Journal of Network and Computer Applications 36 (2013) 1735–1752

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

A novel paths algebra-based strategy to flexibly solve the link mapping stage of VNE problems

Juan Felipe Botero [a],*, Miguel Molina [b], Xavier Hesselbach-Serra [a], José Roberto Amazonas [b]

[a] Universitat Politècnica de Catalunya, Jordi Girona Street, 1 and3, Barcelona, Spain
[b] Escola Politècnica of the University of São Paulo, SP, Brazil

ARTICLE INFO

Article history:
Received 28 August 2012
Received in revised form
10 January 2013
Accepted 18 February 2013
Available online 5 March 2013

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

ABSTRACT

One of the main challenges of network virtualization is the virtual network embedding problem (VNE) that consists of mapping virtual network demands in physical network resources. VNE can be decomposed into two stages: virtual node and virtual link mapping. In the first stage, each virtual node is mapped to a suitable node in the physical network whereas the second stage is in charge of mapping the links connecting virtual nodes to paths in the physical network that suit the virtual network demands.

In this paper we propose the utilization of a mathematical multi-constraint routing framework called “paths algebra” to solve the virtual link mapping stage. Paths algebra provides the flexibility to introduce an unlimited number of linear and non-linear constraints and metrics to the problem while finding all the eligible paths in the physical network to perform the virtual link mapping resulting in better and more flexible embeddings.

1. Introduction and problem

The deployment of new Internet services is nowadays being more and more difficult, the lack of cooperation among stakeholders does not allow radical changes to the Internet architecture (Papadimitriou et al., 2009; Anderson et al., 2005; Tutschku et al., 2009).

Network virtualization has been proposed as the building block for the future internet architecture (Feamster et al., 2007; Chowdhury and Boutaba, 2010). It allows multiple heterogeneous networks to cohabit on a shared substrate network (SN).[1] One of the main recognized challenges of network virtualization will be the efficient allocation of virtual network elements on top of SN elements, this problem is commonly known as the *virtual network embedding/mapping* (VNE) problem. It can be solved in two different stages: node and link mapping. In this paper, we tackle the virtual link mapping stage that can be seen as multiple multi-constraint routing problems.

Several implementation ways of QoS (Quality of Service) control (either by the network, directly by the application or via a mixed solution (Cui et al., 2003; Furini and Towsley, 2001; Jukan and Franzl, 2004; Quoitin et al., 2003; Su and Gellman, 2004; Xiao, 2000)) and the singular nature of each QoS parameters (availability, distance, flow, etc.) allow several network routing solutions to be proposed such as exact and approximated routing algorithms, algorithms based on backward–forward heuristics, linear composition, hybrid, random, routing computation from the origin or destination, reservation and resource allocation protocols, etc. (Bejerano et al., 2005; Fujita et al., 2001; Kuipers et al., 2002; Shen et al., 2004).

In practice however, as the multi-constraint routing is an $\mathcal{NP}$-complete problem (Mieghem and Kuipers, 2004, 2005), we verify that many of these solutions have their merits restricted to an universe whose validity is often defined either by the size of the networks or by their topology, and their intuitive portability does not work for other applications. In other words, the use of a heuristic originally designed for distance metrics such as Dijkstra does not converge with another kind of metrics such as flow (Gouda and Schneider, 2003; Lagoa et al., 2004; Miyamura et al., 2003; Pei et al., 2004; Sobrinho, 2003; Wattenhofer and Venkatachary, 2001).

Analyzing this problem under the perspective of protocols design, it has been necessary to conceive a heuristic or an algorithm to ensure the routing convergence for different types of QoS metrics or QoS metrics composition, in which this problem could be addressed from an integrated and generic manner by means of a mathematical framework which allows validating the proposed solutions independently from network topology or implementation details (Jaggard and Ramachandran, 2005).

* Corresponding author. Tel.: +34 57663501824.
E-mail addresses: jfbotero@entel.upc.edu, juanxfelipe@gmail.com (J.F. Botero), miguel@lcs.poli.usp.br (M. Molina), xavierh@entel.upc.edu (X. Hesselbach-Serra), jra@lcs.poli.usp.br (J.R. Amazonas).

[1] This paper will use indifferently the terms *substrate network* and *physical network*.

1084-8045/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.jnca.2013.02.029

Therefore, besides establishing a homogeneous mathematical basis, the concepts of paths algebra used in this work (Herman, 2008; Herman and Amazonas, 2007) provide a guideline for developing a traffic engineering adaptive tool in which users can define their own path searching policy that can be closer to the existing traffic profile of their networks.

In addition, such mathematical framework allows for systematically comparing different mono-constraint and multi-constraint routing heuristics concerning their convergence guarantees, best path convergence and loop avoidance, and validating a generic and homogeneous solution that can be integrated into a single mathematical outline, flexible enough to be used in the validation or development of new routing protocols that can be used either inside of a single administrative domain (AS—Autonomous Systems) or across different ASs.

1.1. Network virtualization and virtual network embedding

The introduction of the Infrastructure as a Service (IaaS) paradigm (Bhardwaj et al., 2010) will change the current Infrastructure Service Providers (ISPs) based Internet business model. IaaS decouples the role of current ISPs into two new roles: the Infrastructure provider (InP) who owns, deploys and maintains the network infrastructure and the service provider (SP) responsible for deploying network protocols and offer end-to-end services. This new business model is well suited for the future dynamics in networking service requests. Customized services will be demanded by specific group of users, and provided by the SPs that will have to perform optimal allocations of them over the substrate network (SN) (composed of the networks owned by the set of InPs).

Network virtualization 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 (Feamster et al., 2007). Network virtualization basic element is the virtual network (VN). A VN is a combination of active elements called virtual nodes and passive elements called virtual links running on top of the SN. Virtual nodes are interconnected through virtual links, building a virtual topology.

One of the SP tasks is the generation of virtual network requests (VNRs), based on the analysis of the user service demands. Each VNR contains a set of demands of networking and non-networking parameters needed to provide the end-to-end QoE (Quality of Experience) required by the demanded service. After the VNR is created, an algorithm is executed to choose, over the heterogeneous resources provided by the SN, the optimal allocation of the virtual demands on top of the SN with regard to some predefined objective (VNE problem). This mapping, that defines the relationship of virtual network elements to their respective counterparts in the substrate network, can be divided into two stages. First, virtual node mapping, where each virtual node of a VNR is mapped to one substrate node with enough capacity to accomplish the virtual node resource demand. In second place, virtual link mapping, where each virtual link is mapped to a directed path in the substrate network with enough resource capacity to meet the virtual link demand. In this paper, we focus in the second stage, the virtual link mapping. Figure 1 shows the embedding of two VNR on top of one SN.

1.2. Formal VNE problem formulation

The formal description of the VNE problem is as follows. A SN is represented as a directed graph $G=(V,A)$ where vertices represent the SN nodes and arcs represent the SN links. On top of the SN, a set of virtual network requests – each described by its own digraph $G^k=(V^k,A^k)$ – are embedded by assigning a SN node for each virtual node and a SN set of paths for each virtual link.

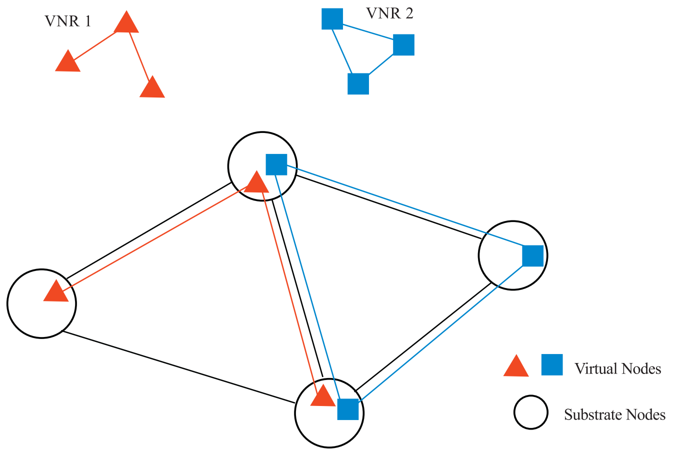

Fig. 1. Embedding of two virtual network requests.

We define a pair of functions to describe the mapping operation realized by the VNE algorithms. Node and link mapping functions. The node mapping function $X: V^k \rightarrow V$ assigns virtual nodes to substrate nodes. Likewise, the link mapping function is defined as $Y: A^k \rightarrow 2^P$ where 2^P consists of all sets of directed paths in the SN. If Y is able to assign a virtual link to a set with more than one element, the VNE problem will allow a virtual link mapping with the use of multi-path routing (i.e., one link is mapped to several SN paths). Otherwise, the result is the SN path used to allocate the virtual link. Both functions must not exceed the resources of either a node or a link in the SN. An optimal VNE is then the result of the node and link mapping functions that satisfies all the above restrictions and, additionally, reaches a given optimization objective.

To accommodate a demand between two virtual nodes inside a virtual network, in this paper, only one path is taken into account (single-path virtual link mapping). This is often a realistic restriction due to the routing protocol used, or simply because it is an explicit management requirement. However, multi-path approaches have been proposed. In Yu et al. (2008) the virtual link demand is split among the possible paths, reducing in this way the computational complexity of the problem. Although this approach is computationally better, the difficulty of its implementation in the SN is higher.

Most of the existing VNE proposals treat the single-path virtual link mapping problem as a mono-constraint problem, that is, their objective is to map the virtual link in substrate paths that minimize/maximize the usage of one resource (typically bandwidth). This paper introduces a virtual link mapping approach supporting multiple constraints thanks to the paths algebra routing framework.

1.3. Paths algebra for VNE

The virtual link mapping stage of the VNE problem may be seen as multiple multi-constraint routing problems. VNE link mapping corresponds to finding the best route(s) on the substrate network for each virtual link in the VN, where best implies the adoption of some optimization criteria.

The paths algebra is an adequate mathematical framework to explore the design space. It solves the multi-constraint problem using linear metrics as bandwidth, number of hops and delay, or non-linear metrics as availability and package loss rate. It can also use a combination of metrics as, for example, QoS taken as

$$\mathrm{QoS} = f(\mathrm{THRU}, \mathrm{PDT}, \mathrm{PDV}, \mathrm{PLR}),$$

where:

- THRU is the throughput;
- PDT is the packet delay transfer;

- PDV is the packet delay variation or jitter;
- PLR is the package loss rate.

In this case, QoS is a function of physical parameters associated to the network's performance.

However, considering that network virtualization is mainly a business strategy, the important metrics may be cost and revenue. The paths algebra may also use these metrics and, for example, associate them with the QoS.

Such flexibility associated to its computing efficiency makes the paths algebra a suitable tool to perform the virtual link mapping stage inside the VNE.

The main contribution of this paper is the introduction of a flexible approach, based on the paths algebra framework, that provides a methodology to solve the link mapping stage of the VNE. Using paths algebra, the virtual link mapping can be performed in a multi-constraint basis, that is, virtual links can be mapped to substrate paths that are characterized by an unlimited number of constraints (or combination of constraints). For instance, if the application that will run in a VN demands extremely low delay and also low packet loss rate (PLR), our approach is able to rapidly provide all the SN paths ordered in first place by delay and then by PLR, so that, the virtual link mapping can choose the best suited path for each virtual link.

1.4. Organization of the paper

After this Introduction, Section 2 reviews the methodologies reported in the literature to solve the VNE problem and identifies their limitations. Section 3 summarizes the paths algebra framework, while Section 4 introduces the novel paths algebra-based strategy to solve the VNE problem. Section 5 presents the simulation scenarios and experimental results. At last, Section 6 concludes the paper and indicates the future work.

2. Methodologies to solve the VNE problem

One of the first approaches to solve VNE proposed by Zhu and Ammar (2006), tries to balance SN *stress*. The stress of a SN element (node or link) is the number of virtual instances mapped on top of it. Two heuristic algorithms are proposed: The "Basic VN assignment algorithm" trying that looks for the minimization of the node and link stress sum, and the "Subdividing Algorithm" that splits each VNR into a set of connected sub-VNs, each with a star topology. After VN splitting, for each sub-VN, node mapping is performed by finding a cluster node and a set of SN nodes (best suited to minimize node and link stresses). The virtual link mapping is realized by connecting the already mapped virtual nodes using the shortest paths in the SN. These approaches consider just node and link stress as objective and do not take into account the constraints imposed by the resource capacities of the network elements.

