Journal of Network and Systems Management (2020) 28:502–552
https://doi.org/10.1007/s10922-020-09536-x

A Holistic Framework for Virtual Network Migration to Enhance Embedding Ratios in Network Virtualization Environments

Mahboobeh Zangiabady[1] · Alberto Garcia-Robledo[2] · Christian Aguilar-Fuster[1] · Javier Rubio-Loyola[1]

Received: 31 July 2019 / Revised: 22 March 2020 / Accepted: 10 April 2020 / Published online: 28 April 2020

Abstract
Network virtualization is a promising technology for overcoming Internet ossification by enabling multiple Virtual Networks (VNs) to coexist on a shared substrate network. One critical aspect in NV environments is the capability of operators to allocate resources in the substrate network to support VNs in an optimal manner. This is known as Virtual Network Embedding (VNE). In the same context, online VN migration is the process meant to re-allocate components of a VN in real-time and seamlessly to the end-users. Although progress has been made to address VN migration, there has been little investigation on integral migration approaches assessed under different VN environment conditions. The main contribution of this paper is a VN migration framework that addresses the online VN migration problem holistically, namely considering different aspects that affect the efficiency of resource (re)allocations and the VNE acceptance ratios, such as migration policies, trigger conditions, and the CPU capacity requirements for Intermediate Substrate Nodes. An evaluation methodology is developed for analyzing the performance of the proposed framework on substrate infrastructures of different sizes and densities. Extensive software simulations on substrate networks of varying size (50 to 250 nodes) and link density (0.06 to 0.6) discover the migration-oriented parameters that contribute to enhance VNE up to 18.7%. We also compare the framework performance against two state-of-the-art mechanisms that improve online VNE while looking for VNE solutions and observed acceptance ratio enhancements up to 3× higher when using our framework on a physical network with 100 nodes and a density of 0.06.

Keywords QoS · VNE acceptance ratio · Virtual network embedding · Virtual link migration problem · Software defined networking

✉ Mahboobeh Zangiabady
me28zangiabady@gmail.com

Extended author information available on the last page of the article

1 Introduction

The Internet is evolving and new technologies to meet the requirements of current and future services are currently under research, development, and deployment. One of these technologies is network virtualization (NV), which is considered a key enabler of the Future Internet as it allows building and operating multiple Virtual Networks (VNs) on top of substrate infrastructures [1]. A VN is a collection of virtual nodes and virtual links (VLs) mapped on a substrate network (SN) [2].

The process of mapping each virtual node to one substrate node and each virtual edge to one or more substrate edges is referred to as the Virtual Network Embedding (VNE) problem. Due to its importance for NV, the VNE problem has attracted a lot of attention from the research community in the last decade [1, 3]. The main reason is that effective embedding is pivotal to exploiting the network resources efficiently and consequently to increase the business value of the infrastructure providers (InPs) [4]. Effective VNE considers both the VN resource requirements as well as the substrate network's available capacity at the time of mapping.

In the vast majority of real-world scenarios, VNE has to be addressed as an online problem. That is, VN requests (VNRs) are not to be known in advance and VNs are mapped to the substrate network as they arrive [5]. Online VNE is typically a much more difficult problem to solve because the embedding algorithm has little (if any) visibility into the request arrivals [6]. The arrival time, resource requirements, and lifetime of the VNs are unknown.

Even if the embedding algorithms are optimal under the current network mapping topology, there is no guarantee that the mapping structure will be always optimal. In the long run, the performance of the VNE algorithms may lead to cases where the physical infrastructure or sections of the physical infrastructure may be underutilized, over-utilized, or combinations of both [7]. This aspect can turn into a problem that can affect the *VNE acceptance ratio* and produce QoS degradation in VNs. In addition, recent analysis has demonstrated that the VN acceptance ratio decreases drastically [8] with practical instances, i.e. instances of large size substrate and VNs. It is therefore necessary to design mechanisms to overcome these limitations.

In this context, the reallocation/reconfiguration of substrate resources has been presented as a potential management tool to enhance resource utilization [9], and in consequence acceptance ratios. However, this concept, also referred to as *VN migration*, has received relatively little attention so far despite its potential to increase the business value of infrastructure providers. Most of the works on VN migration mostly focus on the practical aspects needed to reconfigure the physical elements of the substrate network elements [9–12], and a few works have considered the reallocation of resources, majorly as an amendment of specific embedding algorithms [5, 6, 13, 14]. The online VN migration problem is complex because it involves multifactorial and inter-related aspects, such as the features of the substrate networks, the features of the VNRs, and additive and non-additive constraints.

