Journal of Network and Computer Applications 124 (2018) 121–136

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

Online power aware coordinated virtual network embedding with 5G delay constraint

Khaled Hejja *, Xavier Hesselbach

Dept. Networks Engineering, Universitat Politècnica de Catalunya, C/ Jordi Girona, 1-3, Edif.C3, Campus Nord, 08034, Barcelona, Spain

ARTICLE INFO

Keywords:
Virtual network embedding
Core network virtualization
Power consumption
Coordinated embedding
Delay

ABSTRACT

Solving virtual network embedding problem with delay constraint is a key challenge to realize network virtualization for current and future 5G core networks. It is an NP-Hard problem, composed of two sub-problems, one for virtual node embedding, and another one for virtual edges embedding, usually solved separately or with a certain level of coordination, which in general could result on rejecting some virtualization requests. Therefore, the main contributions of this paper focused on introducing an online power aware algorithm to solve the virtual network embedding problem using less resources and less power consumption, while considering end-to-end delay as a main embedding constraint. The new algorithm minimizes the overall power of the physical network through efficiently maximizing the utilization of the active infrastructure resources and putting into sleeping mode all non-active ones. Evaluations of the proposed algorithm conducted against the state of art algorithms, and simulation results showed that, when end-to-end delay was not included the proposed online algorithm managed to reduce the total power consumption of the physical network by 23% lower than the online energy aware with dynamic demands VNE algorithm, EAD-VNE. However, when end-to-end delay was included, it significantly influenced the whole embedding process and resulted on reducing the average acceptance ratios by 16% compared to the cases without delay.

1. Introduction

Networks virtualization is an integral component of the current and future 5G core networks, offering network operators opportunities to consolidate their equipments into standardized high volume components. This is reflected by efficiently utilizing the physical network resources, through sharing them among several virtual networks (VN), as well as providing more flexibility to manage, expand, and shrink the physical network according to the VNs' characteristics (3GPP TR 28.801, 2017; ITU-T Focus Group, 2017).

In general, network virtualization is realized by allocating sufficient physical resources to satisfy the requirements of a virtual network request (VNR), on top of a substrate network (SN) that has limited residual capacities. This process is known as virtual network embedding problem (VNE) (Fischer et al., 2013), which could be solved for offline scenario, where all VNRs are known in advance, or for online scenario, where VNRs arrive the SN on real time bases and each have specific lifetime. Nevertheless, the VNE problem is a well known NP-Hard problem and can not be solved in polynomial time (Chowdhury et al., 2012).

To better explain the virtual network embedding problem and how it could save the total power consumption in the substrate network, Fig. 1 shows a substrate network of nine nodes and twelve edges, receiving two virtual network requests to be embedded, VNR-1 of four nodes and four edges, and VNR-2 of three nodes and two edges. To realize the embeddings for virtual nodes and edges of VNR-1 and VNR-2, the VNE algorithm will search for candidate substrate nodes and edges that have enough residual capacities to host their demands, while maintaining similar connectivity as in the virtual requests. For VNR-1 in Fig. 1, virtual nodes a, b, c, and d can be hosted on substrate nodes 1, 3, 2, and 4, and the virtual edges a-b, a-c, b-d, and c-d can be hosted on substrate edges 1–3, 1–2, 3–4, and 2–4 subsequently. Accordingly, the VNE algorithm will embed VNR-1 on the candidate substrate resources, and reserve the required capacities as requested by the virtual nodes and edges. The same procedure repeats to embed VNR-2, where virtual nodes e, f, and g will be embedded on substrate nodes 5, 7, 9, and virtual edges e-f and f-g will be embedded on substrate edges 5–7 and 7–9 respectively. Once all virtual network requests were allocated, the remaining nodes, 6 and 8, that do not host any virtual resources could

* Corresponding author.
E-mail addresses: Khaled.Hejja@upc.edu (K. Hejja), Xavier.Hesselbach@upc.edu (X. Hesselbach).

https://doi.org/10.1016/j.jnca.2018.10.005
Received 13 February 2018; Received in revised form 26 August 2018; Accepted 3 October 2018
Available online 6 October 2018

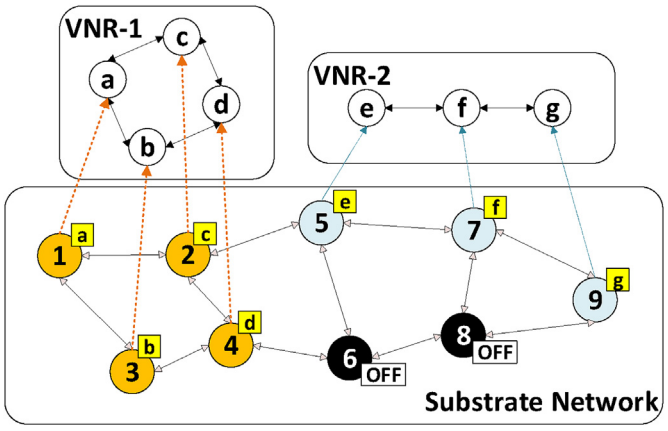

Fig. 1. An illustrative diagram showing a substrate network of nine nodes and twelve edges receiving two virtual network requests to be embedded, VNR-1 of four nodes and four edges, and VNR-2 of three nodes and two edges.

be put into sleeping mode to save the total power consumptions in the substrate network, and to minimize total costs.

Classic approach to handle the VNE problem is usually conducted by dividing it into two subproblems with no coordination between them. The first one deals with allocating virtual nodes onto physical nodes, which is known as virtual node mapping (VNM) subproblem, and the other one, is virtual edge mapping (VEM) subproblem, which embeds virtual edges onto physical paths, connecting the corresponding nodes hosting the virtual nodes in the physical network (Yu et al., 2008). Along such process, VNE usually trades off between minimizing embedding costs, utilizing less SN resources, and maximizing revenues through accepting as much as possible VNRs, while maintaining acceptable quality of services (QoS). However, the fact that the two subproblems of the VNE are solved without coordination, based on finding an acceptable embeddings for virtual nodes separate from embedding the virtual edges, would most probably result on rejecting some virtual demands, or including additional hidden SN resources (Fischer et al., 2013), accordingly, increasing embedding costs and consuming more power.

Therefore, another strategy was developed, performing both VNM and VEM as two separate subproblems, but with some coordination between their solutions, where VNM is performed according to predefined VEM constraints to guide allocating the virtual nodes (Chowdhury et al., 2012). However, even through this provides a sort of coordination between VNM with VEM, still, virtual nodes could be embedded at physical nodes that are farther away from each other, which implies enforcing edges to be mapped at longer physical paths, as well as resulting on additional costs, power consumption, and maybe rejecting some VNRs. Furthermore, regardless of the used strategy, VNE used to be constrained by virtual node's processing capacities, and edge's bandwidth demands, but occasionally considering power consumption, or adding delay as an additional constraint. Accordingly, the lack of considering more constraints throughout the VNE process, most likely would result on degrading the overall quality of the whole embedding process, especially when considering network virtualization for 5G applications, which are sensitive to the amount of delay in the network for example (5G Americas, 2017).

Consequently, the main motivation of this paper is to introduce a new efficient embedding algorithm to solve the VNE grand problem, on real-time bases and including end-to-end delay constraint, while minimizing the overall power consumption in the substrate network. The proposed online power aware and fully coordinated VNE algorithm, denoted by (OPaCoVNE), allocates the virtual nodes and the virtual edges during one stage according to more realistic constraints, such as nodes' processing power, links' bandwidth and end-to-end delay. Moreover, the benefits of the new proposed algorithm will be reflected on realizing real time virtual network requests for 5G applications that are sensitive to delay, in addition to solving the VNE problem for large networks, using less resources, and generating lower costs and higher revenues.

The core idea of OPaCoVNE can be summarized as follows, for every virtual network request VNR^r number *r*, OPaCoVNE will reconstruct it into a collection of pairs, in which each of them will be composed of two virtual nodes and their edge, then the demands of each pair will be listed in a set format, to be called (segment), representing all parameters of the two virtual nodes and their edge, including node's locations and their *CPUs* processing powers, in addition to their edge's demanded bandwidth *BW* and delay. As a result of that, OPaCoVNE will consider all pairs of any VNR as a set of segments to be handled collectively. To perform the embedding process, OPaCoVNE will afterwards select a similar number of substrate pairs, each composed of two nodes and one edge only, which comply with the demanded locations of the virtual pair's nodes and edge, then it will reformulate their resources into segment format similar to the structure of the virtual segment format, representing the substrate pairs' parameters. Ultimately, the algorithm will compare each parameter from the virtual segment to its corresponding parameter in the substrate segment, and if all comparisons were true, OPaCoVNE will embed the virtual nodes and their edge from each virtual pair onto its substrate counterpart.

In light of that, the proposed segmentation approach allowed OPaCoVNE to coordinate virtual nodes and edges' embeddings together, on a corresponding substrate nodes and edges. This will guarantee high acceptance ratios, and efficiently utilizes the limited capacities in the substrate network without any additional recourses, therefore, resulting on minimizing the total power consumption in the substrate network. Subsequently, OPaCoVNE's use of the segmentation approached, differentiated it from the VNE's two steps approach adopted by literature, which used to apply greedy algorithm to embed all virtual nodes first, then run the shortest path algorithm to embed the virtual edges as a second step, with partial or no coordination with the nodes' embedding step.

General performance and evaluations of OPaCoVNE algorithm were conducted against one of the most referenced online VNE algorithms in literature, the D-ViNE developed by Chowdhury et al. (2012), and the power aware performance of OPaCoVNE was also compared against the online power aware VNE algorithm, EAD-VNE developed by Zhang et al. (2015). Finally, this paper provides additional analysis regarding the impacts on OPaCoVNE's performance when end-to-end delay was used, in addition to analyzing power consumption, acceptance ratios and processing time of the algorithm, by varying number, lifetime, and size of the embedded VNRs, as well as varying number of edges in the SN.

Main contributions:

1. As a main contribution by this paper, impacts of end-to-end delay on the performance of the suggested online algorithm, OPaCoVNE, were deeply analyzed, representing direct application for virtualization in future 5G networks.
2. To minimize the total power consumption in the whole SN, segmentation approach was proposed to guarantee coordinating the embeddings of the virtual nodes and edges, while utilizing the least substrate resources to the minimum.
3. Evaluations for the acceptance ratio, cost, and utilizations of the SN nodes and edges of the proposed online algorithm, were conducted against the most referenced online algorithm by Chowdhury et al. (2012), and the power consumption performance was compared against the online power aware work of Zhongbao et al. (2015).
4. Additional evaluations for the proposed online algorithm, OPaCoVNE, were also introduced, showing the impacts of end-end-delay, by varying number and size of the embedded VNRs, as well as varying their lifetimes.

Rest of the paper is organized as follows: Section 2 provides related work. Segments definition and formulation are presented in section 3, followed by an overview of the online scenario and its main objectives in 4. Section 5 will introduce the system models, problem formulation, explanation of the proposed online algorithm, and discussion for its evaluation results will be presented in section 6. Finally, section 7 concludes the paper and highlights some future work.

2. Related work

With regards to the coordinated VNE, the most referenced approach is the one suggested by Chowdhury et al. (2012), which introduced a set of algorithms to correlate between node and edge embedding problems when solving the VNE grand problem. They embedded the virtual nodes onto SN nodes based on their residual capacities, and coordinated the edges embeddings using the multi-commodity flow algorithm. However, since nodes were embedded first then edges afterwards, longer paths could be used, which would result on adding additional costs. In Botero et al. (2012a), the authors developed a greedy algorithm using the shortest path, to embed virtual demands of both nodes and edges together. They allowed hidden nodes in these shortest paths, which may lead to consuming more resources.

Another methodology to solve the embedding of virtual demands was suggested by Botero et al. (2013). They used a strategy using the mathematical frameworks of path algebra, which finds all eligible paths between each pair of nodes in the SN to embed a virtual network request using any type of constraints. A similar recent work using the same mathematical framework was proposed by Nia et al. (2017), they ordered the SN paths using an algorithm based on path algebra mathematics, which ranks and choses SN nodes to host the virtual nodes, then, they search for the best SN path connecting the already chosen nodes if it exists, by selecting the path with highest order based on some determined metrics.