The approach proposed in Lu and Turner (2006) just considers bandwidth constraints and is evaluated in an off-line scenario. In this approach, VN topologies are limited to be "backbone-star" topologies. These topologies are composed of "access nodes"; nodes representing the SN location at which traffic enters to the VN, and "backbone nodes" connecting the access nodes. An iterative algorithm changing the mapping of backbone nodes is performed to embed the different VNRs.

The approach proposed by Yu et al. (2008) looks for the maximization of a long average *revenue*, i.e., the weighted sum of VNR's bandwidth and CPU demands. CPU in nodes and BW in links are the resources considered in this proposal. For each VNR, the embedding is separated into node and link mapping stages. Node mapping follows a greedy algorithm that chooses a set of eligible SN nodes for each virtual node and then maps it to one of them based on its amount of available resources. Two different solutions are proposed for link mapping: the first consists of the allocation of each virtual link in multiple SN paths by solving the well known multi-commodity flow problem, and the second one uses a k-shortest path algorithm (single path embedding) to map each virtual link.

Chowdhury and Boutaba propose an approach for coordinated node and link mapping in Chowdhury et al. (2009, 2012). Their main goal is to minimize VN embedding cost. The embedding cost is defined as the sum of the BW and CPU spent, by the SN, to map a VNR. Geographical location for substrate and virtual nodes and a non-negative distance per VNR indicating how far a virtual node of the VNR can be of its demanded location are new constraints included in this proposal.

Node mapping stage starts with the creation of an augmented SN graph; introducing a set of meta-nodes, one per virtual node, each connected to a cluster of substrate nodes that fullfils location and capacity constraints. The proposed approach solves a Mixed Integer Programming (MIP) formulation of the VNE problem over the extended SN graph that minimizes the VNR embedding cost. As the proposed MIP is $\mathcal{NP}$-complete, its linear programming relaxation is solved, and the result is then rounded in two ways: deterministically or randomly (resulting in two different algorithms: DViNE and RViNE). Link mapping is performed following the same two solutions proposed in Yu et al. (2008).

VNE is performed in just one stage in Lischka and Karl (2009) proposal. Node and link mapping are performed in the same stage by reducing the VNE to the well known $\mathcal{NP}$-complete Subgraph Isomorphism Detection (SID) problem. In graph theory, an isomorphism of graphs G and H ($G \simeq H$) is a bijection between the vertex sets of G and H, $m : V(G) \rightarrow V(H)$, such that any two vertices i and j of G are adjacent in G if and only if $m(i)$ and $m(j)$ are adjacent in H. The $\mathcal{NP}$-complete SID problem tries to find a subgraph G_{sg} of G ($G_{sg} \subset G$) such that $G_{sg} \simeq H$. The VNE is solved by the modification of an existing SID heuristic, consisting of finding an isomorphic subgraph (representing the VNR), fulfilling the VNR demands, inside the substrate network.

Butt et al. (2010) introduced two important contributions to VNE:

The topology awareness is proposed by including the critical resources in the VNE objective function. Critical SN resources are the links and nodes contributing in the network *partition* when mapped; a link or node is *partition's* susceptible when its removal separates the SN into two different disconnected networks.

VNs are embedded with an associated lifetime. Therefore, arbitrary values of them can cause the whole SN embedding to become inefficient. Periodic reconfiguration schemes do not always work in real situations (incurred delay when remapping virtual links and nodes). The proposed approach is to act whenever a VNR gets rejected by identifying virtual links and nodes causing VNR rejection (bottlenecks) and then, relocating them to less critical regions of the SN.

In previous work (Botero et al., 2012), the concept of hidden hops was introduced. A substrate node is a hidden hop if it is part of a SN path used to map a virtual link (i.e., it is an intermediate node in the SN path). Besides the demands of some virtual nodes, each hidden hop will have a demand of resources because it has to be configured to forward packets passing through the SN path.

Fajjari et al. (2011) introduced an ant colony meta-heuristic to solve the VNE. Furthermore, they introduced topology constraints where access virtual nodes should be mapped to SN nodes of the same type (access and core). The result of this heuristic shows improvements in the rejection rate, revenue and average cost.

An optimal cost solution solving link and node mapping in the same stage was recently proposed by Houidi et al. (2011). The optimal solution is found by solving a MIP optimal problem formulation. Although optimal embedding can be performed in this way, as MIPs are complex problems ($\mathcal{NP}$-complete), this solution is not scalable and will take unaffordable running times for large networks. However, it can be used as a baseline for new VNE heuristics.

Cheng et al. (2011) proposed an approach inspired in google's PageRank algorithm. This algorithm ranks the substrate and virtual nodes based on their resources and the network topology. Based on the node ranking, two VNE algorithms are proposed: First, a classical two stage approach where the node mapping is made by assigning the nodes with higher ranking in the VN to the higher ranked SN nodes and link mapping is performed following the algorithms of Yu et al. (2008). The second approach is a backtracking VNE algorithm based on breath first search and node ranking that uses just one stage to map virtual nodes and links.

All these previous proposals address the VNE problem in the single infrastructure provider (InP) scenario (Intra-domain VNE). However, there are two proposals to perform the VNE in an inter-provider scenario. Chowdhury et al. (2010) propose a policy-based end-to-end VNE framework (PolyViNE) to embed VNs across multiple InPs. Inside any InP, any of the previous intra-domain proposals can be used, but across domains a distributed protocol that coordinated the participating InPs and ensures competitive pricing is introduced. Interdomain VNE is also solved in Houidi et al. (2011) by means of heuristics based on linear programming and max-flow min-cut techniques.

Depending on the assumption taken for the SN, the virtual link mapping is solved in two different ways in the aforementioned VNE proposals: single and multi-path mapping. In single-path mapping each virtual link must be mapped just to a single path in the SN whereas in multi-path mapping each virtual link demand can be carried out by several paths in the SN.

To consider unique substrate path to transport a virtual link (single-path mapping) is often a realistic restriction due to the used routing protocol, or simply an explicit management requirement stipulating avoidance of packet re-sequencing in receiving nodes. However, the problem of mapping multiple virtual link demands to a set of single paths in the SN is $\mathcal{NP}$-complete (Botero et al., 2012). Consequently, existing VNE proposals resolve single-path link mapping using heuristics based on K shortest path routing algorithms for increasing k (Zhu and Ammar, 2006; Lischka and Karl, 2009; Yu et al., 2008; Cheng et al., 2011; Botero et al., 2012). This greedy solution allows to find the best path working in a mono-constraint basis, optimizing just one constraint or a function combining a set of constraints.

To avoid the NP-completeness of the virtual link mapping, some VNE proposals (Yu et al., 2008; Chowdhury et al., 2009, 2010, 2012; Houidi et al., 2011; Cheng et al., 2011) split the virtual link demand and transport it through multiple SN paths (multi-path mapping). To do this, the virtual link mapping is reduced to the Multi-commodity Flow Problem (MFP) that provides a multi-path routing solution for each virtual link using optimal linear programming algorithms (Pioro and Medhi, 2004). It is worth noting that optimal linear solutions are not available when the considered constraints are not linear (e.g., packet loss ratio or availability) and that, although the support of path splitting in the SN entails optimal link mapping solutions, the difficulty of its implementation is higher due to packet re-sequencing.

In this paper, we propose a flexible strategy to solve the single-path virtual link mapping based on paths algebra. Contrary to K shortest path-based mono-constraint approaches, we use the paths algebra-based multi-constraint routing algorithm (LADN) (Herman and Amazonas, 2007) to exploit its capability of *finding* all the possible paths between each pair of nodes in the SN and *organizing* them based on an unlimited number of constraints (or combination of constraints). LADN is composed 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) and ORDERPATH (orders the paths according to the defined metrics and priorities). Our strategy tries to map each virtual link in the best compliant path (previously sorted by LADN). Also, our strategy supports linear and non-linear constraints, or even a combination of them.

Summarizing, the contributions of our strategy to the VNE virtual link mapping stage are threefold:

- It introduces constraint flexibility. Besides CPU and BW, an unlimited number of constraints can be introduced. The multi-constraint routing problem can be solved using any set of metrics resulting in the combinations of different constraints.
- As well as linear, non-linear constraints can be easily introduced in paths algebra.
- Paths algebra finds all eligible paths between each pair of nodes in the SN. This feature can be used for virtual network resiliency or on-line topology changes in mapped VNs.

The paths algebra framework allows to consider multi-constraint routing with linear and not linear constraints providing a more flexible and extendable solution for the virtual link mapping stage of the VNE.

3. The paths algebra framework

In 1979, Carrè (1979) proposed a paths algebra that allows the validation and convergence analysis of the mono-constraint routing problem under a single paradigm independently of the network topology.

Expanding this framework and including the multi-constraint routing problem, Gouda and Schneider (2003) proposed a formal definition of routing metrics listing three basic metrics types: flow, distance and reliability; and two metrics compositions: additive composition and lexical composition. Based on this background, the authors identify and validate two properties: boundedness and isotonicity, as the necessary and sufficient conditions to maximize a metric or, in other words, to build a tree using only the optimized paths that connect a starting node to all other nodes in a network.

Using these ideas, Sobrinho (2001) redefined the concept of paths algebra presented in Carrè (1979) and the isotonicity and strict isotonicity properties defined in Gouda and Schneider (2003). He also indicated the isotonicity property as the necessary and sufficient condition to be obeyed by the path weight function to enable the hop-by-hop routing and to ensure the best path computation through a generalized Dijkstra algorithm. Sobrinho (2001) also proposed the strict isotonicity property as the necessary and sufficient condition to be followed by the path weight function in a hop-by-hop routing for loop avoidance. In 2005, Sobrinho (2005) expands the algebra presented before incorporating the concepts of labels and signatures without analyzing the multi-constraint problem.

More recently, Herman and Amazonas (2007) have harmonized the previous approaches and developed a new generic mathematical framework that can be applied without modifications to different applications. The new framework introduced new tiebreak criteria that allow to distinguish paths considered equivalent by the previous approaches. Its basic principle is the multidimensional lexical ordering.

As the mathematical formalism has already been discussed in Herman and Amazonas (2007), in this paper the paths algebra is presented in a simplified way by means of examples.

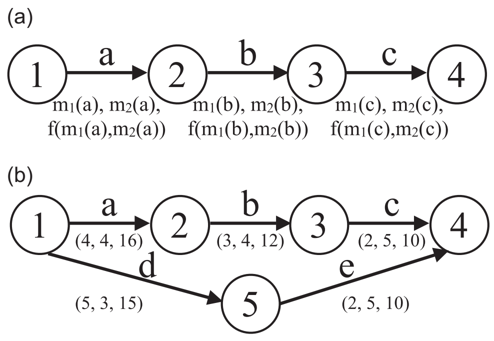
(a)
1
a
2
b
3
c
4
m1(a), m2(a), f(m1(a),m2(a))
m1(b), m2(b), f(m1(b),m2(b))
m1(c), m2(c), f(m1(c),m2(c))
(b)
(4, 4, 16)
(3, 4, 12)
(2, 5, 10)
d
e
5
(5, 3, 15)
(2, 5, 10)

Fig. 2. (a) Example of a simple path. (b) Example of two paths to be ordered.

3.1. 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. 2a. 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 **M** as the set of m adopted routing metrics and **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 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.

3.2. Paths ordering

Consider the network represented in Fig. 2b 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 $\overline{S}[\cdot]=[\min()\max()\mathrm{add}()]^t$.

Table 1
Synthesis result of the network given in Fig. 2b.

Path	S_1 min	S_2 max	S_3 add
α	2; 3; 4	5; 4; 4	38; 28; 16
β	2; 5	5; 3	25; 15

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

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

The result of the synthesis is shown in Table 1. A path α is worse or less optimized than a path β, 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 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.

4. Paths algebra-based virtual link mapping

The example presented in Section 3.2 has shown that the paths algebra provides a great flexibility to the mapping of transport network requirements. Any set of metrics and combination of metrics can be employed, different optimization criteria can be examined simultaneously and the quality of different solutions can be evaluated. It is important to realize that lexical ordering will always provide a total ordering of the paths, while a Cartesian product ordering accounts only for partial ordering.