Notwithstanding the importance of the VN migration problem, there is a lack of a comprehensive study of online virtual network migrations, especially

considering different virtual network environment conditions. This is an important open issue as to date it is still unknown which parameters, configurations, and network conditions have a significant impact on the effectiveness of online resource re-allocations in VN environments. Currently, there are no processes and robust methodologies that allow both the migration of virtual resources and updating and deployment of new services while minimizing service disruption. To the best of our knowledge, an overall approach addressing the VN migration problem considering the above inter-related aspects is still missing in the literature. Such an approach and its assessment would provide researchers and practitioners with relevant information to identify the considerations and conditions that should be addressed carefully in NV environments.

Finally, the allocation of CPU capacity for Intermediate Substrate Nodes (ISNs) onto which VLs are mapped has been largely ignored in literature. However, CPU utilization is not negligible and there are differences in the performance of the VNE problem while considering the ISNs' CPU utilization due to the VLs' forwarding rate requirements [15].

Addressing the lack of management mechanisms to systematize the reconfiguration of VNs in favor of QoS management by exploiting a holistic approach that is thoroughly tested on different network environment conditions and that considers the CPU forwarding requirements of intermediates nodes will be the subject of this paper.

In this paper, we present a VN migration framework to systematically enhance the use of physical network resources of NV environments during online VNE, improving VNE acceptance ratios through timely re-allocation of elements of VNs according to constraints on QoS parameters set by InPs. The proposed framework takes the shape of a specification of the components, their organization and the underlying principles of a VN migration approach that holistically considers the different aspects that affect the efficiency of resource (re)allocations of VNs in physical infrastructures. To this end, we define migration trigger conditions and strategies to select the physical resources where virtual elements would be re-allocated.

Considering that network link failures occur about 10 times more than node failures [16], a pivotal element of our approach is a strategy to address the Virtual Link Migration Problem (VLMP), which is a concept we propose to drive VN migrations systematically by considering QoS additive and non-additive constraints of the VN elements subject of migration. In this paper we develop an efficient linear-time heuristic that solves the VLMP considering both additive (delay and hop count) and non-additive (BW and CPU) constraints while considering the CPU forwarding requirements of ISNs.

Due to the online nature of our problem, we comprehensively evaluate the proposed framework under different online VNE conditions and validations on small to large-scale substrate networks with different sizes (50 to 250 nodes) and link densities (0.06 to 0.6) through extensive software simulations, observing VNE acceptance ratios enhancements of up to 18.7%. We also compare the performance of the migration framework against two state-of-the-art mechanisms aimed at enhancing the performance of online VNE to further show the benefits of our migration approach.

The rest of the paper is organized as follows. Section 2 briefly reviews relevant works VNE enhancement without migrating resources and the VNE enhancement by migrating resources. Section 3 develops the main processes of the migration framework. Section 4 elaborates on the online VN migration process and the VLMP VN migration approach central to this study. Section 5 describes the assessment methodology defined to carry out experiments. Section 6 presents the results of the framework performance assessment for each stage of the assessment methodology, the statistical analysis of the obtained results and a comparative analysis. Section 7 summarizes the contributions and findings of this work, and proposes an architecture to deploy the presented framework in SDN environments. Finally, Sect. 8 gives concluding remarks and future work in the area.

2 Related Work

This section describes works related to this paper, grouping them by their approach: (i) VNE enhancement without migrating resources and (ii) VNE enhancement by migrating resources. We present a summary of the works described in this section, their limitations and how our paper advances the state of the art in their respective areas.

2.1 VNE Enhancement Without Migrating Resources

The aim of the VNE problem is to optimize the use of physical resources while allocating resources for VNs. The research community has developed mechanisms aimed at enhancing the performance of online VNE without reallocating these resources. They are mechanisms developed to optimize the performance of the VNE process while looking for online VNE solutions. This section describes recent works following this approach.

Authors in [17] develop a VNE approach based on a threat-evaluation step that identifies Virtual Machine (VM) vulnerabilities through periodic evaluation. The authors propose an offline VNE heuristic that considers risky VMs and security VMs for the virtual node mapping stage. A VL mapping step based on the risk-aware VMs' mapping step completes the VNE process.

Authors in [18] exploit features of the SN to produce a more efficient VNE approach that increases the utilization of substrate resources. The authors make use of the node degree and clustering coefficient information of the SN to define importance levels for substrate nodes, which are proved to have more potential to embed virtual nodes of VNRs. They claim that their approach increases VNE efficiency with respect to state-of-the-art works in terms of revenue-to-cost ratio and VNE acceptance ratio. Authors in [19] evaluate the relationship between the quality of VN mappings and the SN topology. An exact algorithm was used to address the VNE problem finding that SN connectivity is a critical aspect that affects the performance of the VNE solutions with highly connected networks obtaining lower VNE rejection rates in the considered scenarios.