Moreover, a VNE heuristic based on greedy approach was proposed by Ogino et al. (2017), which prioritizes the virtual edges assignment, rather than the virtual nodes, suggesting that substrate resources shared among multiple virtual networks is more likely to occur on the substrate edge bandwidth. They used the minimum-cost route algorithm, to compute the optimum substrate path for each virtual edge. However, their methodology could allow for a possibility to select longer paths containing non requested SN resources.

Recently, a study of VNE problem under maximum latency constraint was also developed by Bianchi and Presti (2017), to determine a set of substrate network nodes and edges that could satisfy the demands of virtual network's end-to-end delay. Their algorithm used Markov chain reward metrics, coordinating virtual nodes' embedding problem with virtual edges' embedding as two problems. However, giving that they used shortest path algorithm based on delay constraint, and applying a proximity metric to avoid including other non utilized nodes in the selected path, that may offer closer path in terms of network hops, but would include additional nodes that were not utilized, causing additional costs.

The authors in Su et al. (2014) proposed to maximize the accepted VNRs while minimizing the power cost of the whole system. They embed VN nodes on SN nodes that has lowest electricity price, then put other nodes that has no load into sleeping mode. Moreover, Triki et al. (2015) developed an embedding algorithm that embeds a subset of VNRs into a subset of cleanest SN resources while satisfying the VNR constraints. They constrained the VNE process by introducing edge delay, packet loss, used power source, VNR priority and location. While in Nonde et al. (2015), the authors designed an MILP and a real time heuristic algorithm that, considers granular power consumption of all devices in the physical network. They tried to consolidate nodes' embeddings, by filling the ones with the least residual capacity before switching on others.

Moreover, in Chen et al. (2016), the authors considered power efficient VNE using feedback control approach, performing the embeddings on a smaller set of SN resources. A limited mappable area consisting of a selection of candidate nodes was located first, then they checked if VN embedding was successful, if not, then a feedback control approach is triggered to search for a wider mappable area. A modified VNE algorithm was presented by Botero et al. (2012b), which prefers SN nodes consuming less power and selects edges in a power efficient path. In Botero and Hesselbach (2013), they developed a scalable power aware reconfiguration heuristic approach, based on embedding cost and load balancing. The heuristic considers a set of embedded VNRs as input, to perform a power efficient relocation of resources without impacting the acceptance ratio.

In addition to that, a power aware algorithm using migrations was proposed by Ghazisaeedi and Huang (2017), which re-embeds virtual edges according to their off-peak traffic demands, by defining stress rate for the substrate edges, then after reallocating the virtual edges they put into sleeping mode the remaining least stressed edges. However, they did not re-embed the virtual nodes, but they make sure by applying shortest path algorithm of Dijkstra, that the selected substrate path used to migrate the virtual edges is also connecting the terminal nodes of the virtual edge. Lastly, Hou et al. (2016) applied load balancing and power efficient VNE heuristic, by using two embedding phases, one to embed the virtual nodes, and the other one to embed the virtual edges. They embed on physical nodes that are near to each other by using Dijkstra shortest path algorithm. By this way they consolidated the traffic from least utilized nodes and put into sleeping mode their connecting edges.

3. Segments for full coordinated solution of the VNE problem

This paper proposes a new solution for the two subproblems of the virtual network embedding problem, by converting the structure of any virtual network request into a collection of pairs of virtual nodes and their edge, where the demands of each pair will be listed together in a set format called segment, and converting the resources of an exact similar number of substrate network pairs into segment format as well. The motivation behind using the segmentation approach, is to facilitate allocating all virtual pairs belonging to a specific VNR, on the corresponding substrate pairs that have enough resources to host the demands of the virtual pairs' nodes and edges, together in full coordination, without using any additional hidden substrate resources, which guarantees consuming the least total power in the whole network along the virtualization process.

The virtual pairs' u and v segment is denoted by Seg^r_{uv}, and substrate pairs' i and j segment is denoted by Seg_{ij}. The structure of both segments must be similar in format, which means that the number of parameters representing the virtual nodes and edges in the virtual segment Seg^r_{uv}, are exactly similar to the number of parameters representing the substrate nodes and edges in the substrate segment Seg_{ij}, regardless the values of their parameters.

The proposed segmentation format guarantees full coordinated embedding of the virtual nodes and edges, since segments formulation provides direct way to check one-to-one, if each element in the substrate pair's segment has enough resources to host the demands of its counterpart from the virtual pair's segment, precisely, by comparing the first parameter in the virtual segment to the first parameter in the substrate segment, the second to the second, and so on for all the remaining parameters. If the results of 'ALL' checks are true, that is, each virtual node demand found a substrate node to host it, 'AND', each virtual edge demand found a substrate edge to host it, the embedding of the virtual pair's nodes and edges will be realized, together in full coordination onto the corresponding substrate pair.

To clarify what is meant by segments and how they guarantee fully coordinated embedding, segment definition, and segment formulations for a specific substrate path and for a VNR are presented as follows:

3.1. General definition of the basic segment

For any path P^{ij} belonging to a physical network or is part of a virtual network request, consisting of two nodes i and j, connected by a direct edge (i,j), and does not include any intermediate or hidden nodes, P^{ij} segment, denoted by Seg_{ij} is defined as a list including all parameters of the two nodes and the edge, grouped in a set format as shown in Eq. (1).

For example, P^{ij} could be represented by the following specific nodes' parameters, namely, nodes' locations denoted by loc_i and loc_j, processing power capacities cpu_i and cpu_j, and parameters of their connecting edge are bandwidth capacity bw_{ij} and delay d_{ij}. Consequently, the segment defining path P^{ij} can be written in set format as follows:

$$Seg_{ij} = \{loc_i, loc_j, cpu_i, cpu_j, bw_{ij}, d_{ij}\} \quad (1)$$

However, in general any substrate or virtual directed graph can be reconstructed of one or multiple paths, where each path could be composed of multiples of the basic segment above representing the pairs formulating the corresponding paths. Therefore, to generalize the concept of the basic segment in Eq. (1) on any graph, let Set^r be a set of segments listing all demands of the virtual nodes and edges in a VNR^r graph, and let Set^S be a set of segments listing all resources of the candidate physical nodes and edges in a substrate graph to host the virtual nodes and edges in the VNR^r.

The formulation of both segments is clarified in the following subsections, starting by the VNR segment:

3.2. VNR set of segments formulation (Set^r)

For a virtual network request VNR^r, consisting of a collection of pairs of virtual nodes associated with *loc* and *CPU* demands, and virtual edges associated with *BW* and delay demands, Eq. (1) can be generalized to list all these parameters together as shown in Eq. (2) as follows:

$$Set^r = \{Seg^r_{uo}, .., Seg^r_{wx}, .., Seg^r_{pv}\} \quad (2)$$

Where:

$$Seg^r_{uo} = \{loc^r_u, loc^r_o, cpu^r_u, cpu^r_o, bw^r_{uo}, d^r_{uo}\}$$

$$Seg^r_{wx} = \{loc^r_w, loc^r_x, cpu^r_w, cpu^r_x, bw^r_{wx}, d^r_{wx}\}$$

$$Seg^r_{pv} = \{loc^r_p, loc^r_v, cpu^r_p, cpu^r_v, bw^r_{pv}, d^r_{pv}\}$$

u, v, o, p, w, x are virtual nodes $\in$ VNR^r*graph*

An illustrative application of Eq. (2) is shown in Fig. 2a, which presents two VNRs to be embedded, a line graph VNR^{r1} and a directed cycle graph VNR^{r2}. For example, VNR^{r1} is composed of three virtual nodes and two virtual edges forming two pairs. The first virtual pair includes virtual nodes 1.1 and 1.2, and virtual edge (1.1, 1.2), while the second virtual pair includes virtual nodes 1.2 and 1.3, and virtual edge (1.2, 1.3). Notice that the first virtual node location, $loc^{r1}_{1.1}$, demands being located at the location of SN node *a*, while location of second virtual node $loc^{r1}_{1.2}$ at SN node *b*, and $loc^{r1}_{1.3}$ at SN node *e*. At the top of Fig. 2b, the formulation of Eq. (2) was applied to construct the set of segments, Set^{r1}, representing VNR^{r1}, listing all demanded *locs*, *cpus*, *bws*, and the overall end-to-end delay. Accordingly, Set^{r1} is formulated as follows:

$$Set^{r1} = \{Seg^{r1}_{1.1,1.2}, Seg^{r1}_{1.2,1.3}\}$$

Where:

$$Seg^{r1}_{1.1,1.2} = \{loc^{r1}_{1.1}, loc^{r1}_{1.2}, cpu^{r1}_{1.1}, cpu^{r1}_{1.2}, bw^{r1}_{1.1,1.2}, d^{r1}_{1.1,1.2}\}$$

$$Seg^{r1}_{1.2,1.3} = \{loc^{r1}_{1.2}, loc^{r1}_{1.3}, cpu^{r1}_{1.2}, cpu^{r1}_{1.3}, bw^{r1}_{1.2,1.3}, d^{r1}_{1.2,1.3}\}$$

In this way, the demands of VNR^{r1} are translated from a line graph topology, into a set of segments representing the demands of the two pairs formulating VNR^{r1}. The same procedure can be repeated for VNR^{r2} as shown at the top of Fig. 2c.

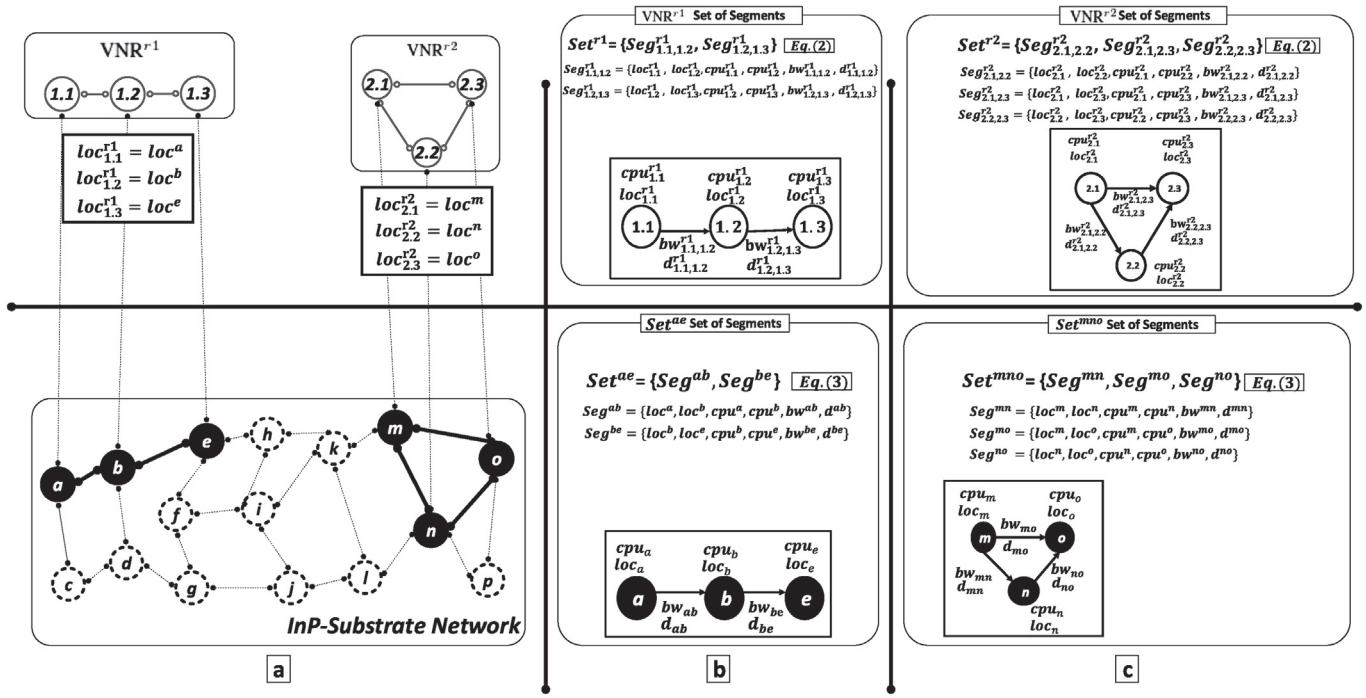

Fig. 2. Example about constructing the physical path segment, Set^S, and VNR segment, Set^r. In a, two VNRs need to be embedded, a line graph VNR^{r1} and a directed cycle graph VNR^{r2}. In the upper part of b and c, the construction of the set of segments belonging to VNR^{r1} and VNR^{r2} is shown, and their corresponding substrate segments are shown in the lower parts of b and c.