As far as network virtualization is concerned, once the requirements of the virtual networks requests have been mapped onto a weighted digraph, the paths algebra is a useful tool to find a suitable assignment of the substrate network. It is very powerful to consider different metrics as spare bandwidth, CPU use, reliability, energy consumption, etc. Besides finding the best path between a source and destination pair of nodes, the paths algebra ranks all eligible paths from best to worst, along the cost

Table 3
Metrics for virtual link mapping.

Metric	Definition
Spare CPU	Remaining CPU load available in a SN node after virtual link mapping
Stress	Number of virtual instances running on top of a specific substrate node or link
Spare memory	The remaining memory capacity of a node after mapping a virtual link
Delay	The individual delay of a link belonging to the SN
Link jitter	The jitter of a SN link used in the mapping of a virtual link
Losses	The percentage of packet lost in a link belonging to the SN
Reliability	The individual reliability of a link belonging to the SN
Energy	The energy consumed by each SN node to process the packets
Cost	The cost incurred by each SN node to process the packets
Revenue	Revenue produced the SN to carry out the VN demands

functions values. This is a valuable information for restoration problems, where alternative paths can be stored in the database.

Hidden hops, introduced in Botero et al. (2012), make reference to the intermediate nodes of a directed path in the SN that is mapping a specific virtual link of a VNR. We claim that a hidden hop entails a resource demand because it has to perform packet forwarding of the traffic that will pass through this virtual link.

We assume SN nodes are already mapped, i.e., the substrate node assigned to each virtual node is known. Node mapping has been discussed in several previous works (see Section 2).

Table 3 shows some (but not limited to) possible metrics that can be applied to the virtual link mapping in the VNE.

All metrics are associated as arc weights to the SN, even when they represent a node characteristic as, for example, the spare memory capacity. This association will be detailed in Section 4.1.

As already mentioned in Section 3.1, individual metrics can be combined into a cost function. There is no restriction neither about how the metrics are combined nor about the number of metrics plus cost functions to be used in a specific VNE problem.

The value of a metrics (or cost function) along a path and its corresponding subpaths is evaluated by means of the paths algebra *Synthesis* operation. As already mentioned, four syntheses may be used: (i) minimization—min(); (ii) maximization—max(); (iii) addition—add() and (iv) multiplication—mult(). The synthesis to be used is metric dependent. For example, to evaluate the path delay we have to add the links delay, while to evaluate the path spare bandwidth capacity we need to find the minimum of the links spare bandwidth.

The paths algebra has been implemented by the Loop Avoidance by the Destination (LADN) algorithm (Herman and Amazonas, 2007). In hop-by-hop routing algorithms each node decides about the best next hop to forward a packet independently from the decision by any other node. This may create loops when, for example, a node x to reach a destination d decides that the packet has to be sent to node y and node y to send the packet to the same destination d decides that the best next hop is node x. The packet will travel forever between x and y. The LADN algorithm takes care of this situation and enforces the *coherence* property (Herman, 2008; Herman and Amazonas, 2007) that avoids loops.

4.1. Solving the VNE

To solve the mapping of virtual links we developed a methodology composed of four stages.

We propose a flexible strategy to solve the single-path virtual link mapping based on paths algebra. Contrary to K shortest path-based mono-constraint approaches, we use the paths algebra-based multi-constraint routing algorithm (LADN) (Herman and Amazonas, 2007) to exploit its capability of *finding* all the possible paths between each pair of nodes in the SN and *organizing* them based on an unlimited number of constraints (or combination of constraints). LADN is composed 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 and
- ORDERPATH orders the paths according to the defined metrics and priorities.

Our strategy tries to map each virtual link in the best compliant path (previously sorted by LADN). Also, our strategy supports linear and non-linear constraints, or even a combination of them.

4.1.1. Dealing with node metrics

The paths algebra has been developed associating weights to the arcs of the digraph that represents the network. In order to deal with the metrics associated to the digraph's nodes it is necessary to perform the following transformation. Consider, for example, the part of a digraph shown in Fig. 3a. In this figure, the weights are associated both to the nodes and arcs. Nodes A and B spare CPU capacity are given by CPU(A) and CPU(B), respectively. The spare bandwidth of the link connecting nodes A and B is the arc weight BW(A-B). When a packet traverses the network going from node A to node B it will be processed by both CPUs and flow through the link connecting both nodes. This being so, we can artificially assign all metrics to the arc connecting both nodes as depicted in Fig. 3b.

As one possible objective could be to maximize the spare CPU capacity, the most restricting condition is given by min[CPU(A), CPU(B)]. So, we can use the metrics combination function provided by the paths algebra and reduce the arc's metrics to the pair (BW(A-B), min[CPU(A), CPU(B)]) as shown in Fig. 3c.

It is important to realize that independently of a specific objective, any VNE problem has to assign the VNRs to SN nodes and links, under the constraint of available physical resources. The possibility of the paths algebra to deal with both node and CPU metrics as well, makes it a complete framework to solve the VNE problem. It can be even envisaged that the development of new algorithms in which nodes and link assignments can be done in a coordinated way instead of separate stages as is presently the case.

4.1.2. Paths enumeration

The first step taken by the paths algebra algorithm is to *find all paths connecting all pairs of nodes in the SN*. It may seem that this is a too expensive procedure. This is true for highly connected networks but it is not true for small and sparse networks, what

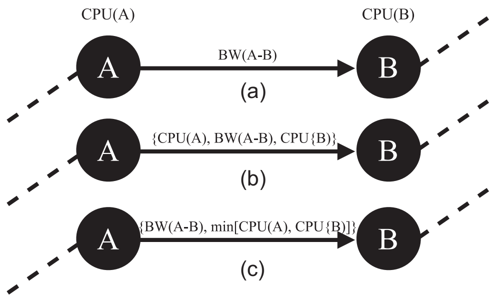

Fig. 3. (a) Weights associated both to nodes and arcs. (b) Nodes' weights associated to the arc connecting the nodes. (c) Final metrics associated to the arc that is to be used by the paths algebra.

is usually the case. It is a procedure to be done only once and the computing time can be neglected as the procedure can be performed off-line (before the embedding process starts). In addition, the procedure pays off in terms of the quality of solution. However, as it will be shown later, the processing time can be significantly reduced by limiting the number of paths to be discovered.

4.1.3. *VN requests ordering*

We consider that mapping optimization must be performed assigning the greatest number of virtual networks requests instead of the greatest number of virtual links. This means that each VN request is considered as a unit independently of how many virtual links it is made of.

It is important to realize that virtual requests consume physical resources, i.e., CPU processing power and BW. Such physical resources are finite and represent the limit of VN assignments. The way they are allocated affects the number of VN requests that can be satisfied. There are different ways to allocate the virtual requests:

(i) allocate them in the order they arrive as if it were a FIFO (First In First Out) procedure;
(ii) allocate them in a non-decreasing, or non-increasing, BW order;
(iii) allocate them in a non-decreasing, or non-increasing, CPU order;
(iv) allocate them in a non-decreasing, or non-increasing, (BW+CPU) order.

Suppose, for example, a substrate network that has a total BW of 10 and there are virtual requests of BW=1, 2, 3, 4, 5. In principle, if we use the non-decreasing order, we will be able to allocate 1, 2, 3 and 4. On the contrary, in a non-increasing order we will be able to allocate only 5, 4 and 1. So, the order in which the VN requests are processed may affect the final result. *A priori* it is not possible to say what is the best policy. If, for example, the number of satisfied VNRs is the optimization criterion, then the non-decreasing order allocation is better. However, if the total amount of assigned BW represents revenue, both policies are equivalent.

Within a virtual request we also have the choice of the order of how the virtual links should be allocated.

A deep investigation of the effect of VN requests and VN links ordering on the quality of the obtained results is out of the scope of this work. We have chosen to use the VN requests FIFO ordering that emulates an on-line operation and non-increasing (BW+CPU) for off-line operations. This will decrease the chance to have to perform backtrack. i.e., to re-assign a previous virtual link in order to accommodate another one.

As far as the VN links ordering is concerned no specific ordering has been done and the VNR matrix is read by rows.

4.1.4. *VN links assignment*

Having performed the aforementioned preparatory steps, a virtual link is picked up to be assigned to the SN. Any assignment will consume SN's resources and the packets will undergo the effects of the path it traverses: delay, jitter, etc. The SN's weight arcs are adjusted to the values they would have if the SN's resources were consumed according to the corresponding VN request. Using the paths algebra the paths are ordered taking into account all specified metrics. The virtual link is assigned to the best path and the SN resources are updated according to the real values consumed by the chosen path. This procedure is repeated until all virtual links have been assigned to the SN. In case a virtual assignment is not found because there are no enough resources to accommodate the request, an analysis can be made to find out if a re-assignment (backtrack) of a previously assigned virtual link would free enough resources to accommodate the current virtual link. This analysis is left for future work. Currently, if there are no available resources to map a virtual link it is necessary to define if the VN request will be simply dropped out or if it will be accommodated with less resources than originally demanded. It is important to point out that the decision is related to the adopted business model and it is not affected by the paths algebra.

4.2. *Proof of concept examples*

In this section we introduce two examples as proofs of concept. The first example deals with only physical resources: CPU and BW. It is a very simple example to show the basics of the paths algebra procedures. The second example introduces PLR and cost/revenue (C/R) to show how the paths algebra deals with non-linear metrics and non-physical resources.

4.2.1. *Example 1*

Consider the SN and VN requests represented in Fig. 4. For SN, weights (in bold) over links indicate links' bandwidth and the nodes' weights (in parenthesis) are the nodes' capacity, link delay is shown in italics. For the VNs, the arcs weights indicate the requested bandwidth, and the requested virtual link delay refers to the maximum delay allowable when the virtual link is mapped, nodes' weights are the total CPU capacity requested for terminal nodes, and the arcs' weights shown between parentheses are the total CPU capacity requested for hidden hops. Total CPU requested is to be understood as the capacity demanded by the whole VN request, independently of the number of times the node is used (as terminal or hidden) by the SN assignment.

In this example, the objectives to be fulfilled are:

- assign the largest number of virtual networks to the substrate network;
- maximize the global spare resources in SN, i.e., the links spare bandwidth added to the nodes spare CPU capacity. This objective can be fulfilled by mapping each virtual link using the shortest available path;
- minimize the path delay.

The metrics, syntheses and ordering relations to be used to fulfill the listed objectives are $\mathbf{M}=\{\text{Hops,Delay}\}$; $\mathbf{S}=\{\text{add,add}\}$; $\preceq_{ML}=\{\geq,\geq\}$.

Additionally, the links spare bandwidth and the nodes spare CPU capacity will be used as lower priority metrics to verify if the

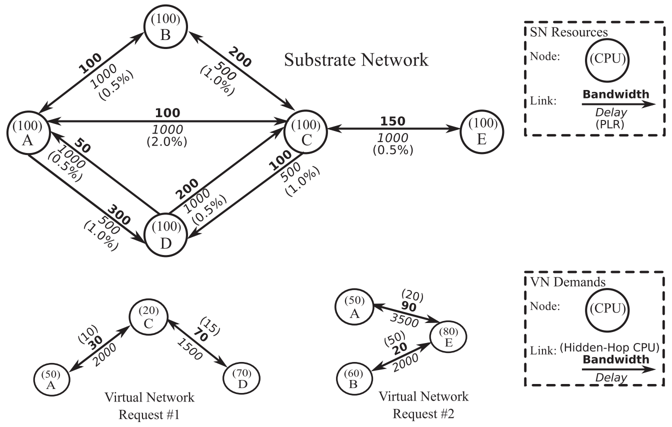

Fig. 4. SN and VN requests used in the example.

SN has enough resources to accommodate the VN requests. The corresponding metrics, syntheses and ordering relations are $\mathbf{M} = \{\mathrm{BW,CPU}\}$, $\mathbf{S} = \{\min,\min\}$; $\preceq_{ML} = \{\leq,\leq\}$.