Authors in [20] propose a constraint management approach that grades the quality of VNE candidate solutions according to the degree of fulfillment of their constraints and exploits this information to drive the metaheuristics to more promising regions of the search process enhancing their performance. Through simulation and formal statistical analysis, the authors' approach enhances the VNE acceptance ratio practically at no time overhead.

As we mentioned earlier, the VNE problem involves mapping each virtual node to one substrate node and each VL to one or more substrate links. VLs are mapped onto one or more substrate links in which, for the latter case, substrate links are connected through ISNs.

Unlike the work presented in this paper, in all the above works addressing online VNE the allocation of CPU capacity for ISNs onto which VLs are mapped has been dismissed. In general, the CPU capacity required by ISNs to meet the QoS rate requirements of VLs for the VNRs is largely disregarded in the models and objective functions optimized in all previous works addressing the VNE problem. To the best of our knowledge, the only work that has considered the ISNs in VNE is [20].

2.2 VNE Enhancement by Migrating Resources

VN migration is a potential management tool to enhance resource utilization. However, this concept has received relatively little attention so far despite its potential to increase the business value of InPs. Most of the works on VN migration have focused on the algorithmic aspects intended to reconfigure substrate network elements supporting virtual elements and networks.

Authors in [9] proposed an approach to migrate virtual routers among physical routers without disrupting the flow of traffic, obviating the need to reconfigure the virtual routers and avoiding routing-protocol convergence delays in the virtual network. Authors in [11] proposed the VN clone migration concept as an alternative to re-embed virtual nodes and links from substrate elements prone to failure. The authors focus on practical aspects and substrate resource capacity requirements to achieve timely and effective migrations in terms of reduced VN downtime and resilience to fault events.

Authors in [21] propose an Integer Linear Programming (ILP) formulation of a VNE reallocation problem denominated ReViNE, and provides a simulated annealing-based heuristic to minimize the number of over utilized substrate links and total bandwidth cost on the substrate network. In [22] authors also propose an ILP formulation of the VN reconfiguration problem in order to maximize the gain of migrations. Authors in [23] address a specific virtualization problem in optical datacenters: reduce the energy consumption by means of a method for constructing a VN by setting the parameters of its topology. They propose a reconfiguration approach called VNR-DCN for reconfiguring the VN within a short period of time with the objective of reducing the number of links based on the instantaneous traffic load.

Work in [6] considers the re-optimization of the mapping of existing virtual links by selecting new underlying paths or by adapting the splitting ratios for the existing paths. In [5] authors propose a heuristic algorithm to address online VNE. The

authors propose an option to migrate virtual links with the intention to maximize the number of coexisting VNs in a substrate network and hence to increase the revenue of the InPs. However, the proposed migration algorithm in [5] does not consider virtual node remapping due to its low complexity requirements.

Authors in [14] propose an algorithm for VNE, namely the VNE-NFL with a re-optimization function. The VNE-NFL approach uses ILP to find embedding solutions where the re-optimization approach is added as a constraint of the VNE algorithm. The authors however suggest that the reductions are higher when the physical network is not loaded and lower when the network gets loaded. This aspect is of relative importance, especially because the validations and simulations were restricted to small sized substrate network instances in all cases not exceeding 50 nodes.

In [7] authors propose an embedding algorithm with reconfiguration options that prioritize the migration of a selection of VNs mapped to the most stressful physical nodes and links. A relevant aspect of this work is that it avoids periodic re-configurations, which can be undesirable for efficient long-run resource allocations.

The authors in [7] propose a strategy driven by two main aspects: i) the identification of a set of critical substrate nodes and links (highly utilized), and ii) the reconfiguration of selected VNs according to a reconfiguration cost function. Authors in [24] propose a survivable virtual network embedding algorithm based on node migration and link remapping (SVNE-NOLR), as an alternative to other protection mechanisms that assign redundant substrate resources with poor effective resource utilization.

In [12] authors investigated the survivability problem in a network virtualization environment and provided a topology-aware remapping policy to deal with substrate node failures. Their approach consists of a hybrid strategy that proactively reserves resources and reactively triggers a fast remapping mechanism using the reserved resources on substrate failures.