3.3. Segment formulation for a substrate graph (Set^S)

For any substrate graph consisting of similar pairs of nodes and edges as the VNR's pairs, without any hidden hopes or edges, reformulate its resources into a set of segments, listing the substrate pairs' *loc*, *CPU*, *BW*, and delay values together as shown in Eq. (3):

$$Set^S = \{Seg_{sn}, .., Seg_{kl}, .., Seg_{md}\} \quad (3)$$

Where:

$$Seg_{sn} = \{loc_s, loc_n, cpu_s, cpu_n, bw_{sn}, d_{sn}\}$$

$$Seg_{kl} = \{loc_k, loc_l, cpu_k, cpu_l, bw_{kl}, d_{kl}\}$$

$$Seg_{md} = \{loc_m, loc_d, cpu_m, cpu_d, bw_{md}, d_{md}\}$$

s, d, k, l, n, m are physical nodes $\in$ *substrate graph*

To complete the above example, the bottom of Fig. 2b illustrates a direct application on how to use Eq. (3). Since the demanded virtual nodes' locations are the locations of the nodes on substrate path P^{ae}, then the set of segments representing P^{ae}, denoted by Set^{ae}, represents a typical similar structure to that of Set^{r1}. Similar in this example means, since VNR^{r1} topology is a line graph, consisting of two pairs, with specific virtual nodes' locations, then the selected substrate graph topology must be a line graph, of two pairs fulfilling the demands of the virtual nodes' locations. Accordingly, applying Eq. (3) to formulate the set of segments, Set^{ae}, representing substrate graph P^{ae} can be written as follows:

$$Set^{ae} = \{Seg_{ab}, Seg_{be}\}$$

Where:

$$Seg_{ab} = \{loc_a, loc_b, cpu_a, cpu_b, bw_{ab}, d_{ab}\}$$

$$Seg_{be} = \{loc_b, loc_e, cpu_b, cpu_e, bw_{be}, d_{be}\}$$

In this way, substrate line graph path P^{ae} has been translated from a graph topology into segment format, fulfilling the virtual nodes' demanded locations and similar in structure to Set^{r1}. The same process can be followed to formulate the set of substrate segments to allocate VNR^{r2} as shown in Fig. 2c.

Now to embed all virtual nodes and edges from VNR^{r1} on the substrate path P^{ae}, in full coordination, each element in the two formulated set of segments, Set^{r1} and Set^{ae} will be directly compared one-to-one, which means for each parameter value in Set^{r1} representing a virtual node in VNR^{r1}, check if its counterpart in Set^{ae} can fulfill its demands, and at the same time, for each parameter value in Set^{r1} representing a virtual edge in VNR^{r1}, check if its counterpart in Set^{ae} has enough residual capacity to host it as well. This process can be achieved ***if and only if*** each comparison statement in the following (if-AND) conditions is true, as follows:

if $loc_a = loc^{r1}_{1.1}$ **AND**

if $loc_b = loc^{r1}_{1.2}$ **AND**

if $loc_e = loc^{r1}_{1.3}$ **AND**

if $cpu_a - cpu^{r1}_{1.1} \geq 0$ **AND**

if $cpu_b - cpu^{r1}_{1.2} \geq 0$ **AND**

if $cpu_e - cpu^{r1}_{1.33} \geq 0$ **AND**

if $bw_{ab} - bw^{r1}_{1.1,1.2} \geq 0$ **AND**

if $bw_{be} - bw^{r1}_{1.2,1.3} \geq 0$ **AND**

if $d_{ab} \leq d^{r1}_{1.1,1.2}$ **AND**

if $d_{be} \leq d^{r1}_{1.2,1.3}$

Accordingly, if all the above conditions were proven true, this guarantees that each virtual node and edge in VNR^{r1} will be hosted by their counterparts in substrate path P^{ae}. Therefore, segments formulation and the comparisons, made the embeddings of virtual nodes fully coordinated with the embeddings of the virtual edges, together on the same substrate path. The same comparison procedure can be followed to check if the demands of VNR^{r2} can be hosted by the directed cycle substrate graph P^{mno}.

4. Evaluation scenario

In real worlds of 5G networks, services, applications, and users interact with the infrastructure network instantly and on real-time. Therefore, in the context of networks' virtualization, the demands must be analyzed and allocated on the substrate network, online, utilizing the shared resources efficiently, adhering to the required service qualities as demanded, and at the same time keep updating the statue of the substrate network's resources regularly, in order to spontaneously evaluate the possibilities of allocating other virtualization demands when they arrive.

Based on that, the proposed algorithm by this paper, OPaCoVNE is designed for online scenario, and its main objective is to successfully embed the VNRs, on real time, while minimizing the overall power consumption in the whole substrate network considering end-to-end delay as a main embedding constraint. The algorithm handles the virtual network requests one-by-one, and keeps monitoring and updating the substrate network frequently, to free up more resources for future usage by new virtual requests.

Detailed online problem formulations, simulation and evaluation for the online scenario are presented in the following section.

5. Online problem formulation

Since virtual network embedding problems deals with making decisions, about an efficient utilizations for using the limited resources of the physical substrate network, the VNE problems were traditionally modeled as an optimization problem of objective function, and constrained by controlling conditions, matching the resources availability against the requirements, while utilizing the scarce physical resources. This is usually referred to as integer linear programming (ILP) problem, which when solved will have positive, integer, and linear decision variables in their final optimal solution (Bradley and Magnanti, 1977). However, optimal solution for VNE as an ILP problem, implies introducing binary constraints to connect one edge only for each node, then, mapping all virtual nodes and edges on their physical counterparts having enough resources to accommodate their demands. Accordingly, virtual edges associated with bandwidth constraints are usually treated as commodities between pairs of nodes, and therefore, embedding a virtual edge optimally is similar to finding an optimal flow for the commodity in any network model, which is an NP-hard problem and will take huge amount of time to solve (Fischer et al., 2013; Kleinberg and Tardos, 2009).

To formally introduce the VNE problem as an ILP, the following paragraphs will introduce network and power consumption models of the VNE, followed by the objective functions and its constraints. Table 1 summerizes all used notations for the formulation of the VNE problem. In addition to that, the proposed online algorithm that solves the VNE in polynomial times will be explained in details, including an illustrative example, its computational complexity, as well as introducing the algorithm's evaluation metrics.

Table 1
Notation.

Notation	Description.
G^S	Directed graph of the physical network.
N^S	Substrate network nodes.
E^S	Substrate network edges.
N^{Sub}	Subset of substrate network nodes.
E^{Sub}	Subset of substrate network edges.
loc_i	Location of node i.
cpu_i^a	Available CPU capacity at node i.
cpu_i	Consumed CPU capacity at node i
CPU_i	Maximum CPU capacity at node i.
U_i	CPU utilization of node i.
bw_{ij}^a	Available bandwidth capacity of (i,j).
bw_{ij}	Consumed bandwidth capacity of (i,j).
BW_{ij}	Maximum bandwidth capacity of (i,j).
f_{ij}	Throughput flows in (i,j).
d_{ij}^a	Current end-to-end delay in (i,j).
P^S	All basic paths in substrate network.
P^{ij}	Basic path of two nodes and one edge.
P^{sub}	Collection of basic paths.
G^V	Virtual network graph.
N^V	Virtual nodes of virtual network graph.
E^V	Virtual edges of virtual network graph.
VNR^r	Virtual network request r.
R	Total number of VNRs.
t, t_a^r, t_e^r	VNR^r duration, arrival, and expiry times.
loc_u	Preferred location to host virtual node u.
cpu_u^r	Demanded CPU capacity by u.
bw_{uv}^r	Demanded bandwidth capacity by (u,v).
d_{uv}^r	Demanded end-to-end delay in (u,v).
PC_i	Power consumption of i.
PC_i^{Busy}	Maximum power of node i.
PC_i^{idle}	Idle power of node i.
Seg_{ij}	Substrate pair i and j segment.
Seg_{uv}^r	Virtual pair u and v segment.
Adj^{sub}	Substrate set of segments adjacency matrix.
Adj^r	VNR adjacency matrix.
x_i^{ur}	1 if i is hosting virtual node u.

5.1. Substrate network model

The physical network $G^S = (N^S, E^S)$ is modeled as a weighted directed graph, where i and $j \in N^S$ are SN nodes, and $(i,j) \in E^S$ is an edge connecting nodes i and j. Each node $i \in N^S$ is associated with loc_i denoting the node's location, cpu_i^a representing current available CPU capacity, cpu_i for consumed CPU capacity, and CPU_i as the maximum CPU capacity of node i. Each substrate edge (i,j) is associated with bw_{ij}^a, bw_{ij}, and BW_{ij} representing available, consumed, and maximum bandwidth capacities. It is assumed that the physical network is well established and stable, therefore it would be possible to extract all the physical paths in advance. Accordingly, $P^S = \{(i,j)\}$ represents the set of all directed basic paths in the whole SN, where each basic path is connecting a pair of SN nodes i and j with an edge. And substrate path $P^{sub} \in P^S$ represents a candidate hosting substrate topology, formulated from a collection of the basic segments P^{ij}, each constructed of a pair of two nodes and one physical edge, and its end-to-end delay is given by d_{ij}^a.

5.2. Virtual network model

The virtual network is modeled as a weighted directed graph $G^V = (N^V, E^V)$, where u and $v \in N^V$ are virtual nodes, and $(u,v) \in E^V$ is a virtual edge. VNR^r is a virtual network request number r out of R total VNRs. For online scenario, each VNR has a lifetime interval denoted by t, arrival time denoted by t_a^r, and expires at t_e^r. Each virtual node $u \in N^V$ is associated with loc_u denoting the preferred location to host the virtual node, cpu_u^r representing the demanded CPU capacity during time t, and each virtual edge (u,v) connecting a pair of virtual nodes u and v, is also associated with bw_{uv}^r, representing the demanded bandwidth capacity during time t, and d_{uv}^r representing the maximum allowed end-to-end delay demanded by virtual edge (u,v).

5.3. Power consumption model

For the online scenario, the linear power model introduced by Dayarathna et al. (2016) and Fan et al. (2007) is used to estimate power consumption of SN servers (PC), including idle power as given in Eq. (4). Nodes' utilization, U_i, is a fraction between (0–1), reflecting the ratio of consumed cpu_i to the maximum CPU_i given by Eq. (5).

$\forall\, i \in N^S$

$$PC_i = pc_i^{idle} + [PC_i^{Busy} - pc_i^{idle}] \times U_i \tag{4}$$

$$U_i = \frac{cpu_i}{CPU_i} \tag{5}$$

5.4. Online objective function definition and formulation

The main target of the objective function for OPaCoVNE is to successfully accommodate all the demands of the arriving VNRs on online scenario, while minimizing overall power consumption in the whole substrate network, by putting into sleeping mode all idle resources. Demands in the online case arrive at certain t_a^r and expire at t_e^r, therefore, the objective function must consider embedding VNRs during the time intervals specified by each related VNR^r.

To make sure that a specific physical node is active, variable x_i^{ur} is used in the objective function formulation, which takes a binary value of 1 if substrate node i is active and assigned to host the virtual node u ($u \leftarrow i$), during the time interval $t \in [t_a^r, t_e^r]$, and 0 otherwise. The objective function is shown in Eq. (6):

$\forall\, t \in [t_a^r, t_e^r],\ \forall\, u \in N^V and\ \forall\, r \in R$

$$\min PC = \sum_{\forall i \in N^S} (pc_i^{idle} + [PC_i^{Busy^t} - pc_i^{idle}] \times U_i^t) \times x_i^{ur} \tag{6}$$

5.4.1. Constraints definition and formulation

Objective function solution will be constrained by location, capacity, flow, and domain constraints as shown bellow.