All the above criteria can be summarized as:

- $\mathbf{M} = \{\mathrm{Hops,Delay,BW,CPU}\}$;
- $\mathbf{F} = \mathbf{M}$;
- $\mathbf{S} = \{\mathrm{add,add,min,min}\}$;
- $\preceq_{ML} = \{\geq,\geq,\leq,\leq\}$.

It is important to note that there are different types of demands in the VN requests. Some resources lessen when virtual link requests are being assigned, and some others do not. Bandwidth and CPU demands are subtracted from the SN resources each time a virtual link is mapped, while delay remains unchanged after a virtual link mapping. Delay acts as a constraint: a SN path with a delay greater than the indicated in the virtual link request cannot be used to map it.

The first step in solving the VNE problem is to enumerate all paths between pair of nodes of the SN. Such enumeration is completely independent of the VN requests. The pair of nodes of the SN are then used as terminal nodes by the VN requests. This sequence is also followed by the LADN algorithm. The obtained paths are:

- (s, d)=(C, D): (C, D), (C, A, D), (C, B, A, D);
- (s, d)=(A, C): (A, C), (A, B, C), (A, D, C);
- (s, d)=(A, E): (A, C, E), (A, B, C, E), (A, D, C, E);
- (s, d)=(B, E): (B, C, E), (B, A, C, E), (B, A, D, C, E).

Next, the VN requests have been sorted in an increasing order of total demand, and within a VN request the virtual links to be assigned to the SN have been sorted in a decreasing order. We calculate total demand as the weighted sum of requested CPU and bandwidth; to simplify the example we consider equal CPU and bandwidth weights of value 1. CPU demand of hidden-hops is not considered to calculate total demand. The final ordering is:

1. VN #1: total demand=240; links ordering: (C, D), (A, C);
2. VN #2: total demand=300; links ordering: (A, E), (B, E).

For each virtual network link request the digraph arcs' metrics are updated by subtracting the bandwidth and CPU capacity demands. The paths are then synthesized and lexically ordered. In this case, it was used as the whole weight-word. The synthesis results for VN #1 virtual link (C, D) are:

Path	*S1* *# hops*	*S2* *delay*
(C, D)	1	500
(C, A, D)	2; 1	2000; 1000
(C, B, A, D)	3; 2; 1	2000; 1500; 500

Path	*S3* *BW*	*S4* *CPU*
(C, D)	30	30
(C, A, D)	30; 30	30; 80
(C, B, A, D)	30; 30; 130	30; 80; 80

The chosen path is (C, D): it is the shortest one; the delay is less than 1500; the spare BW and CPU are positive indicating that the SN has enough resources to accommodate the request. The assignment of the virtual link (A, C) does not present any difficulty and it is mapped to the SN's path (A, C). At the end of the VN #1 assignment the SN's resources are updated and the new values are:

- CPU capacity: (A)=50; (C)=80; (D)=30;
- links BW: (A, C)=70; (C, D)=30.

The other values remain the same.

The synthesis results for VN #2 virtual link (A, E) are:

Path	*S1* *# hops*	*S2* *delay*
(A, C, E)	2; 1	2000; 1000
(A, B, C, E)	3; 2; 1	2500; 1500; 1000
(A, D, C, E)	3; 2; 1	2500; 1500; 500

Path	*S3* *BW*	*S4* *CPU*
(A, C, E)	−20; −20	0; 0
(A, B, C, E)	10; 10; 10	0; 0; 0
(A, D, C, E)	60; 110; 210	0; 0; 0

It is seen that the shortest path does not have enough resources to fulfill the demand because the resulting value of S3 is negative. The chosen path is (A, D, C, E) because it shows a lower delay in the third letter of the weight word S2 than the path (A, B, C, E). It is important to realize that if we were using only the first letter, both paths would be equivalent. After this assignment the SN's resources are updated and the new values are (note that CPU capacities are also updated subtracting hidden-hops D and C demands):

- CPU capacity: (A)=0; (C)=60; (D)=10; (E)=20;
- links BW: (A, D)=210; (D, C)=110; (C, E)=60.

The synthesis results for VN #2 virtual link (B, E) are:

Path	*S1* *# hops*	*S2* *delay*
(B, C, E)	2; 1	1500; 500
(B, A, C, E)	3; 2; 1	3000; 2000; 1000
(B, A, D, C, E)	4; 3; 2; 1	3500; 2500; 1500; 1000; 500

Path	*S3* *BW*	*S4* *CPU*
(B, C, E)	40; 180	20; 40
(B, A, C, E)	40; 50; 80	−5; −5; −5
(B, A, D, C, E)	40; 80; 80; 80	−5; −5; −5; −5

The shortest path (B, C, E) is chosen because it fulfills all conditions. It is also worth noting that it is the only possible solution because the other alternatives do not have enough CPU capacity (S4 with a negative value).

All assignments were made without having to perform any backtrack.

4.2.2. *Example 2*

In this example, consider again the SN and VN requests represented in Fig. 4. However, in this case, we will simultaneously consider the following aspects:

- the use of a non-linear metrics;
- a QoS-aware mapping;
- a business optimization criterion.

Table 4
Modeling of optimization strategies.

Optimization criterion	Optimization constraints	Physical thresholds	QoS	**M** **F**
Minimize cost	Hops	CPU, BW	–	**M** = {Hops, BW, CPU} **F** = **M**
Minimize cost under a maximum allowable delay	Hops	CPU, BW	Delay	**M** = {Hops, Delay, BW, CPU} **F** = **M**
Maximize the spare CPU	CPU	CPU, BW	–	**M** = {CPU,BW} **F** = **M**
Maximize the spare BW	BW	CPU, BW	–	**M** = {BW,CPU} **F** = **M**
Maximize the spare physical resources	CPU, BW	CPU, BW	–	**M** = {BW,CPU} **F** = {CPU + BW}
Minimize cost and maximize throughput, under a maximum allowable delay	Hops, PLR	CPU, BW	Delay	**M** = {Hops,PLR,Delay,BW, CPU} pTHRU = 1 - PLR **F** = {Hops,pTHRU,Delay,BW, CPU}

In Fig. 4 each SN link is also characterized by its Packet Loss Rate (PLR), depicted by the value in parenthesis under the substrate links.

From the PLR, we apply a function of metrics combination and evaluate for each link the packet throughput, designated as pTHRU, as pTHRU = 1−PLR. The synthesis associated to pTHRU is mult() and the ordering relation is $\leq$.

It is important to realize that PLR is both a non-linear and a QoS metrics as well but it introduces no difficulties to the paths algebra as it can be treated by one of the proposed syntheses.

Let us also consider that the demanded BW (in this case equal to BW(A–C) + BW(C–D) = 100) is a measure of revenue (R) and that the employed CPU processing power is a measure of cost (C).

Summarizing, the corresponding syntheses and ordering relations for pTHRU and C are **S** = {mult,add}; $\preceq_{ML} = \{\leq, \geq\}$.

Using only the first letter of weight word for pTHRU and C, the synthesis results for VN #1 are:

link-(C–D)

Path	*S5* *pTHRU*	*S6* *C*	*PLR* (%)	*C/R*
(C, D)	0.9999	90	1.0000	0.90
(C, A, D)	0.9702	105	2.9800	1.05
(C, B, A, D)	0.9752	120	2.4801	1.20

link-(A–C)

Path	*S5* *pTHRU*	*S6* *C*	*PLR* (%)	*C/R*
(A,C)	0.9800	70	2.0000	0.70
(A, B, C)	0.9850	80	1.4950	0.80
(A, D, C)	0.9850	80	1.4950	0.80

When only BW and CPU had been considered, for VN #1 virtual link (C–D), the chosen path was (C, D): it is the shortest one; the delay is less than 1500; the spare BW and CPU are positive indicating that the SN has enough resources to accommodate the request. It is also the best one when PLR and C/R are taken into account. So, for the VN #1 virtual link (C–D) the result does not change if either a QoS or a business criterion is employed.

It is also important to realize that lower PLR does not necessarily implies lower C/R. It is enough to observe the values obtained for paths (C, A, D) and (C, B, A, D).

Formerly, considering only the availability of physical resources, the assignment of the virtual link (A, C) was mapped to the SN's path (A, C). However, under QoS-aware PLR criterion it is the worst path. So, either (A, B, C) or (A, D, C) could be chosen. Under the PLR criterion they are equivalent. If PLR is not used as an ordering criterion but as a threshold, the path (A,C) could still be chosen if a 2.0% loss rate is acceptable.

If only the business criterion C/R is used the path (A, C) would be chosen. However, if a QoS-aware threshold of PLR equal to 1.5% has to be achieved then the choice would be between paths (A, B, C) and (A, D, C).

The example is not developed further, but similar conclusions can also be derived for the VN #2.

In this example, more important than the result itself is to realize how powerful and flexible the paths algebra approach is. It allows the InP to exercise different policies to achieve the SP goals. In this way, both technical and financial performances can be optimized. The access to the performance of all eligible paths also implies that recovery strategies can be implemented.

All the above observations were possible by just inspecting the results of the paths algebra syntheses. However, the paths algebra enables to explore different optimization strategies by just changing the metrics **M** and the function of combined metrics **F**.

Table 4 shows how some different strategies can be modeled using the paths algebra. In this table, the optimization metrics are primarily used to order the enumerated paths. The physical constraints indicate if the achieved mapping is feasible or not. A feasible mapping requires positive physical constraints. A QoS threshold is a lower priority metrics and represents a bound on the value of the corresponding metrics.

In Table 4 any strategy that minimizes cost (a business objective) also minimizes the cost over revenue because the demanded VNR's BW is constant and is a measure of the achievable revenue. For each strategy the number of mapped VNRs may be different and the mapped revenue as well. So, different strategies may be exploited to optimize business objectives and achieve different levels of QoS. The listed strategies also mix linear and non-linear metrics that are treated simultaneously by the paths algebra.

As mentioned before, the LADN is made of four stages: SEARCHPATH, SORTPATH, EVALUATEPATH and ORDERPATH. The SEARCHPATH and SORTHPATH are metrics independent. The paths enumeration depends only on the SN topology. So, these two routines can be run off-line and the results stored to be exploited by different optimization strategies. Different strategies imply running EVALUATEPATH and ORDERPATH, which can be done either on-line or off-line.

4.3. *Preliminary conclusions*

The paths algebra based strategy transforms the link mapping stage of the VNE problem into the simpler problem of mapping network virtualization policies into an appropriate set of metrics or combination of metrics. With paths algebra, the addition of new metrics will not go in detriment of the problem complexity. In addition, the order in which the metrics are considered introduces newer degrees of freedom. The paths algebra is a flexible and powerful tool to explore the design space to find the best strategy according to a specific business model.

It is important to realize that even if the discovery of all eligible paths is computing expensive, it has to be done only once, it can be performed off-line and is metrics independent.

5. Evaluation scenarios and experimental results

In this section we show how to deal with the computational complexity of the multi-constraint routing algorithm, we introduce the simulation scenarios and present the experimental results.

5.1. Computational complexity

The paths algebra-based multi-constraint routing algorithm presented in Herman and Amazonas (2007) is made of four routines:

1. SEARCHPATH: discovers all paths between all pair of nodes of a given network.
2. SORTPATH: selects only the elementary and simple paths.[2]
3. EVALUATEPATH: evaluates the syntheses along the paths.
4. ORDERPATH: order the paths according to the metrics and priorities of a given problem.

When the graph is fully connected with n nodes and $n \times (n-1)$ edges, the maximum number of simple and elementary paths can be obtained by induction and is given by $Np = 2n! + n!/2! + n!/3! + \cdots + n!/(n-2)!$ that can be written as $\sum_{i=0}^{n-2} C_{n,i} \times (n-i)!, n > 3$. Substituting the combination of (n) taken i at a time by $(n)!/[i! \times (n-i)!]$, we obtain $Np = (n)! \sum_{i=0}^{n-2} 1/i!$. Therefore, for this case, the algorithm's execution time is $O(n^n)$, given that $\sum_{i=0}^{n-2} 1/i! < \sum_{i=0}^{\infty} 1/i! = e$, for $n \to \infty$, $e \times n! < e \times n^n \Rightarrow n! < n^n$, in which according to the usual definition of $O(g(n))$, for $n \geq n_0 = 2 : 0 \leq f(n) = ! \leq cg(n) = n^n$.