The work in [25] is a method to prevent covert channel attacks that could compromise private information in cloud computing and data center environments. The authors developed a dynamic migration of virtual links method as a mechanism to satisfy security requirements of tenants. The proposed method has the primary goal of minimizing the migration cost associated with the virtual links migration, optimizing the utilization of the physical resources. One core limitation of the work in [25] is that the dynamic migration of virtual links method does not consider the traffic rate requirements of the virtual links subject of migration, and that is a critical element that affects VN acceptance ratio in VNE environments.

In [26] authors propose a virtual network reconfiguration scheme that aims to balance the load on the substrate network by dynamically reconfiguring the embedding of both virtual nodes and links. The reconfiguration scheme includes two stages. The first stage identifies the virtual nodes to be migrated and the second stage identifies the physical nodes for the migrated virtual nodes. The authors in [26] solved the first stage by proposing a linear programming method and for the second stage a Markov-chain based approach was used to select the physical nodes where virtual nodes would be migrated.

In [27] authors propose an approach to reconfigure the VNs to solve the fragmentation problem. The authors focus on when and how migration should

take place to minimize the migration cost considering aspects like restrictive delay, location and resilience constraints. Authors in [28, 29] provide different approaches to orchestrate the migration process, considering softwarized control paradigms, such as SDN. Their proposed approach concentrates on how the migration is held in a virtualized SDN environment.

Switch migration from over-utilized controllers to under-utilized SDN controllers is studied in [30, 31]. Authors in [30] present a heuristic to solve the switch migration problem. They propose a "shaking" solution that involves shift and swap moves wrapped in a search scheme that evaluates the benefits to both the immigration and outmigration controllers. On the other hand, authors in [31] formulate the switch migration problem as a mixed-ILP, and study their algorithm in scenarios of homogeneous and heterogeneous controllers.

Authors in [32] studied a security problem during migration problems. They propose monitoring techniques in order to discover security attacks on the migration process. In this regard, they designed a Markov Decision Process that let the service provider select where to deploy the monitoring resources based on the attacker's strategies.

The idea of VN mapping problem considering user mobility in 5G metro/access networks is studied in [33]. They designed a VN migration algorithm that supports user mobility. The migration process triggers by anticipating the current location of the virtual node and choosing the goal substrate node where the minimum number of re-mapping has happened.

Energy consumption in current optical networks has been studied in [34]. They proposed energy-aware VN migration algorithms that focus on when to trigger migration, where to migrate virtual resources, and how to perform migration of virtual networks.

In [35] a reactive reconfiguration technique is triggered to migrate the running embedded VNs from old physical resources to the new physical resources in order to avoid rejection of the new coming VNRs. They formulated the proposed approach as an ILP problem which assures that the new coming VNRs can definitely be embedded on a physical network with minimum resources.

A topology-aware mechanism is discussed in [36] in order to detect overutilized physical nodes and links. They designed a series of algorithms in order to reconfigure rejected VNRs after solving the problem of overutilized physical resources. In [37] a simulated annealing algorithm is proposed for VN migration. Their approach focuses on balancing the traffic over the substrate network which resulted in decreasing the number of physical nodes and links bottlenecks.

Regarding our previous work, in [38] we introduce the concept of self-adaptive VN migration triggering, in order to enable a migration agent to learn the critical moments when VN migrations are likely to be profitable in the long term. To this end, a reinforced learning approach is utilized and configured to have as actions the triggering of the migration of physical paths. The concept of maximum substrate graph spanning tree is also introduced to quantify how strong a substrate graph is in terms of its capacity to remain connected after link embedding link pruning stages, an aspect that severely impacts VNE acceptance ratio. With this

method, the number of VN migrations is significantly reduced while simultaneously considering VNR acceptance ratios.

In [39] we first introduced the concept of VLMP for VN migration and performed an experimental study of different migration policies that help in systematizing QoS-oriented virtual network migration, showing that our VLMP approach is able to improve the acceptance ratio of different VNE meta-heuristics. In [40] we provide an experimental study of the VNE acceptance ratio enhancement for different migration trigger conditions on an array of substrate networks with different substrate link densities and network sizes.

Table 1 presents a comparison between the state of the art and our work in terms of different features. Table 1a shows that some works reported in the literature (e.g. [30, 34, 38]) analyse the migration performance considering both additive and non-additive QoS parameters of the virtual elements subject to reconfiguration. With this in mind, in this work we consider a heuristic to solve the VN migration problem, all in all, considering both additive (i.e. delay and hop count) and non-additive (i.e. CPU and bandwidth) QoS parameters.