Location constraints

$$\forall\, u \leftarrow i \quad loc_u^r = loc_i \tag{7}$$

$$\forall\, (u,v) \leftarrow (i,j) \quad Adj_{(u,v)}^r = Adj_{(i,j)}^{sub} \tag{8}$$

Capacity constraints

$$\forall\, t \in [t_a^r, t_e^r],\ \forall\, u \leftarrow i \quad \sum_{r \in R} cpu_u^{r^t} \le CPU_i \tag{9}$$

$$\forall\, t \in [t_a^r, t_e^r],\ \forall\, (u,v) \leftarrow (i,j) \quad \sum_{r \in R} bw_{uv}^{r^t} \le BW_{ij} \tag{10}$$

$$\forall\, t \in [t_a^r, t_e^r],\ d_{ij}^a \le d_{uv}^{r^t} \tag{11}$$

Flow constraints

$\forall\, t \in [t_a^r, t_e^r]$

$$\sum_{\forall m \in N^S} f_{sm}^t - \sum_{\forall m \in N^S} f_{ms}^t = bw_{uo}^{r^t} \tag{12}$$

$$\sum_{\forall m \in N^S} f_{dm}^t - \sum_{\forall m \in N^S} f_{md}^t = -bw_{pv}^{r^t} \tag{13}$$

$$k \ne s, d \quad \sum_{\forall m \in N^S} f_{km}^t = \sum_{\forall m \in N^S} f_{mk}^t \tag{14}$$

Domain constraints

$$\forall\ t \in [t_a^r, t_e^r],\ \forall\ i \in N^S\ x_i^{ur^t} \in \{0,1\} \tag{15}$$

$$\forall\ t \in [t_a^r, t_e^r],\ \forall\ u \in N^V \sum_{\forall i \in N^S} x_i^{ur^t} = 1, \tag{16}$$

At each time interval $[t_a^r, t_e^r]$, Eq. (7) ensures that each virtual node is located at its demanded location, and Eq. (8) ensure that each virtual edge is only located on a single substrate edge. Eq. (9) ensures that the total consumed CPU processing power capacity at substrate node i is less than or equal to the maximum CPU capacity at that SN node, while Eq. (10) ensures the same for bandwidth capacity. End-to-end delay in P^{ij} is controlled through Eq. (11) to be less than or equal to the maximum demanded delay by VNR^r. Eq. (12) ensures that the net flows getting out of the source node s is the demanded flow $bw_{uo}^{r^t}$ by virtual edge (u, o) during time interval t. While Eq. (13) ensures that, the net flow getting out of the destination node d is the forwarded flow $bw_{pv}^{r^t}$ by virtual edge (p, v). And Eq. (14) ensures that all flows are transferred through any intermediate node k between source node s and destination node d, and nothing remains at that intermediate node. Regarding network domain, Eq. (15) ensures to solve the problem as ILP, and Eq. (16) to guarantee each virtual node is mapped only to one substrate node.

5.5. OPaCoVNE to solve VNE in full coordination

Optimal solution for VNE is known to be NP-Hard and computationally intractable, since it can be reduced to the multi-way separator problem, which is NP-Hard by itself (Chowdhury et al., 2012). Some of the main reasons highlighting why solving VNE is challenging, could be due to randomness of the arrival of VNRs depending on users' demands, as well as topology and limited SN resources. However, the virtual edges' embedding problem is what makes VNE problem exceptionally an NP-hard, since it can be reduced to the unsplittable flow problem, which is NP-hard (Bradley and Magnanti, 1977; Kolliopoulos and Stein, 1997). Consequently, solving VNE problem in polynomial time is not possible. Therefore, majority of VNE approaches followed heuristic or meta-heuristic algorithms to solve VNE optimization problems in a reasonable polynomial time (Fischer et al., 2013).

Accordingly, to solve the objective function, fully coordinating virtual nodes and edges' embeddings together, while minimizing the total power consumption in the whole substrate network, this paper proposes the online algorithm, OPaCoVNE, as a power aware VNE heuristic, which can provide final results for the VNE problem in milliseconds of time. Its main advantage in reducing the total power consumption relies on using only direct edges between any pair of nodes, then utilizes the segmentation approach for comparing each element in the VNR^r set of segments, Set^r, against their counterparts in the SN path's set of segments, Set^S, considering the following constrains, namely: *loc*, *CPU*, *BW*, and end-to-end delay. If the elements of the substrate segment has enough resources to accommodate the demands as stated by the elements of the VNR's segments, a successful embedding occurs.

5.6. OPaCoVNE explained

The OPaCoVNE methodology is shown in the flowchart shown in Fig. 3 and the pseudo-code is shown in Algorithm 1. At each time t, if a VNR^r arrives, the code structures its set of segments, and based on the demanded locations for the virtual nodes, the code will build a similar SN topology to the VNR^r topology, then it formulates the SN set of segments, and compares both sets to check if the candidate SN segments, have enough resources to host all demands of VNR^r during the time interval. At each iteration, the algorithm keeps checking whether any VNR has expired, in oder to remove its demands from the hosting resources, and it updates the whole SN accordingly. In parallel with that, at each iteration cycle, OPaCoVNE evaluates the power consumption of each SN node if it is idle or loaded, and turns it off, if it has zero utilization, to save the overall power consumption in the SN.

More elaborations about the steps of OPaCoVNE are discussed in the subsections.

5.6.1. Initialization

OPaCoVNE starts by listing all pairs of the VNR^r, then using the demanded locations of the virtual nodes and edges, the algorithm will allocate the substrate pairs complying with the demanded locations' constraints, without using any hidden hops or edges, consequently, it will construct the candidate substrate topology path P^{sub} similar to the topology structure of VNR^r. It is assumed that the substrate network topology is physically fixed, therefore, the main elements formulating substrate network, such as number and connectivity of the SN nodes and edges, are also fixed and does not change, but only their capacities varies due to the consumption after each time interval t.

Algorithm 1 OPaCoVNE Pseudo-Code.

1. Input: G^S and G^V.
2. **while** $t \neq 0$ **do**
3. **for** each VNR$^r \in R$ arriving the SN randomly at time t
   Formulate Set^r as Eq. (2).
4. **for** the set of all SN pairs $P^{ij} \in P^S$:
   - **List** all SN pairs matching the nodes' and edges locations of VNR^r pairs, and ensure.
   they are connected by one SN edge only.
   - Formulate Set^S representing P^{sub} as Eq. (3).
   - **Compare** Set^r against Set^S.
5. - **If** satisfied,
   - **Main Output** - Embed VNR^r on P^{sub}.
   - Update all parameters of SN nodes and edges.
   - **else** go to next VNR, step-3.
6. **for** All embedded VNRs.
   - Check and offload expiring VNRs.
   - Update SN's *CPU* and *BW* resources.
7. **for** All SN nodes.
   - **Calculate** current U_i.
   - **if** $U_i = 0$
   - Turn-off SN node i to save power.
8. Evaluate Metrics.
9. **If** VNRs list not empty, **go** to next VNR step-3.

5.6.2. Segmentation

This is the differentiating aspect of OPaCoVNE algorithm compared to others, mainly because it facilitates solving the two subproblems of VNE, virtual nodes' embedding problem and virtual edges' embedding problem, together, providing full coordination, while assuring not to use any additional substrate resources, to minimize total power consumption in the physical network. To do that, first OPaCoVNE will reformulate all pairs of the arriving VNR^r into a set of segments, Set^r, following the format of Eq. (2). Then, it will construct the set of segments, Set^S, representing all pairs of the candidate SN topology path P^{sub} following Eq. (3). This will prepare OPaCoVNE to check the possibilities of embedding each virtual pair of nodes and their edge, onto a corresponding substrate pair of nodes and their edge, one-to-one, together, and without utilizing any hidden hopes or edges.

5.6.3. Embedding decision

This is the main step in making sure that each pair of virtual nodes and their edge are embedded together, resulting on solving the two subproblems of the VNE in full coordination altogether. To guarantee fully coordinated embeddings, OPaCoVNE compares each element in the SN set of segments Set^S to its counterpart in Set^r, one-to-one, at the same time, and in one comparison step, as shown in the comparison inequalities given by Eq. (17). For example, each loc_i and cpu_i^a element in Set^S is

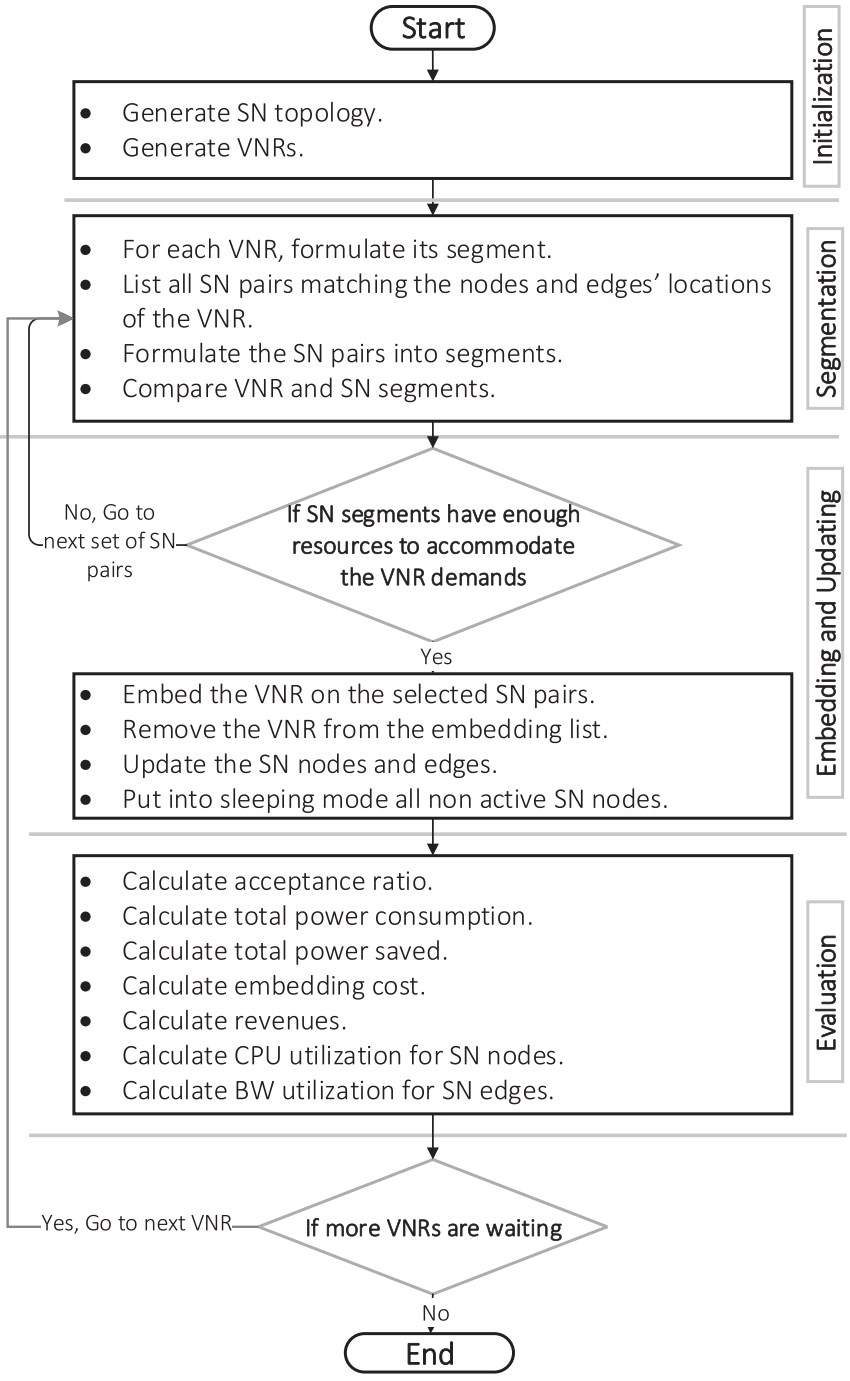

Fig. 3. Flowchart 1 presenting the general structure of the methodology used by OPaCoVNE. It starts by the initialization phase, then segmentation, followed by the embedding and updating phases, and concludes by the evaluation phase.

compared to its counterpart loc_u^r and cpu_u^r in Set^r, and at the same time, do the same for the remaining elements representing BW and delay. Notice the use of '**AND**' in Eq. (17) to guarantee the full coordination ***if and only if*** every element in Set^S satisfies the needs of its counterpart in Set^r, while embedding each virtual edge on a single substrate edge only, through using the adjacency matrices of the candidate substrate path, Adj^{sub}, and Adj^r for the VNR^r.