This means that for fully connected networks the paths algebra-based multi-constraint routing algorithm is computationally explosive. In practice, we have verified that for real networks with a small number of nodes and sparse connectivity and it performs well. However, it is not guaranteed that it would perform adequately for any network but, as it will be shown in the sequence, we can control the number of operations performed by the algorithm without sacrificing the quality of results. By doing so, we can apply the algorithm to networks with hundreds of links.

In order to estimate the algorithm's performance we generated 100 random uniform and transit stub topologies with 10, 20, 50 and 100 nodes, and evaluated the total number of paths. The results have shown that the number of paths do increase as n^n. Thus, it is necessary to introduce some modification in order to decrease the algorithm's complexity.

Consider a given network specified by its adjacency matrix A where the element $a[i,j]$ is equal to 1 when there is a link connecting the source node i to the destination node j. If we take

[2] Elementary and simple paths are those paths that traverse each graph's node and edge only once.

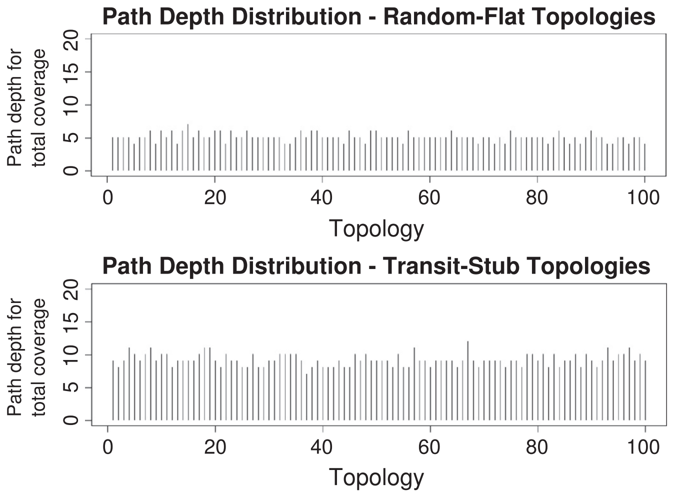

Fig. 5. Path depth distribution for 50 nodes in random flat and transit stub topologies.

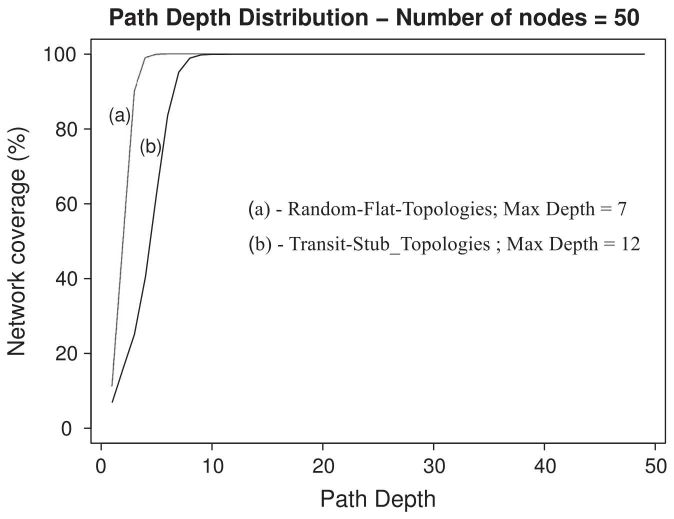

Fig. 6. Connectivity increase as a function of the path depth for 50 nodes random flat and transit stub topologies.

$A^k = A \times A \times \cdots \times A$, k times, where $a_k[i,j] \neq 0$ indicates that there is at least a path of length k connecting the source node i to the destination node j. It is then possible, for each network, to evaluate the minimum length of paths that guarantee full connectivity, i.e., that there exists at least one path between any pair of source and destination nodes (Chartrand, 1985).

Figure 5 shows the path depth which is the path length necessary to obtain full connectivity for 100 topologies both for 50 nodes random flat and transit stub cases. For an n nodes topology the largest path length is $(n-1)$. So, for the results shown in Fig. 5 the maximum length is $(n-1) = 49$. It can be seen that in both cases the necessary path length for obtaining 100% coverage is much less than the maximum value. Another way to appreciate this finding is observing Fig. 6 that shows the connectivity increase as a function of the path length. These results were obtained by evaluating the average coverage for each path length over all topologies. It can be seen that for both types of topologies, 100% connectivity is achieved for path lengths quite smaller than the maximum value. As the number of employed topologies is reasonably large, we may state that the obtained results are representative for classes of topologies studied in this paper.

Table 5 summarizes the results for topologies with 20, 50 and 100 nodes.

Table 5
Minimum path depth for 20, 50 and 100 node topologies.

Number of nodes	Transit stub topologies	Random flat topologies
20	10	12
50	12	7
100	13	5

Table 6
Comparison between the average maximum of paths with the number of paths up to the length equal to 6 links.

Number of nodes	Average Maximum number of paths	Number of paths up to length=6
Random flat topologies		
20	3162	692
50	117,489	6,607
100	1,995,262	41,687
Transit stub topologies		
20	1738	447
50	16,982	2,042
100	66,069	3,467

Table 6 compares the average maximum number of paths with the number of paths up to the length equal to 6 links for random flat and transit stub topologies with 20, 50 and 100 nodes. It can be seen that limiting the maximum length of the paths produces an important reduction on the number of paths to be discovered and processed. This modification has been introduced into the multi-constraint routing algorithms, providing a control over the number of operations performed by the algorithm and enabling the possibility to work with large networks. It is important to realize that the path length limit introduced in the algorithms is not an arbitrary value. In fact, it is determined as a function of the SN that is going to be processed. It is just a matter of finding the exponent k such that $\bigcup_{i=0}^{i=k} A^i = \mathbf{1}$, where $\mathbf{1}$ represents a matrix where all elements are equal to 1. On the other hand, it may be argued that limiting the search space may impact the possibility of finding a mapping for a VN request. However, the results obtained for the VNE mapping show that, if there is any impact, it seems to be of negligible practical importance.

5.2. Simulations environments

The simulation environment used for VNE is the Algorithms for Embedding of Virtual Networks (ALEVIN) (Fischer et al., 2011; VNREAL, 2011). ALEVIN is a framework to develop, compare, and analyze virtual network embedding algorithms. A set of popular VNE algorithms has been implemented in ALEVIN and more algorithms can be added to allow researchers run simulations, investigate and do modifications for a new set of optimization criteria.

As already mentioned, the algorithm used to implement paths algebra, called LADN (Herman and Amazonas, 2007), was developed as a simulation tool for testing multi-constraint hop-by-hop routing algorithms in which the optimization strategy is user-defined. Considering that in hop-by-hop routing algorithms each node decides about the best path independently of any other nodes, loops may occur. The algorithm eliminates loop occurrences by imposing the property of coherence on a global basis to the next hop choice made by each node.

The LADN algorithm was modified to solve the VNE problem, i.e., to map virtual links to physical paths in a substrate network. Node mapping is performed by the algorithms implemented in ALEVIN and is out of the scope of this work.

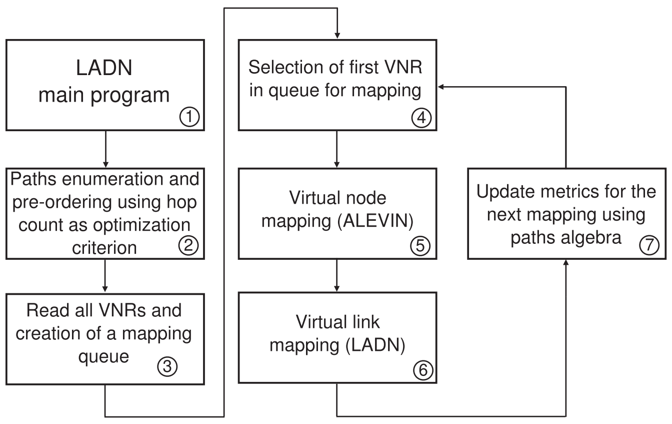
LADN main program ①
Paths enumeration and pre-ordering using hop count as optimization criterion ②
Read all VNRs and creation of a mapping queue ③
Selection of first VNR in queue for mapping ④
Virtual node mapping (ALEVIN) ⑤
Virtual link mapping (LADN) ⑥
Update metrics for the next mapping using paths algebra ⑦

Fig. 7. LADN and ALEVIN integration for VNE.

Figure 7 shows the diagram of the implementation, the VNE solution using the association of paths algebra LADN and the ALEVIN environment. The ALEVIN environment is in charge of generating the VNRs and of performing nodes mapping. The other tasks are performed by the LADN. Presently the environments communicate by files exchange that impacts the processing time. A complete integration will be implemented in a near future.

According to Fig. 7, block 1 starts the LADN environment and block 2 enumerates the eligible paths and pre-order them according to the number of hops metrics. Block 2 also implements the maximum length of the searched paths in order to limit the processing time.

Block 3 defines if mapping will be done on-line or off-line. In the case of on-line mapping, the VNRs must be processed in the order of their arrival as if they were stored in a FIFO (first in first out) queue. In the case of off-line mapping, the VNRs are ordered according to some policy, e.g., most consuming first (MCF), least consuming first (LCF), etc. In this work, the MCF policy has been selected.

Block 4 selects the first VNR waiting to be mapped and sends it to the ALEVIN along the current status of the SN, i.e., the available resources in the SN.

Block 5, implemented in ALEVIN, produces a virtual nodes to physical nodes mapping and sends the results back to the LADN environment.

Block 6 maps the virtual links to the physical links according to the set of metrics and priorities selected by the user. If the mapping is successful the algorithm proceeds to block 7 to evaluate the new available resources in the SN and to update the metric matrices. If there are still pending VNRs, the algorithm goes back to block 4 and repeats the procedure. If there are no pending VNRs, the algorithm stops and stores the results. If the mapping is not successful, the algorithm records the failure for further analysis and tests about pending VNRs.

5.3. Simulation methodology

For the comparison of former VNE algorithms with the paths algebra algorithm, we use the following methodology to create scenarios. The scenarios are parameterized by:

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

5.3.1. 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, α and β, that determine the probability of an edge connecting two nodes, as given by

$$P(u,v) = \alpha e^{-d(u,v)/(\beta \times L)}, \tag{1}$$

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 α increases the probability of edges between any nodes in the graph. An increase in the parameter β 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.

5.3.2. *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 the 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

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

as well as the mean demand on a virtual node for a given overall load ρ as

$$E[\mathrm{ND}_X] = \rho \cdot E[\mathrm{NR}_X] \cdot \frac{|V|}{|V^k| \cdot k^{\max}}, \tag{3}$$

in which

- $|V|$ is the number of substrate nodes;
- k is the number of virtual networks; and
- $|V^k|$ is the number of virtual nodes per virtual network.

Eq. (2) results in a maximum resources demand

$$\mathrm{ND}_X^{\max} = 2 \cdot E[ND_X] \tag{4}$$

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 ρ 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

$$E[|A|] = E[p] \cdot |V| \cdot (|V| - 1). \tag{5}$$

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

$$E[\mathrm{LR}_Y] = \frac{0 + \mathrm{LR}_Y^{\max}}{2} = \frac{\mathrm{LR}_Y^{\max}}{2} \tag{6}$$

and the average link demand is given by

$$E[\mathrm{LD}_Y] = \rho \cdot E[\mathrm{LR}_Y] \cdot \frac{E[|A|]}{E[|A^k|] \cdot k^{\max}} \tag{7}$$

which results in

$$\mathrm{LD}_Y^{\max} = 2 \cdot E[\mathrm{LD}_Y]. \tag{8}$$

However, it is necessary to ensure that $\mathrm{LD}_Y^{\max} \leq \mathrm{LR}_Y^{\max}$ holds. In particular, it is necessary to ensure that $E[|A|] < E[|A^k|] \cdot k^{\max}$ in Eq. (7) holds for $0 \leq \rho \leq 1$. To that end, we have to enforce the constraint

$$|V|^2 < k^{\max} \cdot |V^k|^2. \tag{9}$$

If the constraint in Eq. (9) is violated, the approximation provided above will achieve a higher load value than the targeted load ρ for the link resources and even result in $\mathrm{LD}_Y^{\max} > \mathrm{LY}_Y^{\max}$ which can never be fulfilled.