Table 1b shows that, unlike the work presented in this paper, in most previous works addressing online VNE and VN migration the allocation of CPU capacity for ISNs onto which VLs are re-mapped has been dismissed. Hence, without loss of generality, all these solutions would embed a non-negligible number of virtualized networks that, when operative, would suffer from potential QoS degradations due to the depletion of CPU capacity in ISNs. For the objectives of this paper, this is an important drawback of all VNE approaches developed so far, especially because the migration of virtual components within elements of a SN should guarantee the fulfillment of the QoS requirements of the VNs. To tackle this problem, in this paper we consider the CPU capacity that ISNs should allocate to meet the requirements of the VNRs.

Table 1c, d show that the evaluations in most of previous works are limited to isolated trigger parameters and network conditions. Some works (e.g. [7, 25, 27, 37]) trigger re-optimization periodically, which is a fixed parameter that disregards the performance of the substrate network, or the performance of the embedding approach. In order to provide an integrated approach towards more efficient resource (re)allocations, in this paper we propose a migration framework that includes an array of four trigger conditions that encompasses the key ideas in previous works for both periodic and reactive triggering, as discussed in Sect. 4.1.

Table 1e shows that few solutions ([39, 40]) explicitly enable the InP to choose from alternative objective cost functions that involve one or more QoS parameters for optimization. This is a common drawback that hinders InPs from being able to suit reported migration works to their particular needs. In this work we propose a VN migration framework that is flexible enough to allow the InP to select from an array of migration policies that combine different sets of QoS constraints/thresholds to drive VN migrations, while simultaneously achieving efficient utilization of network resources, as discussed later in Sect. 4.5.

Table 1f and g show that most works share two limitations, one being the strong dependency of the reconfiguration procedures with the embedding approach and the other being the lack of evaluations under different conditions and different substrate

Table 1 In the table a work is checked in column (a) if it considers both additive (e.g. delay, jitter, hop count) and non-additive (e.g. CPU capacity, memory, bandwidth) QoS parameters in the SN model

Ref.	Year	(a) Considers additive and non-additive QoS params.	(b) Considers capacity of ISNs	(c) Considers migration triggering	(d) Considers multiple triggers	(e) Considers multiple migration policies	(f) Assessed by enhancing multiple VNE algorithms	(g) Assessed on SNs with varying density and size	(h) Assessed on an SDN environment
Our work	2020	✓	✓	✓	✓	✓	✓	✓	×
[34]	2020	✓	×	✓	×	×	×	×	×
[32]	2019	×	×	×	×	×	×	×	✓
[27]	2019	×	×	✓	✓	×	✓	×	×
[28]	2019	×	×	✓	×	×	×	×	✓
[30]	2019	✓	×	✓	×	×	×	×	✓
[33]	2019	×	×	✓	×	×	×	×	×
[38]	2019	✓	×	✓	✓	×	×	✓	×
[31]	2019	✓	×	×	×	×	×	×	✓
[26]	2018	×	×	✓	×	×	×	×	×
[21]	2017	×	×	✓	×	×	×	×	×
[25]	2017	×	×	✓	×	×	×	×	×
[11]	2016	×	×	✓	✓	×	×	×	×
[39]	2016	✓	✓	×	×	✓	✓	×	×
[40]	2016	✓	✓	✓	✓	✓	✓	✓	×
[23]	2014	✓	×	×	×	×	×	×	×
[24]	2014	×	×	✓	×	×	×	×	×
[12]	2013	×	×	✓	×	×	×	×	×
[14]	2013	×	×	×	×	×	×	×	×
[22]	2013	×	×	✓	×	×	×	×	×
[5]	2012	✓	×	✓	×	×	✓	×	×

Table 1 (continued)

Ref.	Year	(a) Considers additive and non-additive QoS params.	(b) Considers capacity of ISNs	(c) Considers migration triggering	(d) Considers multiple triggers	(e) Considers multiple migration policies	(f) Assessed by enhancing multiple VNE algorithms	(g) Assessed on SNs with varying density and size	(h) Assessed on an SDN environment
[37]	2012	×	×	✓	×	×	×	×	×
[35]	2012	×	×	✓	×	×	×	×	×
[36]	2010	×	×	✓	×	×	✓	×	×
[6]	2008	✓	×	✓	×	×	×	×	×
[9]	2008	×	×	✓	×	×	×	×	×
[7]	2006	×	×	✓	×	×	×	×	×