Therefore, if the inequalities in Eq. (17) are '**ALL**' true for every candidate substrate and virtual pair, which means that the demands of every virtual node and edge can be accommodated by the corresponding resources in the substrate path represented by Set^S, accordingly, a successful fully coordinated embedding for the virtual nodes and edges is performed for both of them together. Decision matrix for the embedding process is given as follows:

$$
\begin{aligned}
&i \in Set^S \text{ and } u \in Set^r \textbf{ if } loc_i = loc_u^r \textbf{ AND} \\
&(i,j) \in P^{sub} \text{and } (u,v) \in E^V \quad Adj^{sub}_{(i,j)} = Adj^r_{(u,v)} \textbf{ AND} \\
&i \in Set^S \text{ and } u \in Set^r \textbf{ if } cpu_i^a - cpu_u^r \geq 0 \textbf{ AND} \\
&(i,j) \in Set^S \text{ and } (w,x) \in Set^r \textbf{ if } bw_{ij}^a - bw_{wx}^r \geq 0 \textbf{ AND} \\
&\textbf{if } d_{ij}^a \leq d_{uv}^r
\end{aligned}
\tag{17}
$$

5.6.4. Updating

Once a successful embedding occurs, the algorithm updates all changed SN resources and moves to next VNR^{r+1}. However, in case that the selected SN set of segments does not have enough resources to accommodate VNR^r demands, the algorithm rejects VNR^r, and jumps to the next VNR from step-3. This process keeps on going until no more VNRs to be handled.

5.7. PaCoVNE computational time complexity

Regardless the number of VNRs and based on the adjacency matrix of SN, OPaCoVNE searches and lists all pairs in $O(|N^S| + |E^S|)$ processing time, depending on the total number of nodes N and edges E formulating the SN. Once all pairs are listed, the actual embedding process starts, which consumes almost a negligible processing time in milliseconds, since it is based on checking all elements in the comparison statement as in Eq. (17).

5.8. General illustrative example

A brief example to explain the proposed OPaCoVNE is shown in Fig. 4. At the top of the figure, four VNRs need to be embedded on the substrate network. Each VNR has clear specific location demands, as well as *CPU*, *BW*, and *delay*. For example VNR^1 requests that, the location of virtual node 1.1 must be at the location of substrate node *a*, virtual node 1.2 to be located at *b*, and 1.3 at *c*. Therefore, OPaCoVNE algorithm will first confirm the location constraints of the nodes, and if each pair of the located substrate nodes are connected by one direct

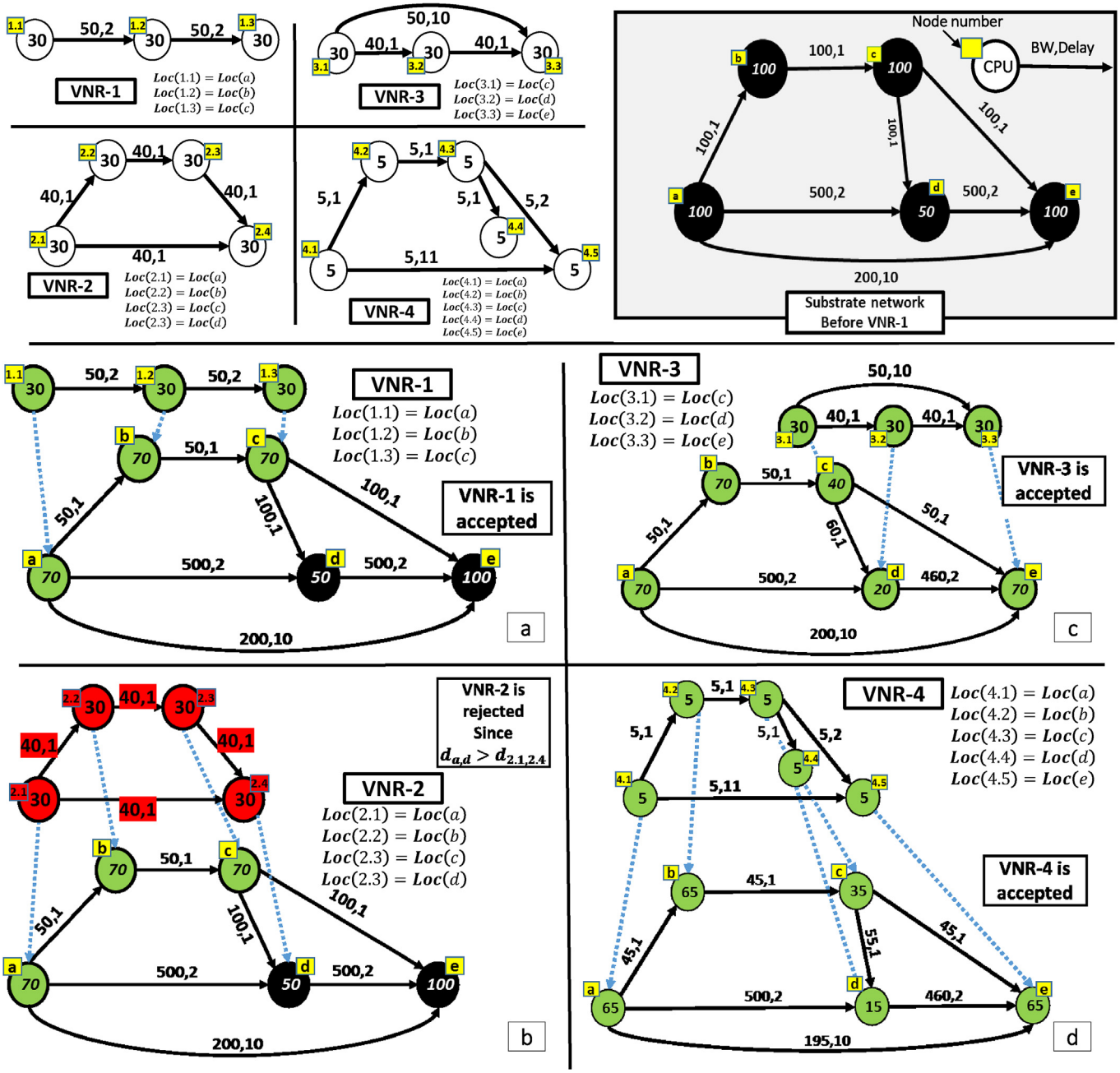

Fig. 4. Numerical example showing the basics of OPaCoVNE. At the top left, Four VNRs need to be embedded on the substrate network as given at the top right. Each VNR has clear specific location demands, as well as *CPU*, *BW*, and *delay*. In a,c, and d, VNR-1, VNR-3, and VNR-4 were allocated and the substrate network was updated, while in b, VNR-2 was rejected due to the delay constraint.

substrate edge only, as shown in Fig. 4a, where substrate nodes a and b are directly connected by the edge (a, b), as well as nodes b and c are also directly connected by edge (b, c). Next, OPaCoVNE will move to check *CPU*, *BW*, and *delay* demands of VNR^1, where virtual nodes 1.1, 1.2, and 1.3 demand that the substrate nodes a, b, and c must have at least 30 units of *CPU* per each of them, which is true, since each of a, b, and c has 100*CPU* units. The same for the *BW* demands of virtual edges $(1.1, 1.2)$ and $(1.2, 1.3)$, demanding 50BW units and maximum of 2 delay units in substrate edges (a, b) and (b, c), which is also true, since each of them has 100 current *BW* units and 1 current delay unit. Accordingly, VNR^1 will be embedded on substrate nodes and edges of path P^{abc}, while substrate nodes d and e will be turned-off, since they are not utilized to minimize the total power consumption in the network. Following the same procedure, VNR^3 and VNR^4 will be embedded on substrate paths P^{cde} and P^{abcde} respectively as shown in Fig. 4c and d. However, for VNR^2 even though substrate path P^{abcd} fulfills its *loc*, *CPU*, and *BW* demands, yet it was rejected, since delay demand of virtual edge $(2.1, 2.4)$ was less than the current edge delay in the candidate substrate edge (a, d).

5.9. Evaluation metrics

The performance of OPaCoVNE algorithm will be evaluated based on acceptance ratio, total power consumption, saved power, total revenues, total cost, *CPU* and *BW* utilizations, and processing time.

5.9.1. Average acceptance ratio, AR

Is a ratio to represent how OPaCoVNE algorithm is performing, and is calculated by averaging and dividing the number of successfully embedded VNRs at each time interval by the total number of VNRs R (Chowdhury et al., 2012; Fischer et al., 2013).

$$AR = \frac{1}{T}\sum_{\forall t\in T}\frac{1}{R}\ Total\ Number\ of\ Embedded\ VNRs * 100 \tag{18}$$

5.9.2. Average power consumption, PC

Calculated by averaging the total power consumed by all SN nodes and edges after each time interval t, (Botero and Hesselbach, 2013),

$$PC = \frac{1}{T}\sum_{\forall t\in T}\sum_{\forall i\in N^S} PC_i^t \tag{19}$$

5.9.3. Average saved power, PS

The average amount of saved power when the proposed power reduction strategy was used. It is calculated after each time interval t, given by subtracting the total power consumed by the whole SN without power reduction strategy PC^{t-}, from the total power consumed by all SN after applying the power reduction strategy PC^{t+}.

$$PS = \frac{1}{T}\sum_{\forall t\in T}\left(\sum_{\forall i\in N^S} PC_i^{t-} - \sum_{\forall i\in N^S} PC_i^{t+}\right) \tag{20}$$

5.9.4. Average cost, Co

The average amount of allocated virtual processing powers multiplied by the processing unite price μ, and throughputs multiplied by the flow unite price ρ, at each time interval t (Chowdhury et al., 2012).

$$Co = \frac{1}{T}\sum_{\forall t\in T}\left(\sum_{\forall u\in N^V}\mu\times cpu_u^{r^t} + \sum_{\forall (u,v)\in E^V}\rho\times bw_{uv}^{r^t}\right) \tag{21}$$

5.9.5. Average revenues, Rev

Representing revenues from all accepted virtual requests at each time interval t, calculated by adding the total demanded processing powers and throughputs (Chowdhury et al., 2012).

$$Rev = \frac{1}{T}\sum_{\forall t\in T}\left(\sum_{\forall u\in N^V} cpu_u^{r^t} + \sum_{\forall (u,v)\in E^V} bw_{uv}^{r^t}\right) \tag{22}$$

5.9.6. Average CPU utilization, CPU_{util}

It represents the average SN nodes' utilization trend after all simulation iterations. Its defined as a ratio between consumed CPU cpu_i^t, and maximum *CPU* resources, averaged over all SN nodes (Chowdhury et al., 2012).

$$CPU_{util} = \frac{1}{T}\sum_{\forall t\in T}\frac{1}{N^S}\left(\sum_{\forall i\in N^S}\frac{cpu_i^t}{CPU_i} * 100\right) \tag{23}$$

5.9.7. Average BW utilization, BW_{util}

It represents the average utilization of SN edges after all simulation iterations. And is defined as a ratio between consumed bw_{ij}^t, and the maximum *BW*, averaged over all SN edges (Chowdhury et al., 2012).

$$BW_{util} = \frac{1}{T}\sum_{\forall t\in T}\frac{1}{E^S}\left(\sum_{\forall (i,j)\in E^S}\frac{bw_{ij}^t}{BW_{ij}} * 100\right) \tag{24}$$

6. OPaCoVNE evaluations

To generally evaluate the new online power aware and fully coordinated virtual network embedding algorithm, OPaCoVNE, three sets of simulations were conducted in this paper:

1. OPaCoVNE overall performance in terms of acceptance ratio, embedding cost, in addition to nodes and edges' utilizations will be compared against the most referenced online algorithm, D-ViNE by Chowdhury et al. (2012), as explained in subsection 6.1.
2. To investigate OPaCoVNE's power aware performance in terms of average power consumption, average generated revenue, and number of switched off substrate nodes, the performance of OPaCoVNE was compared against the state of art algorithm, EAD-VNE, by Zhongbao et al. (2015), as detailed in subsection 6.2.

Table 2
Comparing OPaCoVNE to D-ViNE algorithm.