5.3.3. *Scenarios*

As described in Section 5.3.1, we use the Waxman topology generation. For evaluation, we used $\alpha = \beta = 0.5$ and distributed the coordinates of the nodes uniformly in an 1×1 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. (1) 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 $\mathrm{NR}_{\max} = 100$.

Table 7 shows the parameters used in the simulations. The values chosen for the number of substrate nodes and the number of virtual nodes per virtual network seems to be a good trade-off between runtime of some algorithms and realistic scenarios. The loads' range from 0.2 through 0.7 also covers situations from lightly loaded to heavily loaded situation.

As Waxman the topology generation is probabilistic, we perform N=20 runs for each value of load.

5.4. *Experimental results*

This paper proposes a new strategy to face the network embedding optimization, permitting linear and non-linear parameters. Relevant traditional QoS parameters such as packet loss ratio cannot be included in current methods. According to the current state of the art, only results with linear parameters are published yet. Therefore, this paper analyzes and compares results with the relevant linear parameters based strategies. These comparisons show that the proposed procedure in this work is more powerful, since it produces similar results and, in addition, provides the capacity to include non-linear parameters. Apart from the comparison with existing methods, an example with non-linear parameters is depicted in Section 4.2.2.

Table 7
Parameters chosen for the simulation scenarios.

Parameter description	Chosen values
Number of substrate nodes ($\lvert V\rvert$)	20
Number of VNRs (k)	10
Number of virtual nodes per virtual network ($\lvert V^k\rvert$)	10
Range of loads (ρ)	{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8}

Table 8
Algorithms used in the simulations.

Algorithm's name	Algorithm's description
GARSP (Yu et al., 2008)	**G**reedy **A**vailable **R**esources with k-**S**hortest **P**aths.
PathsAlgebra policy=**M1**	Greedy available resources for node mapping and paths algebra for link mapping using $\mathbf{M}$=(Hops,BW,CPU), $\mathbf{S}$=(add,min,min) and ordering relations $(\geq, \leq, \leq)$.
PathsAlgebra policy=**M2**	Greedy available resources for node mapping and paths algebra for link mapping using $\mathbf{M}$=(Hops,TotalBW), $\mathbf{S}$=(add,add) and ordering relations $(\geq, \leq)$.

Hops means the number of hops in the path.
BW means the links available bandwidth after mapping.
Total BW is the total SN available BW after mapping.

Optimal cost-based solutions for the VNE problem have been provided in the literature. In Houidi et al. (2011), VNE is formulated as a MIP and solved using exact algorithms (Wolsey, 1998). However, MIPs are not scalable and the time needed to solve them in medium to large networks is not affordable. Besides, unlike our approach, the MIP solution proposed in Houidi et al. (2011) provides multi-path virtual link mapping. To make this cost-optimal solution comparable with our approach, additional binary variables indicating that just one path can carry the virtual link demand should be added to the MIP model, resulting in further complexity that would increase the time to solve the MIP even for very small substrate and virtual networks.

Consequently, to evaluate our paths algebra-based approach, we compare it against a well-known single-path VNE heuristic (Yu et al., 2008). Simulations have been performed for all scenarios described in Section 5.3.3. Our approach considers just the single path-based virtual link mapping stage of the VNE. As a consequence, to compare our approach against other VNE solutions, we reuse the virtual node mapping made by them and focus on the virtual link mapping stage. As most of the current VNE approaches performing single path-based virtual link mapping use k shortest path approach (see Section 2), we evaluate the performance of the paths algebra approach against the GARSP algorithm (also called baseline algorithm in Yu et al., 2008) available in the ALEVIN environment.

Table 8 summarizes the characteristics of the algorithms used in the simulations. It is important to note that all algorithms produce single path solutions. Multipath investigation is left for future work.

Note that two different policies for the Paths Algebra are being employed. The difference between them is the set **M** of metrics and the appropriate syntheses and ordering relations for each set. The maximum path length is evaluated according to the SN that is being processed.

Among the several parameters that can be used to compare the results obtained by the different algorithms we have chosen to use the following:

- *VNR acceptance ratio*: The percentage of virtual networks successfully mapped;
- *Mapped revenue ratio*: The percentage of mapped revenue over the total revenue that could be mapped;
- *Cost/revenue* (C/R) *relationship*: The lower this relationship is, the better is the result.

Out of 160 simulated cases, the paths algebra produced the best result 155 times, i.e., in 96.87% of the cases.

The discussion of the results shown in this section and the exploitation of numerical results is made in Section 5.5.

Table 9
Summary of the best results obtained for different parameters and policies.

Parameter	M1 + M2	GARSP
Accepted VNRs	155	5
Ratio mapped revenue	144	16
C/R relationship	133	27

5.5. Results compared to cost-based Heuristic

The total number of times that each algorithm has been exercised is given by $N \times k \times |\rho| = 20 \times 10 \times 8 = 1600$, and as we are comparing 3 different policies, the total number of VNR assignment attempts is 4800.

The dimension of the sample space does not allow to derive quantitative conclusions with statistical significance but it is large enough to provide qualitative comparisons and obtain an understanding of the observed behaviors.

Table 9 summarizes the best results obtained for different parameters and policies.

It is clear that the paths algebra policies performance almost always exceeds that of the other strategy. It is not possible at this stage to indicate a specific reason why the paths algebra policies is not always the best. Looking at the whole set of results we observe that there is not any common pattern among the cases that the paths algebra has not produced the best result. Such cases need to be further investigated to identify the specific reason underlying a degraded performance of paths algebra. Possible explanations for such cases may be:

- different policies may employ different node mappings that may affect link mapping. If this is the case, a better coordination between node and link mapping has to be developed;
- the non-paths algebra strategies may achieve a better result by using longer paths. If this is the case an attempt may be made by relaxing the limit of paths length and trying to find a mapping;
- the order the VNRs are processed may impact the final result. A backtrack procedure may be needed.

It is also important to realize that there are cases in which M1 produces the best result and cases in which M2 produces the best result. The right interpretation of this observation is that the paths algebra is powerful and flexible to allow the identification of the adequate policy to optimize the performance of a given parameter. Moreover, it is not necessary to choose a policy beforehand. Several policies can be simultaneously implemented without any significant computational time increase. Remember that the enumeration of eligible paths is computationally explosive. However, the number of operations is under control by the limitation imposed on the paths length and the enumeration of

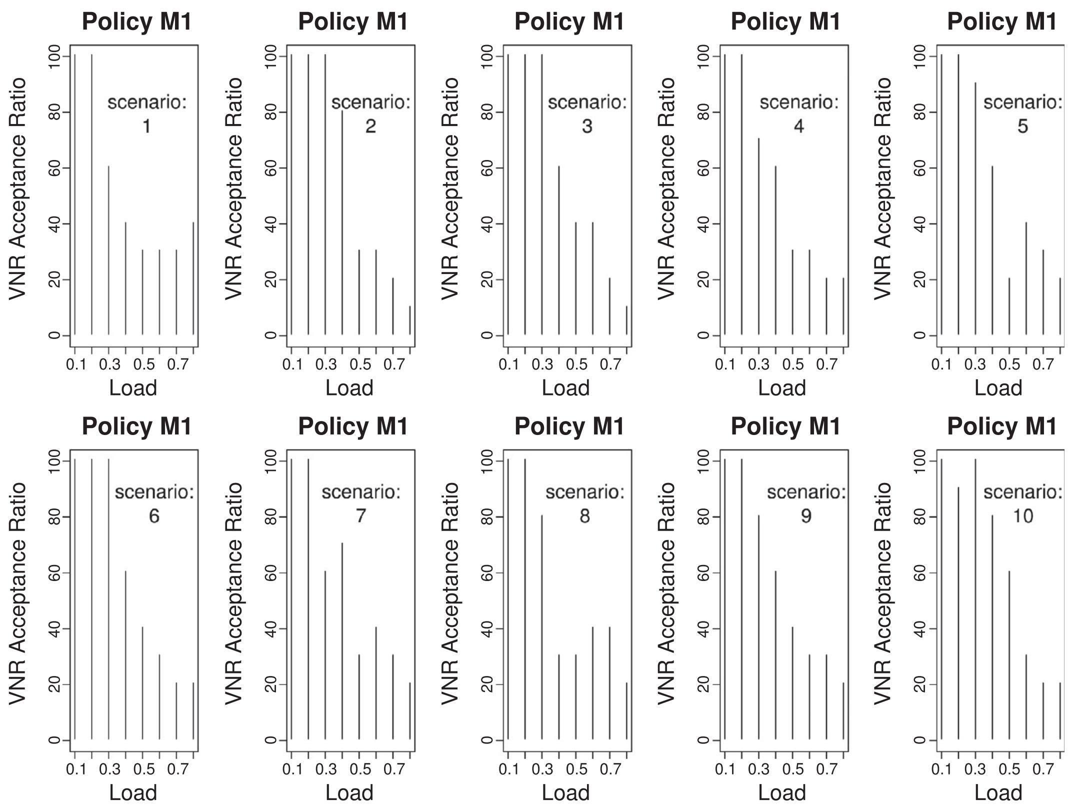

Fig. 8. VNR acceptance ratio for M1 policy.

eligible paths may be performed off-line and is independent of any optimization strategy.

Figure 8 shows part of (only 10 out of 20 simulated scenarios) the obtained results of accepted VNRs for the M1 policy. The histogram like graphs represent the results of each evaluated scenario. In all cases we observe a trend of a decrease in the number of accepted VNRs as the load increases. Some scenarios do not present a monotonic behavior. Such instabilities have to be investigated in a case by case basis and it is most likely that the explanations proposed above concerning the reasons why the paths algebra is not always the best one may apply.

Figure 9 summarizes the results obtained for the VNR acceptance ratio. In top three figures, figures (a)–(c), are shown the mean values for all policies along the 95% confidence interval. The two bottom figures, figures (d) and (e), show the comparison between the paths algebra policies (M1 and M2) and the GARSP strategy. The comparison is made based on the mean values and it is clearly seen that the paths algebra policies outperform the GARSP strategy. In addition all policies present the same behavior and the 95% confidence interval is similar for all of them.

The observed behavior can be understood if we consider that a SN offers a certain amount of resources to be used by the VNE procedure. The total resource that will be actually used is a fraction of the total CPU and BW available. When the load ρ offered to the SN is low, there are enough resources to absorb the demand and all VNRs are successfully mapped. Increasing the load offered to the SN a value is reached, ρ_0, that all VNRs are still successfully mapped but all SN resources are used. Above ρ_0, the excess load cannot be mapped and the VNR will be dropped. So, for a load $\rho > \rho_0$ we may expect that the percentage of accepted VNRs will roughly follow a ρ_0/ρ trend.

As the revenue is proportional to the number of accepted VNRs, the ratio mapped revenue has the same behavior as the percentage of accepted VNRs as it can be seen in Fig. 10.

In order to understand the obtained results for C/R it is necessary to consider how Cost and Revenue are evaluated. Let us assume that there is a VNR characterized by:

- virtual source node: A
- virtual destination node: B
- requested bandwidth: BW(A–B)=100
- requested CPU: CPU(A)=20; CPU(B)=30

For this request the associated revenue is: R=CPU(A) + BW(A–B) + CPU(B)=20 + 100 + 30=150.

Let us assume that the paths algebra algorithm finds two possible mappings summarized in Table 10.

It can be readily seen that the best solution is achieved when a virtual link is directly mapped on a single substrate link. In this case, $C/R = 1.0$. Each time a hidden node has to be used, the cost increases by the cost of the CPU plus the cost of the bandwidth. As in these examples the cost of the hidden nodes' CPU is taken equal to zero then for a mapping that uses k hidden nodes the cost is given by

$$\mathrm{C} = \mathrm{CPU(A)} + \mathrm{CPU(B)} + (1+k)\mathrm{BW}.$$

As the revenue for a given VNR is a fixed number given by $\mathrm{R} = \mathrm{CPU(A)} + \mathrm{CPU(B)} + \mathrm{BW}$, then C/R is given by:

$$\mathrm{C/R} = 1 + k\frac{\mathrm{BW}}{\mathrm{R}},$$

i.e., C/R increases linearly as the number of hidden nodes increases.