It is checked in column (b) if it considers the resources (e.g. CPU utilization) needed in ISNs to forward packets. It is checked in column (c) if either proposes or discusses the considered VN migration trigger condition(s) (e.g. periodical triggering, threshold-based triggering). It is checked in column (d) if it considers more than one VN migration trigger condition. It is checked in column (e) if it explicitly enables the InP to choose from alternative objective cost functions that involve one or more QoS parameters for optimization. It is checked in column (f) if its performance is assessed by enhancing more than one stand-alone VNE algorithm. It is checked in column (g) if its performance is assessed by scaling both the size (number of substrate nodes) and the density (node connection probability) of SNs. It is checked in column (h) if its performance is assessed on either a simulated or an actual SDN testbed (with either SDN router or switch controllers)

network sizes. Moreover, none of the previous works addressing VN migration (besides our works in [38] and [40]) have evaluated their VN migration approaches in medium- to large-sized substrate networks with different link densities. In contrast, our migration approach can be customized to any online VNE approach and is thoroughly assessed on various low- to large-scale physical networks (50 to 250 nodes) with varying density (0.06 to 0.6), as demonstrated in Sect. 6.

Finally, Table 1.h shows that few works are assessed on either a simulated or an actual SDN testbed with SDN router or switch controllers. Unlike studies in [28, 30–32], in our work we do not conduct such an assessment. This represents an important drawback. Nonetheless, in Sect. 7.2 we propose a SDN VN migration architecture that enhances previously reported SDN concepts with the VN migration concepts described in this paper to facilitate the adoption of our approach as a component of real-world SDN solutions.

Overall, most works deal with algorithmic aspects and primarily discuss the question of "how to migrate". Nevertheless, besides our work in [40], none of these works discusses holistically the critical questions of "when", "what to migrate", and "from which physical resources to which physical resources". There is a lack of a holistic framework for VN migration to enhance embedding ratios in NVE.

In this paper, we embody our prior work in [39, 40] and extend our contribution by presenting a methodologically assessed VN migration framework that includes different migration trigger conditions, strategies to select the physical resources where virtual elements would be re-allocated while considering QoS additive and non-additive constraints of the VN elements subject to migration.

As shown in Table 1, to the best of our knowledge, the assessment of our framework is the first study that considers multiple VN environment conditions: different VN migration policies modeling both additive and non-additive QoS parameters, different VN migration triggers, and a methodological assessment on SNs of different scales and densities while considering the requirements of ISNs. This study provides a detailed analysis of the impact of these interrelated aspects and identifies the parameters and strategies that should be considered more carefully while addressing this complex problem.

3 Virtual Network Migration Framework

As we mentioned earlier, the majority of works on VN migration has been mostly focused on the practical aspects needed to reconfigure the physical elements among the SN elements, and very few works have considered the reallocation of resources, majorly as an amendment of specific embedding algorithms. However, online VN migration is a complex problem that has several interrelated aspects that deserve special attention:

1. The actual reconfiguration of substrate resources in most real-world scenarios has to take place in runtime and as such, it is important to evaluate different VN migration triggers and compare their performance in favor of efficient resource utilization. This is a critical aspect because, on the one hand, inappropriate trig-

gers may result in unnecessary migrations and on the other hand, late migrations may cause QoS degradation for the services deployed on the VNs [27, 38].
2. The search of physical resources (substrate nodes and links) where virtual elements would be re-allocated must be fast due to the online nature of the VN migration problem. In this respect, it is important to analyze different substrate resources' selection strategies and evaluate their effects on the resource allocation efficiency.
3. The virtual elements subject to migration have additive and non-additive QoS constraints [40]. In this respect, it is worth analyzing the effect of such constraints' fulfillment on the performance of the VN migration problem.
4. VLs are mapped onto one or more substrate links in which substrate links are connected through ISNs. Current VNE and VN migration approaches ignore the CPU capacity that ISNs require to forward traffic on the VLs mapped onto them while computing the VN embedding cost. However, as demonstrated in our previous published work [15], there are differences in the performance of the VNE problem while considering the ISNs' CPU utilization due to the VLs' forwarding rate requirements. Therefore, it is relevant to consider the requirements of ISNs in VNE and VN migration approaches.
5. Given that NV technology will probably scale up to Internet-size network infrastructure [41, 42], it is necessary to analyze the VN migration problem in practical scenarios, namely on medium- to large-sized substrate networks (e.g. of more than 100 nodes).

The requirements and implications of online VN migration to guarantee the required levels of QoS of VNs and improve the resource utilization are still poorly explored challenges in NV environments. To address this open issue, throughout the rest of this section we define a novel VN migration framework and define the context and the scope of the problems addressed in this paper. The various elements of the proposed online VN migration framework are depicted in Fig. 1 for which a description is provided hereafter.