Item	OPaCoVNE	D-ViNE
Scenario	Online	Online
Goal	Minimize total power consumption	Minimize embedding cost
Strategy	Confirm locations' constraints	Confirm locations' constraints
	Check residual capacities then consolidate	Check residual capacities then embed
Embedding coordination	Fully coordinated	Partially coordinated
Location constraints	Yes	Yes
Virtual nodes embedding	Based on location and residual capacities	Based on location and residual capacities
Virtual edges embedding	Direct edge between each pair of nodes	Using multi-commodity flow algorithm
Power consumption	Yes	No
End-to-end delay	Yes	No

3. Furthermore, four different experiments were also conducted as shown in subsection 6.3 to test the impacts of varying number, lifetime, and size of the embedded virtual requests, as well as the impacts of varying size of substrate edges, on OPaCoVNE's average power consumption, acceptance ratio and processing time.

6.1. Evaluating overall performance of OPaCoVNE

The overall performance of OPaCoVNE was compared to D-ViNE, which targeted minimizing the embedding cost while improving acceptance ratio, by mapping the virtual nodes based on the total flow passing through the edges of a candidate SN, and applies multi-commodity flow algorithm to map the virtual edges onto paths connecting the hosting SN nodes. Table 2 provides a high-level comparison between OPaCoVNE and D-ViNE algorithms, listing their used strategies, and how they embed the virtual nodes and edges.

6.1.1. Simulation settings

Substrate network and virtual network topologies were randomly generated using Waxman algorithm, setting $\alpha = 0.7$, $\beta = 0.9$, and mean probability of a pair of two nodes being connected set equal to 0.5. Similar to (Chowdhury et al., 2012), the substrate network includes 50 nodes, *CPU* and *BW* resources given as real numbers, uniformly distributed between 50 and 100, and delay in each substrate edge was randomly selected between 1 and 50. Number of virtual nodes per VNR was randomly determined by a uniform distribution between 2 and 10, assuming that VNRs arrive according to Poisson process pattern, each having an exponentially distributed lifetime with an average of 1000 time units. Arriving rates of VNRs were varied between 4 and 8 per 100 time units, over simulation time of 50,000 units. The virtual *CPU* and *BW* resources were real numbers uniformly distributed between 0–20 and 0–50 respectively, while delay in each virtual edge was randomly selected between 20 and 100. Substrate network and virtual network topologies were assumed located on the same grids, and locations of virtual nodes were selected based on the corresponding locations of the substrate nodes. Table 3 summarizes all simulation parameters.

Table 3
Simulation settings to test OPaCoVNE against D-ViNE.

Parameter	SN	VNR
Nodes	50	2–10
CPU	50–100	0–20
BW	50–100	0–50
Delay	1–50	20–100
PC^{Busy}	$15 * CPU$	
pc^{idle}	165 *Watts*	
time units	0–50,000	
α	0.7	
β	0.9	
p_{wax}	0.5	
VNRs/100 *time units*	4–8	
VNR *lifetime*	1000 *time units*	

6.1.2. Discussing results of OPaCoVNE's overall performance

Analyzing acceptance ratio: The average acceptance ratio of OPa-

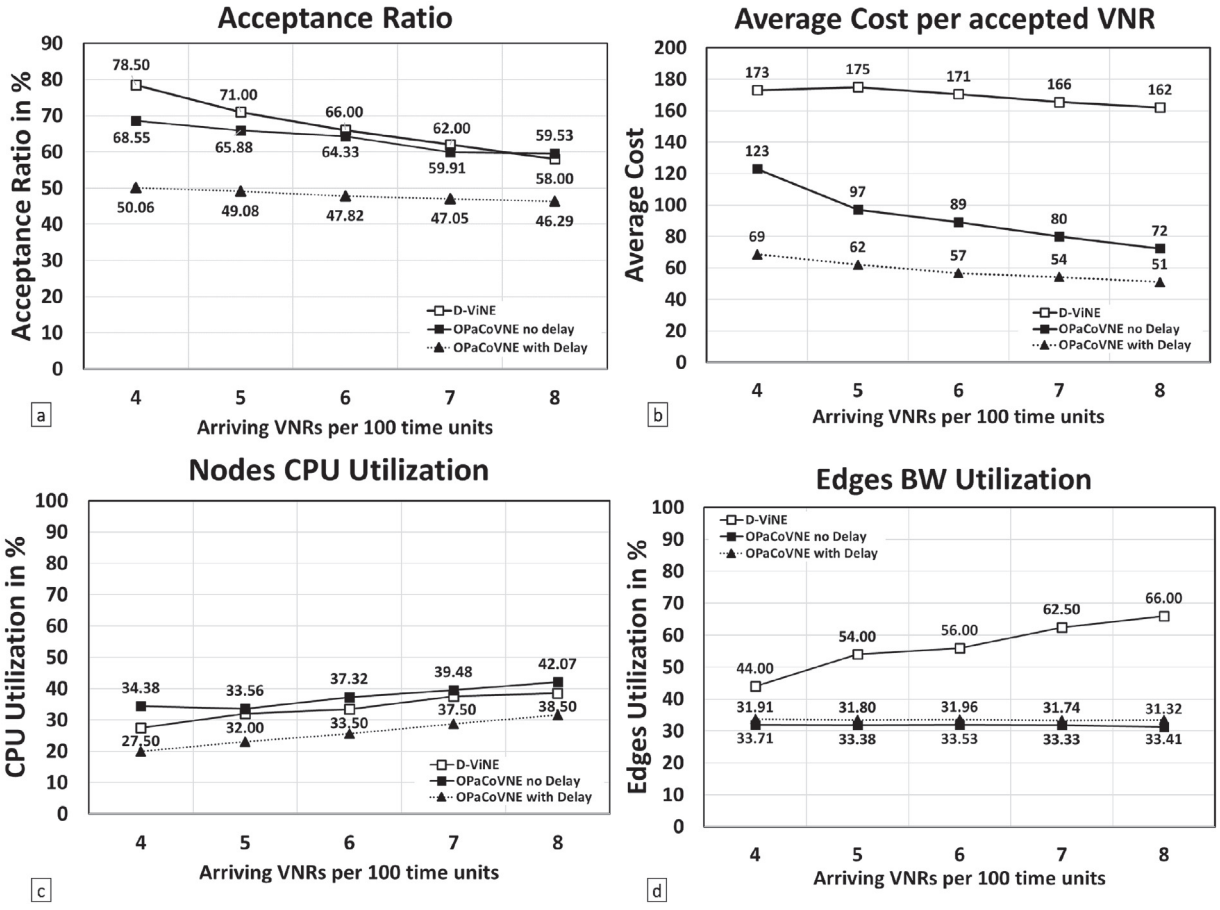

Fig. 5. Overall Performance of OPaCoVNE with and without end-to-end delay compared to D-ViNE. In a, the average acceptance ratio of OPaCoVNE was 63.64% compared to 67.10% for D-ViNE. In b, the average cost using OPaCoVNE was around 92.28 compared to D-ViNE of 169.20. In c, nodes' utilization using OPaCoVNE was in average around 37.36% compared to 33.80% for D-ViNE, and in d, average edges' utilization was around 31.75% lower than D-ViNE of 56.5%.

Table 4
Comparing OPaCoVNE to EAD-VNE algorithm.

Item	OPaCoVNE	EAD-VNR by Zhongbao et al. (2015)
Scenario	Online	Online
Goal	Minimize power consumption	Minimize power consumption, high revenues.
Strategy	Confirm locations' constraints	Confirm locations' constraints
	Use resource of residual capacities then consolidate	Use resource of residual capacities
Embedding coordination	Fully coordinated	Partially coordinated
Location constraints	Yes	Yes
Virtual nodes embedding	Based on location and residual capacities	Based on location and residual capacities
Virtual edges embedding	Direct edge between each pair of nodes	Using shortest path algorithm
Power consumption	Yes	Yes
End-to-end delay	Yes	No

CoVNE was 63.64% compared to 67.10% for D-ViNE as shown in Fig. 5a, which indicates the close similarity in the overall performance of both algorithms. However, the fact that OPaCoVNE strategy in selecting only direct edges to minimize the used resources when embedding virtual edges, in order to minimize the total power consumption, has negatively impacted the algorithm's acceptance ratio, yet OPaCoVNE still managed to show solid and comparable results to D-ViNE in general, while leaving more free resources to host more demands. More precisely, this is due to applying the segment technique which insures that the virtual nodes and virtual edges were mapped together on the SN path that, complies with the location constraints and has enough resources and acceptable end-to-end delay, without any hidden nodes or edges.

Similar to OPaCoVNE, D-ViNE maps virtual nodes on physical nodes having enough residual capacities, and maps virtual edges through applying multi-commodity flow algorithm to find the appropriate SN edge between the already mapped virtual nodes. Nevertheless, the way D-ViNE selects the SN nodes to host virtual nodes, may result on selecting SN nodes that are further away from each other, which when applying the multi-commodity flow algorithm to map the virtual edges, could force using longer paths, therefore, consuming more resources, but would have better acceptance ratios. That was clear for the acceptance ratios of D-ViNE when the loads were between 4 and 7. However, the results of OPaCoVNE started to converge with those of D-ViNE, most probably due to the multi-commodity flow algorithm's way of selecting more edges, and allowing for splitting the flows between more than one substrate path, thus, raising the possibilities of congested edges, which in turn would cause degradation in the acceptance ratio. Moreover, referring to Fig. 5a, and focusing on the load of 8 VNRs per 100 time units, OPaVoVNE acceptance ratio started to get better than that of D-ViNE, most likely since it would always stick to using less edges during the embedding process, therefore, keeping more free resources to be used to fulfill the demands of the new virtualization requests.

Average Cost: Average cost per accepted VNR is shown in Fig. 5b, confirms the successful strategy of OPaCoVNE in minimizing the use of substrate network resources as least as possible while embedding VNRs, by selecting direct edges between the hosting substrate nodes without any hidden hopes. Therefore, in average OPaCoVNE cost was around 92.28 compared to D-ViNE cost of 169.20, which most likely tends to use more edges when embedding VNRs than OPaCoVNE. The processing power unite price μ, and throughput unite price ρ were set equal to 1.

SN nodes Utilization: Fig. 5c shows nodes' average utilization. SN nodes in OPaCoVNE are in average moderately utilized resulting on 37.36%, which is very close when compared to 33.80% for D-ViNE. This is a reflection of the similarity between both OPaCoVNE and D-ViNE in utilizing SN nodes having enough residual resources to embed virtual nodes. Also it is important to point out that both algorithms first select the SN nodes according to the location constraint, then they check for the residual resources afterwards. In the case of including end-to-end delay, OPaCoVNE nodes' utilization is less than without delay, mainly since OPaCoVNE with end-to-end delay accepted less number of demands, thus has less utilized SN nodes.

SN Edges Utilization: Average edges utilization in OPaCoVNE resulted on 31.75%, and was lower than D-ViNE of 56.5% as shown in Fig. 5d. This is mainly because OPaCoVNE is allocating precisely the same number of SN edges as the demanded edges without any hidden edges or nodes. However, the way D-ViNE embedding virtual edges using the multi-commodity flow algorithm, could have resulted on using longer paths including more SN edge than requested, which raises its overall edges' utilization.

Impact of delay on OPaCoVNE: The impact of end-to-end delay on OPaCoVNE, was negative in general over all simulations as shown in Fig. 5. However, since D-ViNE did not use delay as a constraint, the results of OPaCoVNE were shown just to reflect how much the performance of OPaCoVNE without delay is better than when it was included.

6.2. Evaluating power consumption performance of OPaCoVNE

This subsection compares the power aware performance of OPaCoVNE against EAD-VNE algorithm, which applies power aware with dynamic demands as well (Zhongbao et al., 2015). Table 4 compares the two algorithms, giving that for virtual nodes' embedding, both algorithms first use location constraint to select the substrate nodes, then embed the virtual nodes if enough residual resources are available. For the virtual edges, EAD-VNE uses shortest path, which may include additional edges, while OPaCoVNE uses direct edges connecting the pair of substrate nodes, without any hidden nodes or additional edges. Table 5 provides general settings to compare OPaCoVNE against EAD-VNE.