All strategies evaluated in this paper take the minimization of cost as a priority. This means that all strategies try to map a VNR using a shortest available path and the shortest paths will be consumed first. When the load increases, in order to satisfy the VNRs, longer paths have to be used and the cost will increase. It is then expected that C/R will show

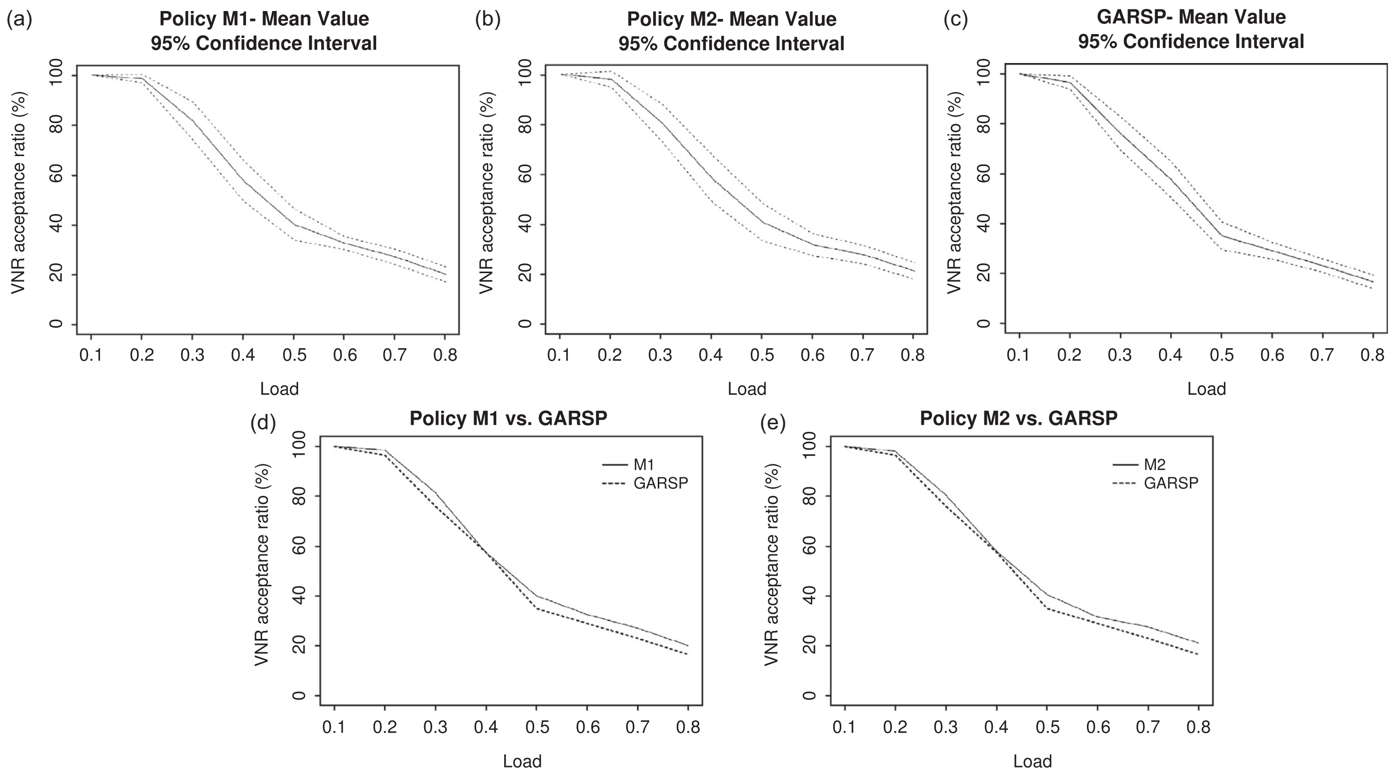

Fig. 9. VNR acceptance ratio mean values behavior for (a) M1, (b) M2 and (c) GARSP policies and comparisons: (d) M1 vs. GARSP; (e) M2 vs. GARSP.

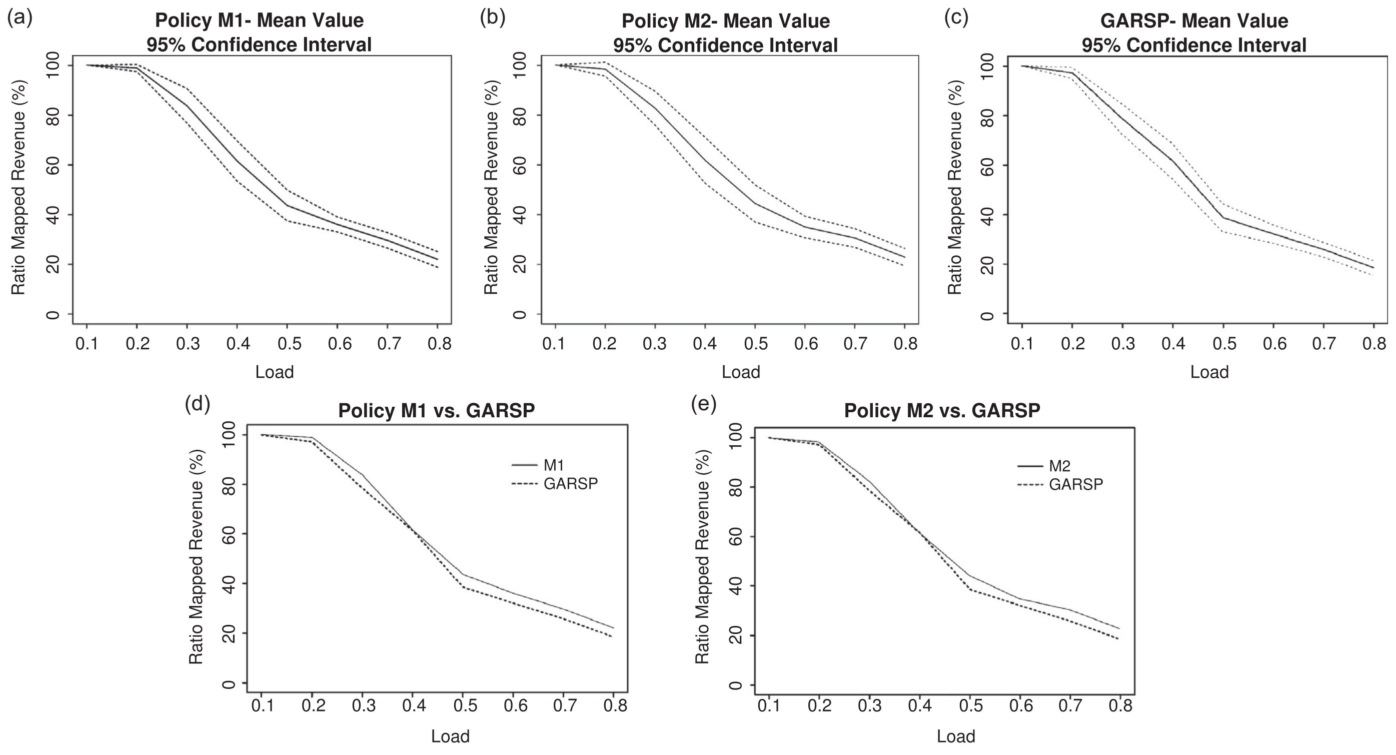

Fig. 10. Ratio mapped revenue mean values behavior for (a) M1, (b) M2 and (c) GARSP policies and comparisons: (d) M1 vs. GARSP; (e) M2 vs. GARSP.

either a constant behavior or slowly increasing as the load increases.

Figure 11 compares the C/R relationship performance of the adopted strategies. The obtained results are in agreement with what is expected and the paths algebra policies outperform the GASP strategy.

According to the whole set of results, it is important to realize that depending on the performance criterion that is chosen, the best strategy may not be the same for all loads but it is always a paths algebra strategy. The meaning of this result is twofold:

1. Even if the numerical results do not vary significantly, it is important to emphasize that, according to the best of our knowledge, the paths algebra sets the current limit of achievable performance.
2. Different paths algebra strategies can be exploited to achieve the best performance according to the chosen criterion.

In other words we may state that the paths algebra is a powerful and flexible strategy.

Such flexibility allows a detailed exploration of the solutions space and the identification of criteria that best suit the objectives of the InP, taking into account technical requirements to guarantee a given QoS. It is important to note that the exploration of the solutions space is feasible as far as processing time is concerned because the paths enumeration routine (that is the most time consuming) does not have to be processed. Paths enumeration can be made off-line and needs to be processed again only when there is a change in the physical network.

In this work, we have chosen to use number of hops, spare bandwidth, spare CPU and total spare bandwidth as metrics. Spare bandwidth and spare CPU are synthesized using the min() function, that is a non-linear function. Other non-linear metrics as packet loss ratio and reliability could be also employed.

In summary, we may state that:

the paths algebra does not impose any restriction on the type and number of metrics that are used. Right on the contrary, linear and non-linear metrics can be used together; any combination of metrics can also be employed.

The paths algebra empowers either the InP or the SP to look for the best strategy that may satisfy the needs of end customer and at the same time maximize the financial performance of the system. Strategies can be changed whenever necessary to adapt to new demand conditions to keep the system operation close to the best achievable performance.

In this work, we show the results only in terms of the VNR acceptance ratio, mapped revenue ratio and cost/revenue (C/R) relationship. The complete set of results obviously records the mapped paths. However, the paths algebra orders all eligible paths and, therefore, not only the best solution is available but all possible solutions can be accessed. This is a very valuable feature to implement recovery techniques in case of failure. Not only an alternate path for fast recovery is known by the system but also any performance degradation can be anticipated and dealt with.

The feature that all eligible paths are enumerated and ordered also shows the potential of the technique to implement multipath

Table 10
Comparison of two possible mappings.

Mapping	1	2
Node mapping	A→1; B→2	A→4; B→5
Path	1;2	4; 3; 5
Cost	CPU(1) + CPU(2) + BW(1–2)	CPU(4) + CPU(3) + CPU(5) + BW(4–3) + BW(3–5)
	=20 + 30 + 100	=20 + 0 + 30 + 100 + 100
	=150	=250
Revenue	CPU(A) + CPU(B) + BW(A–B)	CPU(A) + CPU(B) + BW(A–B)
	=150	=150
C/R	150/150=1	250/150=1.67

Note: CPU(3)=0 because the cost of hidden nodes is not being considered.