3.1 Virtual and Physical Network Monitoring

The proposed VN migration framework addresses bi-directional embedding-migration scenarios where the time at which resource availability changes due to new VNs embedding, or when resources are released due to VNs leaving the substrate network is unknown. More importantly, in these scenarios, the way the embedding algorithm allocates substrate resources upon the arrival of VNRs is unknown as well, i.e. the migration process is decoupled from the embedding approach. In this dynamic resource allocation context, the migration process is a complex activity that relies on analysis of real-time data about the status of the substrate and the virtual networks.

Resource management coordination between VNE and VN migration is necessary to make sure that both online processes, namely embedding and migration,

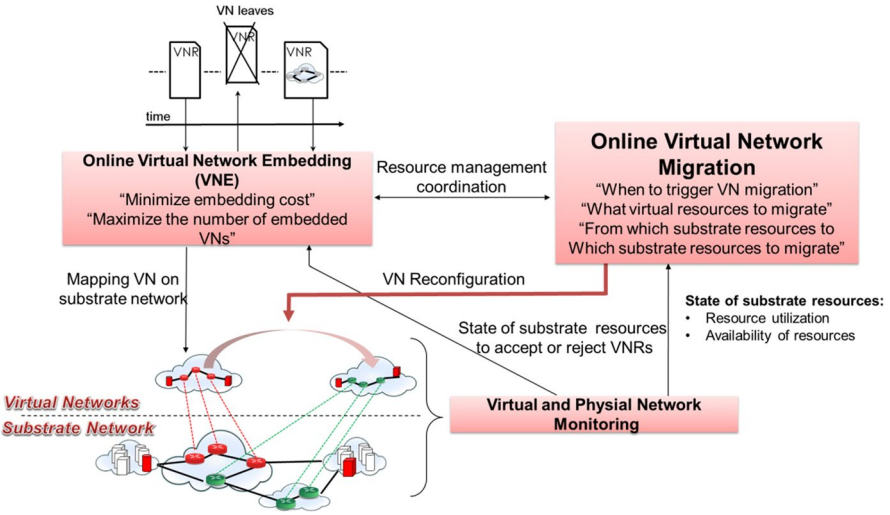

Fig. 1 The three main components of the proposed VN migration framework: the online VNE process, the online VN migration process and the virtual/physical monitoring process. The framework considers the on-line re-allocation of virtual elements between substrate resources during on-line VNE. While on-line VNE focuses on mapping VNs on the substrate network as they arrive, on-line VM migration is in charge of resource re-allocations in the substrate network in favor of more efficient resource management

resolve their objectives with stable and updated information of resource utilization in the virtual and physical networks.

With this in mind, the framework relies on monitoring infrastructures [43, 44] (see bottom part of Fig. 1) that can collect, process, and disseminate network and system information from/to the virtual and physical network entities for the management entities in real-time. The monitoring infrastructure provides to the embedding process the necessary data to accept (map and allocate resources) or to reject VNRs. Also, it provides relevant information about resource utilization and availability of resources to the VN migration process.

3.2 The Online VNE Process

The allocation of resources in the substrate network to support the virtualized networks, namely the VNE process (see upper left part of Fig. 1), is achieved online in our framework as in most real-life scenarios [1]. Online VNE maps VNs at the moment that each VN request is received, by evaluating the current state of resources in the substrate at the time of the VNR and by selecting the resources that would support its resource requirements.

In our framework we address VNE as a combinatorial optimization problem [1] solved by means of metaheuristic algorithms. Our platform includes implementations of four metaheuristics, namely Genetic Algorithm (GA) [45], Particle Swarm Optimization (PSO) [46], Ant Colony (ACO) [45], and Harmony Search

(HS) [47]. In [20] we present the implementation details of the VNE process based on the GA, PSO, and HS algorithms. It is important to note that unlike most works addressing re-configuration aspects, the migration framework presented in this paper is not restricted to these VNE metaheuristics.

The VNE algorithms exploited in our framework fall in the category of Uncoordinated VNE [1]. Uncoordinated VNE algorithms generate a VNE solution in two consecutive stages. The first stage solves virtual node embedding while the second stage solves virtual link embedding. Our VNE algorithms first find an embedding for the virtual nodes using a metaheuristic approach, and then execute a link embedding algorithm that uses the node embedding solution provided by the metaheuristic.