6.2.1. OPaCoVNE versus EAD-VNE

Exact simulation settings of EAD-VNE were applied on OPaCoVNE, assigning 4 VNRs to be embedded per 100 time units, each having average lifetime of 500 time units, and number of nodes per VNR were randomly distributed between 2 and 10. As shown in Fig. 6a, the average power consumption per SN node using OPaCoVNE is 23.54% lower than

Table 5
Simulation settings to test OPaCoVNE against EAD-VNE.

Parameter	SN	VNR
Nodes	50	
CPU	50–100	0–20
BW	50–100	0–50
Delay	1–50	20–100
PC^{Busy}	$15 * CPU$	
pc^{idle}	165 *Watts*	
time units	0–50, 000	
α	0.7	
β	0.9	
p_{wax}	0.5	
Experiment-1		
VNRs/100 *time units*	4	
VNR *lifetime*	500 *time units*	
VNRs' *nodes*	Random 2–10	

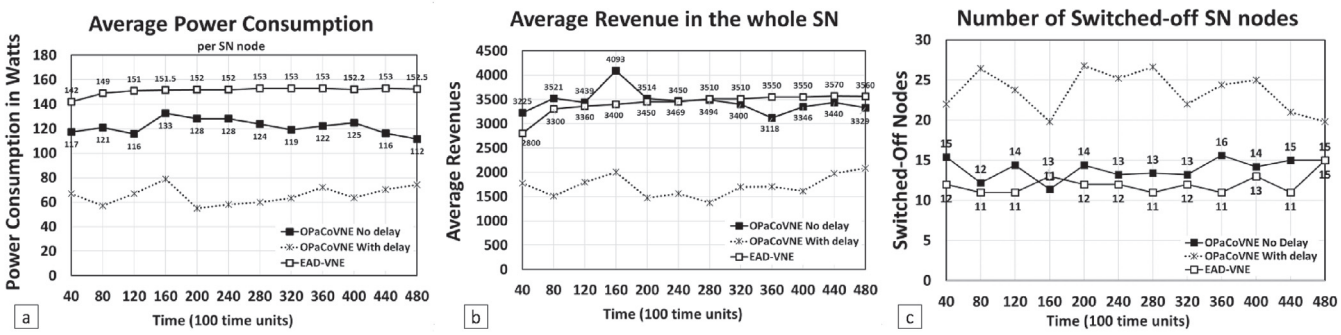

Fig. 6. OPaCoVNE performance against EAD-VNE. In a, the average power consumption per substrate node using OPaCoVNE is 23.54% lower than EAD-VNE, while in b it shows that OPaCoVNE revenues are almost near to those of EAD-VNE, which in average were less by 4.4%, and in c it shows that OPaCoVNE managed to turn-off more substrate nodes than EAD-VNE by 22.2%.

EAD-VNE along the simulation duration, confirming the better power aware performance of OPaCoVNE. Fig. 6b presents average revenue in the substrate network, showing that OPaCoVNE revenues are almost near to those of EAD-VNE, and in average were less by 4.4%. Noting that OPaCoVNE and EAD-VNE embed virtual nodes based on location and residual nodes' resources constraints, yet, in contrary to EAD-VNE, which uses shortest path to embed virtual edges, OPaCoVNE's way in embedding each virtual edge on a single substrate edge, may cause some VNRs to be rejected, which would reduce the overall revenues of OPaCoVNE in general.

Nevertheless, the fact of using direct edges by OPaCoVNE is precisely why its power consumption is better than EAD-VNE, since OPaCoVNE strategy uses least substrate network resources as much as possible, through using direct edges with no hidden hopes at all. This is confirmed referring to Fig. 6c, which shows that OPaCoVNE strategy managed to turn-off more substrate nodes than EAD-VNE by 22.2%.

6.3. Varying number, lifetime, size of VNRs, and size of substrate edges

Table 6 lists specific settings for experiments 2–5 that were designed to evaluate OPaCoVNE in more details. The four experiments were specifically designed to evaluate the online power consumption and acceptance ratio behaviors of OPaCoVNE, when end-to-end delay was included as a major embedding constraint. In the second experiment, number of arriving VNRs were varied between 2 and 10 each 100 time units, while the third experiment varied VNRs' lifetime between 200 and 1000 time units, and the fourth experiment varied number of nodes per each embedded VNR between 2 and 10. Finally, fifth experiment evaluated OPaCoVNE power consumption and acceptance ratios for five different substrate network topologies.

6.3.1. Average power consumption of OPaCoVNE

As shown in Fig. 7a–d, average power consumption and impacts of adding delay constraint on OPaCoVNE's performance were reported using different settings according to experiments 2–5. From Fig. 7a, varying number of embedded VNRs between 2 and 10 per 100 time units, reflects the loading impacts on OPaCoVNE's power consumption, which showed increasing trend in general from 76% when the number of embedded VNRs were as low as 2, to 96% for 10 loaded VNRs each 100 time units. Same increasing trends are also shown in Fig. 7b, but in this case number of embedded VNRs were fixed on 4 per 100 time units, while varying their lifetime from 200 to 1000 time units. In both experiments, impact of delay on the overall power consumption resulted on less values than the cases when delay was not included, yet it showed increasing trends in both settings.

Furthermore, Fig. 7c presents the results when fixing number of embedded VNRs per 100 time units to 4, while varying number of virtual nodes per VNR between 4 and 10, which resulted on slight

Table 6
Experiments 2–5 to test OPaCoVNE.

Experiment-2	Varying number of arriving VNRs
VNRs/100 *time units*	2–10
VNR *lifetime*	500 *time units*
VNRs' *nodes*	Random 2–10
Experiment-3	Varying VNRs' lifetime
VNRs/100 *time units*	4
VNR *lifetime*	200–1000 *time units*
VNRs' *nodes*	Random 2–10
Experiment-4	Varying number of nodes per VNR
VNRs/100 *time units*	4
VNR *lifetime*	500 *time units*
VNR's *nodes*	from 4 to 10
Experiment-5	Different SN topologies
VNRs/100 *time units*	4
VNR *lifetime*	500 *time units*
VNR's *nodes*	Random 2–10
Topologies	Average SN edges (Edges/Node)
T1	380(7)
T2	530(10)
T3	630(12)
T4	770(15)
T5	860(17)

increases in the average power consumption when delay was not included, suggesting that the change in VNRs' size has minor impact in general. However, this conclusion changed dramatically when delay was included, where power consumption was higher for low sized VNRs, but decreased for larger VNRs. This result on power consumption with delay would most likely be due to high rejections of large sized VNRs.

Finally, Fig. 7d reported the average power consumption of OPaCoVNE for five different substrate network topologies, which were varied by increasing the number of edges per each of them. As shown, increasing number of connected edges per topology did not have any significant impact on OPaCoVNE's average power consumption in general, but when delay was include OPaCoVNE's power consumption showed slight decreasing trend for larger connected topologies, yet still it did not show any significant impact too. This is expected, since OPaCoVNE uses location constraint, suggesting that performance of OPaCoVNE will very slightly be affected, regardless of the size of connected edges in the infrastructure.

6.3.2. Average acceptance ratio of OPaCoVNE

OPaCoVNE's average acceptance ratios for experiments 2–4 had a decreasing trend with and without delay as shown in Fig. 7e–g. For experiment-2, number of arriving VNRs per 100 time units varied between 2 and 10, the average acceptance ratio varied between 92.68% in the case of low number of arriving VNRs, and decreased

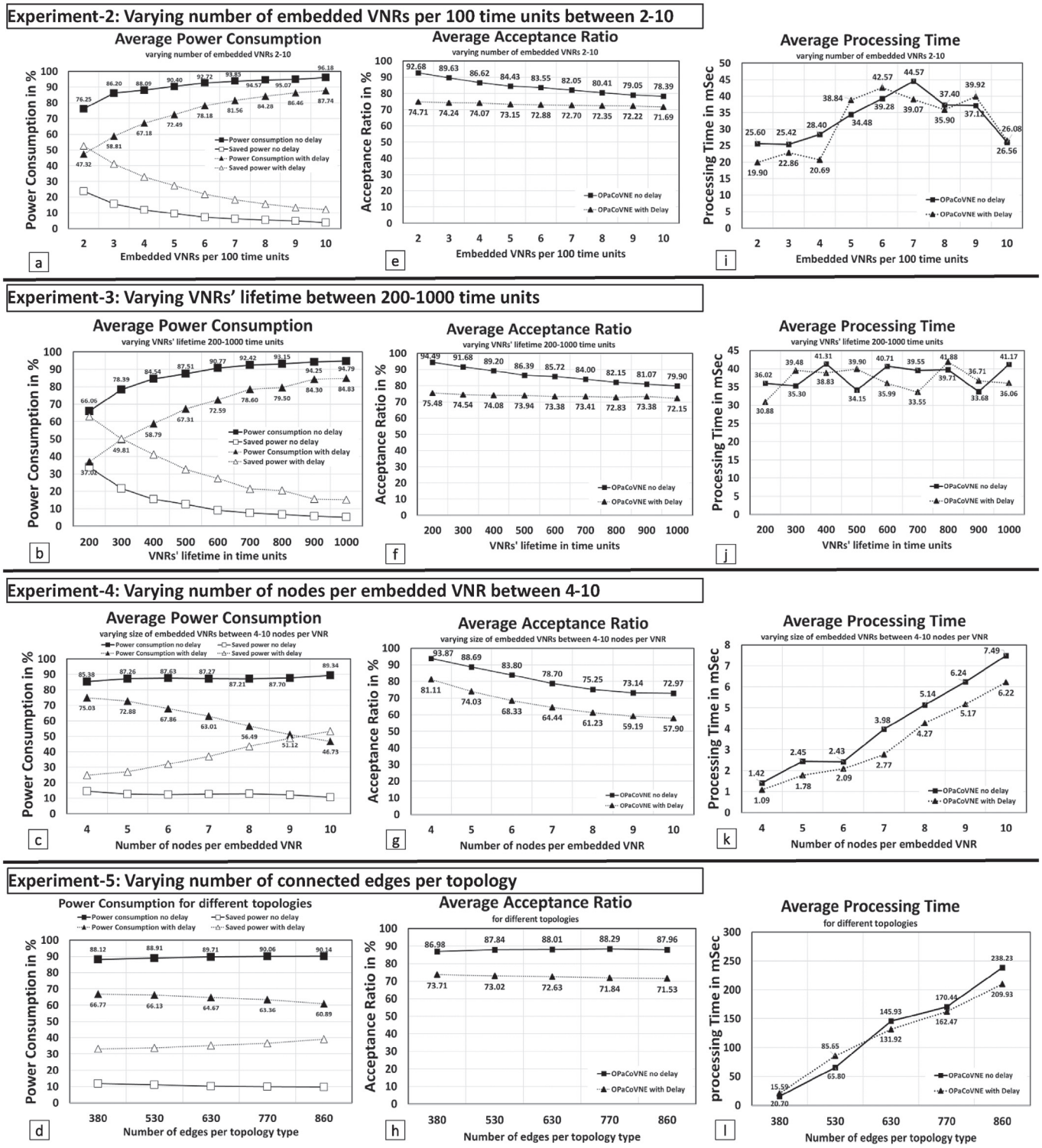

Fig. 7. OPaCoVNE performance for experiments 2–5 evaluating its online power consumption, acceptance ratio, and processing time. The results of the second experiment are shown in a, e, and i, where the number of arriving VNRs varied between 2 and 10 each 100 time units. Third experiment results are shown in b, f, and j, which varied VNRs' lifetime between 200 and 1000 time units. Fourth experiment results are shown in c, g, and k, which varied the number of nodes per each embedded VNR between 2 and 10, and finally, the fifth experiment results are shown in d, h, and l, which evaluated OPaCoVNE power consumption, acceptance ratios, and processing times for five different substrate network topologies.

to 78.39% when number of arriving VNRs was 10. Same trend was reported for experiment-3, when VNRs' lifetimes varied between 200 and 1000, which resulted on average acceptance ratio between 94.49% for short lifetimes, down to 79.90% for lifetimes as long as 1000 time units. Moreover, the average acceptance ratio for experiment-4, when number of virtual nodes per VNR varied between 2 and 10 resulted on 93.87% for small sized VNRs, decaying to 72.97% for larger VNRs.

However, in the case of experiment-5, which was designed to examine the impact of different substrate topologies on OPaCoVNE's performance while using location constraint, acceptance ratio trend slightly increased, from 86.98% for small connected substrates, to 87.96% for larger connected configurations as shown in Fig. 7h. This result implies that solving VNEs using location constraints was barely affected by the size of substrate network topology, compared to the resources' capacities of substrate network.