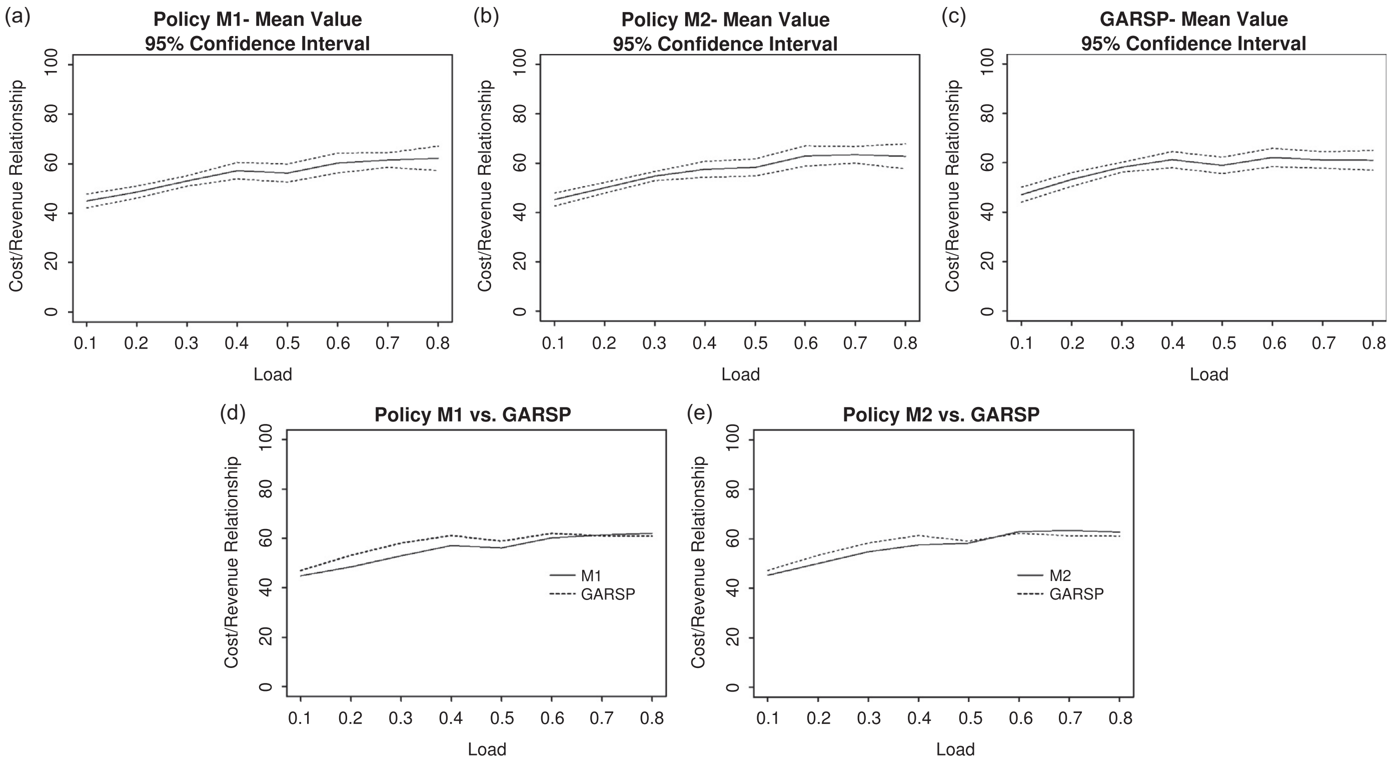

Fig. 11. Cost revenue (C/R) mean values behavior for (a) M1, (b) M2 and (c) GARSP policies and comparisons: (d) M1 vs. GARSP; (e) M2 vs. GARSP.

solutions. Different multipath strategies may be envisaged, as for example:

- always look for a multipath solution to promote load balancing;
- look for a multipath solution immediately after a single path solution has not been found for a given VNR;
- look for a multipath solution after the completion of a single path phase in order to map the set of discarded VNRs.

Multipath strategies are out of the scope of this work and will be investigated in the near future.

6. Conclusions and future work

In this work we have presented a novel strategy to solve the link mapping stage of the VNE problem based on the paths algebra framework.

The paths algebra technique has been originally developed to solve the multi-constraint routing problem. It is a powerful technique that can work simultaneously with any number of linear and non-linear parameters permitting the optimization of technical and business related criteria. Besides, it provides great flexibility allowing to introduce any number of network constraints to the virtual link mapping stage and to provide the facility to organize the complete set of substrate paths between any pair of nodes in mono and multi-constraint environments.

Several parameters can be used to compare the results obtained by the different algorithms. In this work we have chosen to use: VNR acceptance ratio, mapped revenue ratio and cost/revenue (C/R) relationship. It has been shown that for all offered loads the paths algebra strategies produce similar or better results when compared with shortest path-based strategies both with the same node mapping approach.

The process of finding all the possible paths between any pair of nodes shows an increasing complexity for big networks. To overcome this limitation, the computational complexity has been kept low for all practical purposes by limiting the maximum length of the paths to be considered. This procedure produces an important reduction on the number of paths to be discovered and processed. This modification has been introduced into the multi-constraint routing algorithms, providing a control over the number of operations performed by the algorithm and enabling the possibility to work with large networks. The results obtained for the VNE mapping show that, if there is any impact in the embedding, it is of negligible practical importance.

We conclude that the paths algebra is a powerful and flexible strategy. Such flexibility allows a detailed exploration of the solutions space and the identification of criteria that best suit the objectives of the infrastructure provider, taking into account technical requirements to guarantee a given QoS. In addition, the paths algebra does not impose any restriction on the type and number of metrics that are used. Right on the contrary, linear and non-linear metrics can be used together; any combination of metrics can also be employed.

Besides, paths algebra orders all eligible paths and, therefore, not only the best solution is available but all possible solutions can be accessed. This is a very valuable feature to implement additional strategies such as survivability or planning.

New VNE proposals show that the performance of the embedding is improved when the node and link mapping stage are performed in a coordinated way, i.e., nodes are mapped in a way that optimizes the link mapping stage. In future work we plan, taking advantage of paths algebra, to improve our approach by including a coordinated node mapping stage that could provide a two stages-based VNE solution or even a solution performed in one stage.

In addition, we plan to investigate backtracking strategies to act when a VNR cannot be satisfied due to bottlenecks present in the substrate network. The investigation of multi-path strategies based on paths algebra can also be an exciting branch for future research.

Acknowledgments

This work has been supported by the Spanish Government, MICINN, under research Grant TIN2010-20136-C03.

References

Anderson T, Peterson L, Shenker S, Turner J. Overcoming the internet impasse through virtualization. Computer 2005;38:34–41.
Bejerano Y, Breitbart Y, Orda A, Rastogi R, Sprintson A. Algorithms for computing qos paths with restoration. IEEE/ACM Transactions on Networking 2005;13: 648–61.
Bhardwaj S, Jain L, Jain S. Cloud computing: a study of infrastructure as a service (iaas). International Journal of Engineering and Information Technology 2010;2:60–3.
Botero J, Hesselbach X, Fischer A, Meer H. Optimal mapping of virtual networks with hidden hops. Telecommunication Systems 2012;51:273–82.
Butt NF, Chowdhury NMMK, Boutaba R. Topology-awareness and reoptimization mechanism for virtual network embedding. In: Networking, 2010. p. 27–39.
Carrè B. Graphs and networks, Oxford applied mathematics and computing science series.Walton Street, Oxford: Oxford University Press; 1979.
Chartrand G. Introductory graph theory.New York: Dover Publications, Inc; 1985.
Cheng X, Su S, Zhang Z, Wang H, Yang F, Luo Y, et al. Virtual network embedding through topology-aware node ranking. SIGCOMM—Computer Communication Review 2011;41:38–47.
Chowdhury NMK, Boutaba R. A survey of network virtualization. Computer Networks 2010;54:862–76.
Chowdhury N, Rahman M, Boutaba R. Virtual network embedding with coordinated node and link mapping. In: INFOCOM 2009. Brazil: IEEE; 2009.
Chowdhury M, Samuel F, Boutaba R. Polyvine: policy-based virtual network embedding across multiple domains. In: Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures, VISA '10, India, 2010.
Chowdhury M, Rahman MR, Boutaba R. Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. IEEE/ACM Transactions on Networking 2012;20:206–19.
Cui Y, Xu K, Wu J. Adjustable multi-constrained routing with a novel evaluation method. In: Conference Proceedings of the 2003 IEEE international on performance, computing, and communications conference, 2003, China, 2003.
Fajjari I, Aitsaadi N, Pujolle G, Zimmermann H. Vne-ac: virtual network embedding algorithm based on ant colony metaheuristic. In: 2011 IEEE international conference on communications (ICC), Japan, 2011.
Feamster N, Gao L, Rexford J. How to lease the internet in your spare time. SIGCOMM—Computer Communication Review 2007;37:61–4.
Fischer A, Botero JF, Duelli M, Schlosser D, Hesselbach X, De Meer H. ALEVIN—a framework to develop, compare, and analyze virtual network embedding algorithms. Electronic Communications of the EASST 2011;37:1–12.
Fujita N, Iwata A. Adaptive and efficient multiple path pre-computation for qos routing protocols. In: Global telecommunications conference, 2001. GLOBECOM '01. USA: IEEE; 2001.
Furini M, Towsley DF. Real time traffic transmissions over the internet. IEEE Transactions on Multimedia 2001;3:33–40.
Gouda MG, Schneider M. Maximizable routing metrics. IEEE/ACM Transactions on Networking 2003;11:663–75.
Herman WP. Formulação algèbrica para a modelagem de algoritmos de roteamento multi-restritivo hop-by-hop. PhD thesis. São Paulo: Escola Politècnica da USP; 2008.
Herman WP, Amazonas JR. Hop-by-hop routing convergence analysis based on paths algebra. In: Proceedings of the 2007 electronics, robotics and automotive mechanics conference—CERMA 2007, Mexico, 2007.
Houidi I, Louati W, Ameur WB, Zeghlache D. Virtual network provisioning across multiple substrate networks. Computer Networks 2011;55:1011–23 (Special issue on Architectures and Protocols for the Future Internet).
Jaggard AD, Ramachandran V. Relating two formal models of path-vector routing. In: INFOCOM 2005—24th annual joint conference of the IEEE computer and communications societies, USA, 2005.
Jukan A, Franzl G. Path selection methods with multiple constraints in service-guaranteed WDM networks. IEEE/ACM Transactions on Networking 2004;12:59–72.
Kuipers F, Van Mieghem P, Korkmaz T, Krunz M. An overview of constraint-based path selection algorithms for qos routing. IEEE Communications Magazine 2002;40:50–5.

Lagoa C, Che H, Movsichoff B. Adaptive control algorithms for decentralized optimal traffic engineering in the internet. IEEE/ACM Transactions on Networking 2004;12:415–28.

Lischka J, Karl H. 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, Spain, 2009.

Lu J, Turner J. Efficient mapping of virtual networks onto a shared substrate. Technical Report. Washington University in St. Louis, 2006.

Mieghem PV, Kuipers FA. Concepts of exact QoS routing algorithms. IEEE/ACM Transactions on Networking 2004;12:415–28.

Mieghem PV, Kuipers FA. Conditions that impact the complexity of QoS routing. IEEE/ACM Transactions on Networking 2005;13:717–30.

Miyamura T, Kurimoto T, Aoki M. Enhancing the network scalability of link-state routing protocols by reducing their flooding overhead. In: Workshop on high performance switching and routing, HPSR, Italy, 2003.

Papadimitriou P, Maennel O, Greenhalgh A, Feldmann A, Mathy L. Implementing network virtualization for a future internet. In: 20th ITC specialist seminar on network virtualization ‘‘concept and performance aspects’’, Vietnam, 2009.

Pei D, Zhao X, Massey D, Zhang L. A study of bgp path vector route looping behavior. In: 24th international conference on distributed computing systems, 2004. Proceedings, Japan, 2004.

Pioro M, Medhi D. Routing, flow, and capacity design in communication and computer networks. Morgan Kaufmann Publishers; 2004.

Quoitin B, et al. Interdomain traffic engineering with BGP. IEEE Communications Magazine 2003;41:122–8.

Shen L, Xu M, Xu K, Cui Y, Zhao Y. Simple quality-of-service path first protocol and modeling analysis. In: IEEE international conference on communications, China, 2004.

Sobrinho J. On the convergence of path vector routing protocols. In: 2001 IEEE workshop on high performance switching and routing, USA, 2001. p. 292–6.

Sobrinho JL., Network routing with path vector protocols: theory and applications. In: Proceedings of the 2003 conference on applications, technologies, architectures, and protocols for computer communications, SIGCOMM '03, Germany, 2003.

Sobrinho JL. An algebraic theory of dynamic network routing. IEEE/ACM Transactions on Networking 2005;13(5):1160–73.

Su P, Gellman M. Using adaptive routing to achieve quality of service. Performance Evaluation 2004;57:105–19.

Tutschku K, Zinner T, Nakao A, Tran-Gia P. Network virtualization: implementation steps towards the future Internet. In: KiVS 2009, Kassel, 2009.

VNREAL, ALEVIN—algorithms for embedding virtual networks 〈http://alevin.sf.net〉, May 2011.

Wattenhofer R, Venkatachary S. The impact of internet policy and topology on delayed routing convergence. In: INFOCOM 2001—20th annual joint conference of the IEEE computer and communications societies, USA, 2001.

Wolsey LA. Integer programming.New York: John Wiley & Sons; 1998.

Xiao X. Providing quality of service in the internet. PhD thesis. Michigan State University; 2000.

Yu M, Yi Y, Rexford J, Chiang M. Rethinking virtual network embedding: substrate support for path splitting and migration. ACM SIGCOMM Computer Communication Review 2008;38:17–29.

Zhu Y, Ammar M. Algorithms for assigning substrate network resources to virtual network components. In: Proceedings of the IEEE INFOCOM, 2006. p. 2812–23.