depend on the available resources and change after each successful mapping. It has only one static factor that depends on the topology and takes into account the input/output nodes' degrees. We conjecture that the results can be further improved if a more thorough characterization of the topology is considered. The closeness metrics measures the proximity of the nodes to each other, while the betweenness evaluates which nodes are more traversed by paths connecting pair of nodes. These metrics can be used to identify nodes more appropriate to be hubs or intermediate and which ones are more adequate to be terminals and host virtual nodes.

## Acknowledgments

This work has partially been supported by the Spanish Government under project TEC2013-47960-C4-1-P, by the Catalan government under grant 2014 SGR 1375, by the CODI project 2014-856 and the CODI sustainability strategy 2014-2015 of the University of Antioquia (Colombia).

## References

Beck, M.T., Fischer, A., Botero, J.F., Linnhoff-Popien, C., de Meer, H., 2015. Distributed and scalable embedding of virtual networks. J. Netw. Comput. Appl. 56, 124–136. http://dx.doi.org/10.1016/j.jnca.2015.06.012 ⟨http://www.sciencedirect.com/science/article/pii/S1084804515001393⟩.

Bhardwaj, S., Jain, L., Jain, S., 2010. Cloud computing: a study of infrastructure as a service (iaas). Int. J. Eng. Inf. Technol. 2. 60-63.

Botero, J., Hesselbach, X., 2009. The bottlenecked virtual network problem in bandwidth allocation for network virtualization. In: Communications, 2009. IEEE Latin-American Conference on LATINCOM '09. pp. 1–5. http://dx.doi.org/10.1109/LATINCOM.2009.5305042.

Botero, J.F., Hesselbach, X., 2013. Greener networking in a network virtualization environment. Comput. Netw. 57 (9), 2021–2039. http://dx.doi.org/10.1016/j.comnet.2013.04.004.

Botero, J., Hesselbach, X., Duelli, M., Schlosser, D., Fischer, A., de Meer, H., 2012. Energy efficient virtual network embedding. IEEE Commun. Lett. 16 (5), 756–759.

Botero, J.F., Molina, M., Hesselbach-Serra, X., Amazonas, J.R., 2013a. A novel paths algebra-based strategy to flexibly solve the link mapping stage of vne problems. J. Netw. Comput. Appl. 36 (6), 1735–1752.

Botero, J.F., Hesselbach, X., Fischer, A., De Meer, H., 2013b. Optimal mapping of virtual networks with hidden hops. Telecommun. Syst. 52 (3), 1–10.

Botero Vega, J.F., 2013. Study, evaluation and contributions to new algorithms for the embedding problem in a network virtualization environment (Ph.D. thesis), Universitat Politècnica de Catalunya, ⟨http://www.tdr.cesca.es/handle/10803/120754⟩.

Cheng, X., Su, S., Zhang, Z., Wang, H., Yang, F., Luo, Y., Wang, J., 2011. Virtual network embedding through topology-aware node ranking. SIGCOMM Comput. Commun. Rev. 41, 38–47.

Cheng, X., Su, S., Zhang, Z., Shuang, K., Yang, F., Luo, Y., Wang, J., 2012. Virtual network embedding through topology awareness and optimization. Comput. Netw. 56 (6), 1797–1813. http://dx.doi.org/10.1016/j.comnet.2012.01.022.

Chowdhury, N.M.M.K., 2009. Network virtualization: state of the art and research challenges. IEEE Commun. Mag. 47 (7), 20–26.

Chowdhury, N.M.K., Boutaba, R., 2010. A survey of network virtualization. Comput. Netw. 54 (5), 862–876.

Chowdhury, N., Rahman, M., Boutaba, R., 2009. Virtual network embedding with coordinated node and link mapping. In: INFOCOM 2009, IEEE, Rio de Janeiro, Brazil, pp. 783–791.

Chowdhury, M., Rahman, M., Boutaba, R., 2012. Vineyard: virtual network embedding algorithms with coordinated node and link mapping. IEEE/ACM Trans. Netw. 20 (1), 206–219.

de Paula Herman, W., de Almeida Amazonas, J., 2007. Hop-by-hop routing convergence analysis based on paths algebra. In: Electronics, Robotics and Automotive Mechanics Conference, 2007. CERMA 2007, pp. 9–14. http://dx.doi.org/10.1109/CERMA.2007.4367653.

Dietrich, D., Rizk, A., Papadimitriou, P., 2015. Multi-provider virtual network embedding with limited information disclosure. IEEE Trans. Netw. Service Manag. (99), 1–14. http://dx.doi.org/10.1109/TNSM.2015.2417652.