6.3.3. Average processing time of OPaCoVNE

The average processing time results for OPaCoVNE while performing experiments 2–5 are shown in Fig. 7i–l. In experiment-2, the average processing time was about 25.60 ms when the number of arriving VNRs were low, then increased as the number of arriving VNRs was increased up to 7 per 100 time units. Nevertheless, for larger number of arriving VNR between 8 and 10, average processing time decreased again, most probably due to an already loaded and limited substrate resources, which caused high rejected VNRs, therefore decreasing the processing time accordingly.

On the other hand, for experiment-3, average processing time was in the range of 37.96 ms as shown in Fig. 7j, giving that the number of arriving VNRs was fixed to 4, but varying VNRs' lifetimes between 200 and 1000 time units. This result suggests that OPaCoVNE's processing time performance was slightly affected when varying VNRs' lifetimes.

However, the general behavior of average processing time had an increasing trend when changing the size of VNRs or substrate network topologies as shown in Fig. 7k and l. The processing time trend for smaller VNRs varied between 1.42 and 7.49 ms, while for substrate networks that has large number of connected pairs of nodes, the average processing time varied considerably between 15.59 ms for topology $T1$ where average number of substrate edges was 380, to 238.23 ms for the case of topology $T5$, which had 860 edges.

In all previous experiments, OPaCoVNE code was developed using Eclipse IDE for Java Developers, version: Mars.2 Release (4.5.2). The used machine was Lenovo laptop, system model 20*CLS2RG*00, processor Intel(R) Core(TM) i7-5600U CPU, 2.60 GHz, 2 Cores, 4 logical processors, RAM 8 GB, and the operating system was Microsoft Windows 10 Enterprise.

6.3.4. Impact of delay on OPaCoVNE's performance

All simulation results showed that impact of delay on VNE process was clearly the most significant parameter among all varied variables while testing OPaCoVNE. Specifically, referring to Fig. 7e–h, OPaCoVNE's average acceptance ratios with delay, were less than when it was not included by 13.05%, 14.38%, 17.69%, and 17.39% for experiment-2 to experiment-5 respectively. Similar trends can be seen by referring to OPaCoVNE's results for average power consumption and processing time shown in Fig. 7.

In all experiments, values of SN delays were randomly assigned in the range 1–50 ms (5G Americas, 2017) and between 20 and 100 ms for VNRs. Accordingly, each time OPaCoVNE attempts to embed any VNR, it carefully tries to guarantee embedding on substrate edges connecting directly the selected substrate nodes, having enough bandwidth resources and end-to-end delay within the demanded ranges.

These values confirm the importance of including end-to-end delay as a major constraint when solving VNE problem, as a direct evaluation metric for real world 5G networks.

6.4. Summery of the findings from the simulations

To summarize the overall results from the three simulation sets mentioned in subsections 6.1, 6.2, and 6.3, the following points highlight the main findings:

1. Inclusion of end-to-end delay had significantly impacted VNE process, as reflected by lower acceptance ratios across all simulations in the range of 16%, when compared to the ratios without delay. Suggesting the importance of including end-to-end delay as a major VNE constraint when applied on the delay sensitive 5G networks.
2. Overall online performance of OPaCoVNE, measured by acceptance ratio was in average in the range of 63.64%, giving that it used less resources than the most referenced online algorithm by Chowdhury et al. (2012).
3. In terms of power consumption, OPaCoVNE managed to save 23.54% of the substrate network power compared to the online energy aware VNE algorithm, EAD-VNE by Zhongbao et al. (2015).
4. The results confirmed the effectiveness of the proposed segmentation approach, which allowed for a very precise and full coordination between the embeddings of both, virtual nodes and edges, together at the same time, without including any hidden resources.
5. The use of direct edges when embedding the virtual edges, guaranteed minimizing the use of substrate network resources to the minimum before activating others. In this way more substrate resources were left free to accept more virtual network requests, while insuring less power consumptions in the substrate network, and including end-to-end delay as a main constraint.
6. The drawback of the proposed algorithm is that, it had to make a trade-off between minimizing the total power consumption and accepting more virtualization demands, by limiting the use of the substrate network to minimum. Nevertheless, the suggested algorithm managed to provide very solid performance compared to others.

7. Conclusions

This paper introduced a new online embedding algorithm, OPaCoVNE, solving the two subproblems of VNE process in full coordination using end-to-end delay as a main constraint. The algorithm restructures virtual nodes and edges' demands in one set of segments, and formulates an exact similar set of segments for a specifically selected physical path topology, which comply with the exact demanded locations to embed the virtual nodes. Consequently, the embeddings occurs by comparing the two segments, one-to-one, and checking if each element in the physical segment can accommodate the demands of their counterparts from the virtual segment. In addition to that, to minimize the total power consumption in the whole substrate network, the proposed algorithm insures that each pair of the substrate network nodes hosting the virtual nodes, are directly connected by a single edge with no hidden hops or edges, which guarantees utilizing the least substrate network resource as low as possible.

The overall performance results of OPaCoVNE showed that, without delay, it managed to minimize the average power consumptions in the substrate network by 23.54%, however, when end-to-end delay was factored in, performance of OPaCoVNE was degraded across all evaluation metrics, suggesting that, introducing end-to-end delay as a major constraint, had clear impact on the whole VNE process, and therefore, it has to be one of the main metrics when evaluating real world 5G networks.

As a future work, the authors are planning to conduct further research, about reducing the power consumption of the OPaCoVNE algorithm and increase the acceptance ratio when non-direct edges are considered, considering end-to-end delay for specific 5G applications.

Acknowledgment

This work has been partially supported by the Ministerio de Economy of the Spanish Government under project TEC2016-76795-C6-1-R and AEI/FEDER, UE.

References

3GPP TR 28.801 (V15.0.0), 2017. Study on Management and Orchestration of Network Slicing for Next Generation Network. 3GPP TR 28.801.

5G Americas, 2017. 5G Network Transformation, White Paper. 5G Network Transformation.

Bianchi, F., Presti, F., 2017. A Markov reward based resource-latency aware heuristic for the virtual network embedding problem. SIGMETRICS Perform. Eval. Rev. 44 (4), 57–68, https://doi.org/10.1145/3092819.3092827.

Botero, J., Hesselbach, X., 2013. Greener networking in a network virtualization environment. Comput. Network. 57 (9), 20121–22039, https://doi.org/10.1016/j.comnet.2013.04.004.

Botero, J., Hesselbach, X., Fischer, A., Meer, H., 2012a. Optimal mapping of virtual networks with hidden hops. Telecommun. Syst. 51 (4), 273–282, https://doi.org/10.1007/s11235-011-9437-0.

Botero, J., Hesselbach, X., Duelli, M., Schlosser, D., Fischer, A., de Meer, H., 2012b. Energy efficient virtual network embedding. Commun. Lett., IEEE 16 (5), 756–759, https://doi.org/10.1109/LCOMM.2012.030912.120082.

Botero, J., Molina, M., Hesselbach, X., Amazonas, J., 2013. A novel paths algebra-based strategy to flexibly solve the link mapping stage of VNE problems. J. Netw. Comput. Appl. 36 (6), 1735–1752, https://doi.org/10.1016/j.jnca.2013.02.029.

Bradley, Hax, Magnanti, 1977. Applied Mathematical Programming. Addison-Wesley.

Chen, X., Li, C., Jiang, Y., 2016. A feedback control approach for energy efficient virtual network embedding. Comput. Commun. 80, 16–32, https://doi.org/10.1016/j.comcom.2015.10.010.

Chowdhury, M., Rahman, M., Boutaba, R., 2012. ViNEYard: virtual network embedding algorithms with coordinated node and link mapping. IEEE/ACM Trans. Netw. 20 (1), 206–219, https://doi.org/10.1109/TNET.2011.2159308.

Dayarathna, M., Wen, Y., Fan, R., 2016. Data center energy consumption modeling: a survey. IEEE Commun. Surv. Tutor. 18 (1), 732–794, https://doi.org/10.1109/COMST.2015.2481183.

Fan, X., Weber, W.D., Barroso, L.A., 2007. Power provisioning for a warehouse-sized computer. In: Pro. 34th Annu. ISCA, pp. 13–23, https://doi.org/10.1145/1250662.1250665.

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

Ghazisaeedi, E., Huang, C., 2017. Off-peak energy optimization for links in virtualized network environment. IEEE Trans. Cloud Comput. 5 (2), 155–167, https://doi.org/10.1109/TCC.2015.2440246.

Hou, W., Yu, C., Guo, L., Wei, X., 2016. Virtual network embedding for power savings of servers and switches in elastic data center networks. Sci. China Inf. Sci. 59 (12), 122307:1–122307:14, https://doi.org/10.1007/s11432-016-5590-0.

ITU-T Focus Group, 2017. IMT-2020 Deliverables. ITU-T Focus Group.

Kleinberg, J., Tardos, E., 2009. Algorithms Design. Addison-Wesley.

Kolliopoulos, S., Stein, C., 1997. Improved approximation algorithms for unsplittable flow problems. In: Proceedings 38th Annual Symposium on Foundations of Computer Science, Miami Beach, FL, pp. 426–436.

Nia, N., Adabi, S., Nategh, M., 2017. A coordinated heuristic approach for virtual network embedding in cloud infrastructure. KSII Trans. Internet Inf. Syst. 11 (5), 2346–2361, https://doi.org/10.3837/tiis.2017.05.002.

Nonde, L., El-Gorashi, T., Elmirghani, J., 2015. Energy efficient virtual network embedding for cloud networks. J. Lightwave Technol. 33 (9), 1828–1849, https://doi.org/10.1109/JLT.2014.2380777.

Ogino, N., Kitahara, T., Arakawa, S., Murata, M., 2017. Virtual network embedding with multiple priority classes sharing substrate resources. J. Computer Netw. 112 (C), 52–66, https://doi.org/10.1016/j.comnet.2016.10.007.

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

Triki, N., Kara, N., El Barachi, M., Hadjres, S., 2015. A green energy-aware hybrid virtual network embedding. Comput. Netw. 91, 712–737, https://doi.org/10.1016/j.comnet.2015.08.016.

Yu, M., Yi, Y., Rexford, J., Chiang, M., 2008. Rethinking virtual network embedding: substrate support for path splitting and migration. ACM SIGCOMM CCR 38 (2), 17–29, https://doi.org/10.1145/1355734.1355737.

Zhang, Z., Su, S., Zhang, J., Shuang, K., Xu, P., 2015. Energy aware virtual network embedding with dynamic demands: online and offline. Comput. Network. 93, 448–459, https://doi.org/10.1016/j.comnet.2015.09.036.

Khaled Hejja a PhD student at the networks engineering department, Universitat Politecnica de Catalunya, Barcelona, Spain. Worked for the Palestinian Telecommunication incumbent operator PalteGroup between 1998 and 2015, as the head of network planning and optimization. His research interests include, core and radio network virtualization, network planning and applications of artificial intelligence in the optimization of the telecommunication network.

Dr. Xavier Hesselbach (http://www-entel.upc.edu/xavierh/), Associate Professor at the Department of Network Engineering (Dept. Enginyeria Telematica) at the UPC, and IEEE Senior Member, received the M.S. degree with honors in Telecommunications Engineering in 1994 and the PhD. degree with honors in 1999, from the Universitat Politecnica de Catalunya (UPC). In 1993, he joined the Design, modelling and evaluation of broadband networks group in the Network Engineering Department of the UPC. His research interests include networks virtualization, resources management, broadband networks, quality of service and green networking. He has been involved in several national and international projects, and is author in 4 books and more than 100 national and international publications in conferences and journals. In 1994 he received the award from the COIT/AEIT of Spain for the best Master Thesis on Networks and Telecommunication Services. He has participated in the technical program committees of several conferences, he has been the Information Systems and Internet Chair in Infocom 2006, and guest editor of the Ad Hoc Networks Journal and the Journal of Electrical and Computer Engineering. He has taken part in several European and Spanish research projects, such as the EuroNGI/FGI/NF Network of Excellence, COST293, Mantychore and All4Green, being main UPC researcher in the Mantychore and All4Green projects.