Fajjari, I., Aitsaadi, N., Pujolle, G., Zimmermann, H., 2011. Vne-ac: Virtual network embedding algorithm based on ant colony metaheuristic. In: 2011 IEEE International Conference on Communications (ICC), pp. 1 –6.

Feamster, N., Gao, L., Rexford, J., 2007. How to lease the internet in your spare time. SIGCOMM Comput. Commun. Rev. 37 (1), 61–64.

Fischer, A., Botero, J.F., Duelli, M., Schlosser, D., Hesselbach, X., Meer, H.D., 2011. Alevin - a framework to develop, compare, and analyze virtual network embedding algorithms. In: Electronic Communications of the EASST 37, pp. 1–12.

Fischer, A., Botero, J., Till Beck, M., de Meer, H., Hesselbach, X., 2013. Virtual network embedding: a survey. IEEE Commun. Surveys Tutorials 15 (4), 1888–1906. http://dx.doi.org/10.1109/SURV.2013.013013.00155.

Guan, X., Choi, B.-Y., Song, S., 2015. Energy efficient virtual network embedding for green data centers using data center topology and future migration. Comput. Commun. http://dx.doi.org/10.1016/j.comcom.2015.05.003.

Haider, A., Potter, R., Nakao, A., 2009. Challenges in resource allocation in network virtualization In: 20th ITC Specialist Seminar, vol. 18, p. 20.

Houidi, I., Louati, W., Ameur, W.B., Zeghlache, D., 2011. Virtual network provisioning across multiple substrate networks. Comput. Netw. 55 (4), 1011–1023.

Houidi, I., Louati, W., Zeghlache, D., 2015. Exact multi-objective virtual network embedding in cloud environments. Comput. J. http://dx.doi.org/10.1093/comjnl/bxu154.

Jian, D., Tao, H., Jian, W., Wenbo, H., Jiang, L., Yunjie, L., 2015. Virtual network embedding through node connectivity. J. China Univ. Posts Telecommun. 22 (1), 17–56. http://dx.doi.org/10.1016/S1005-8885(15)60620-3.

Lischka, J., Karl, H., 2009. A virtual network mapping algorithm based on subgraph isomorphism detection. In: VISA '09: Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures, pp. 81–88.

Mechtri, M., Hadji, M., Zeghlache, D., 2015. Exact and heuristic resource mapping algorithms for distributed and hybrid clouds. IEEE Trans. Cloud Comput. PP (99), 1–14. http://dx.doi.org/10.1109/TCC.2015.2427192.

Shanbhag, S., Kandoor, A.R., Wang, C., Mettu, R., Wolf, T., 2015. Vhub: Single-stage virtual network mapping through hub location. Comput. Netw. 77, 169–180. http://dx.doi.org/10.1016/j.comnet.2014.12.006.

Su, S., Zhang, Z., Liu, A., Cheng, X., Wang, Y., Zhao, X., 2014. Energy-aware virtual network embedding. IEEE/ACM Trans. Netw. 22 (5), 1607–1620. http://dx.doi.org/10.1109/TNET.2013.2286156.

Till Beck, M., Fischer, A., de Meer, H., Botero, J., Hesselbach, X., 2013. A distributed, parallel, and generic virtual network embedding framework In: 2013 IEEE International Conference on Communications (ICC), pp. 3471–3475. http://dx.doi.org/10.1109/ICC.2013.6655087.

Xiao, A., Wang, Y., Meng, L., Qiu, X., Li, W., 2014. Topology-aware virtual network embedding to survive multiple node failures In: Global Communications Conference (GLOBECOM), 2014 IEEE, pp. 1823–1828. http://dx.doi.org/10.1109/GLOCOM.2014.7037073.

Yu, M., Yi, Y., Rexford, J., Chiang, M., 2008. Rethinking virtual network embedding: substrate support for path splitting and migration. ACM SIGCOMM CCR 38 (2), 17–29.

Zhang, Z., Cheng, X., Su, S., Wang, Y., Shuang, K., Luo, Y., 2012. A unified enhanced particle swarm optimization-based virtual network embedding algorithm. Int. J. Commun. Syst. n/a–n/a.

Zhang, S., Qian, Z., Wu, J., Lu, S., Epstein, L., 2014. Virtual network embedding with opportunistic resource sharing. IEEE Trans. Parallel Distrib. Syst. 25 (3), 816–827.

Zhang, Z., Su, S., Lin, Y., Cheng, X., Shuang, K., Xu, P., 2015. Adaptive multi-objective artificial immune system based virtual network embedding. J. Netw. Comput. Appl. 53 (3), 140–155. http://dx.doi.org/10.1016/j.jnca.2015.03.007.

Zhu, F., Wang, H., 2014. A modified ant colony optimization algorithm for virtual network embedding. J. Chem. Pharmaceut. Res. 6 (7), 327–337.