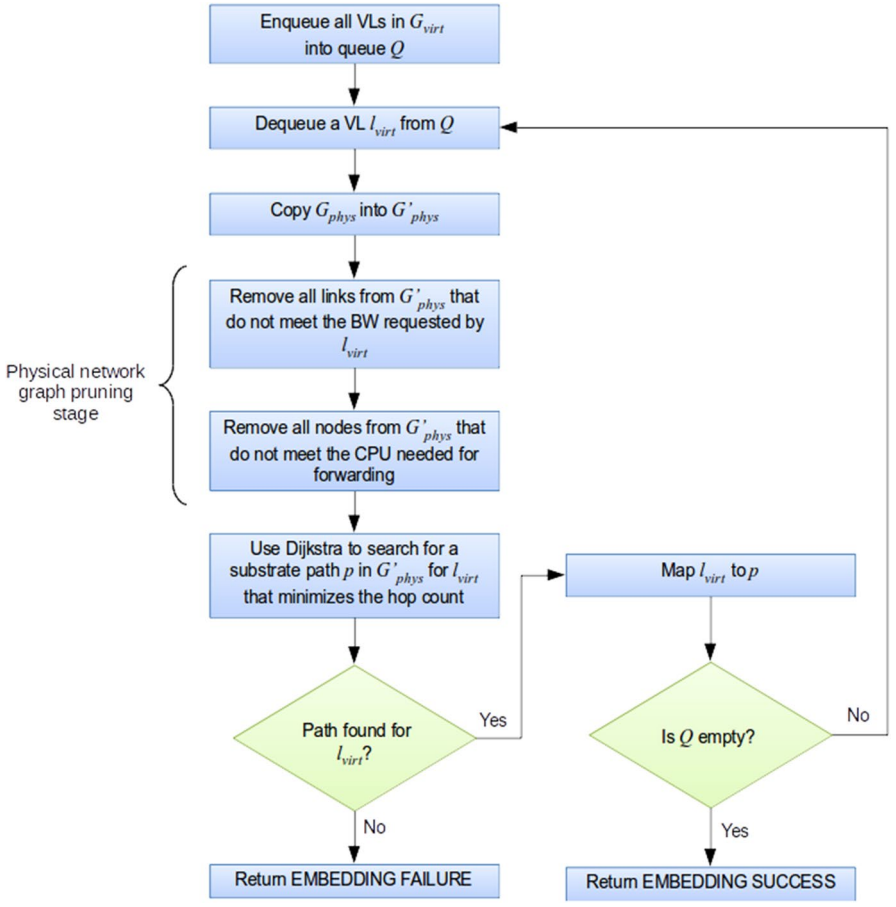

Fig. 2 Link embedding algorithm in the online VNE process. G_{phys} denotes the graph for the physical network. G_{virt} denotes the graph for the VNR to be embedded. The Dijkstra algorithm is used to assign virtual links to a substrate path. It minimizes the number of hops on the mapped paths. CPU requirements of ISNs to fulfill the forwarding rate requirements of the virtual links subject to migration are also considered

Figure 2 depicts the VNE link embedding algorithm. Similarly to [24, 45, 48, 49], after the node mapping stage the VNE metaheuristics use a shortest path algorithm to assign virtual links to a route of substrate nodes that connect the mapped virtual nodes. The shortest path algorithm minimizes the hop count of the mapped paths, as in [20]. In addition, we consider the CPU requirements of ISNs to fulfill the forwarding rate requirements of the virtual links subject of migration.

Note in Fig. 2 that the link embedding algorithm first prunes the physical graph in order to ignore: (i) nodes that do not comply with the CPU availability necessary for forwarding, and (ii) links that do not comply with the available BW requested by the virtual link. This stage is necessary to ensure that the mapped path will have enough available resources to meet the virtual link requirements.

In [50] authors notice that most of the VNE rejections are caused by bottleneck (overloaded) substrate links. In this regard, we have observed in our framework' VNE algorithms that one of the main reasons for a high embedding failure rate is the physical network graph being disconnected into different connected components during the physical graph pruning stage due to the presence of node and link bottlenecks.

When this happens, it is possible that the source and target physical nodes are separated (or even be pruned) and located at two different connected components, as depicted in Fig. 3. Node pruning due to unavailability of CPU resources for forwarding can be especially disruptive, since pruning a single node may implicate the removal of many links, as can be observed in Fig. 3b. Under these conditions, VNRs are rejected due to the unavailability of a physical path between sources and targets.

A solution is to balance resource utilization of congested physical links by means of timely path migrations in such a way that the physical network stays "strong enough" to remain connected in future link mapping attempts. Such a solution is executed by the online VN migration process described in the next section.

4 The Online VN Migration Process

While the online VNE process focuses on mapping VNs on the substrate network as they arrive, the online VN migration process (upper right part of Fig. 1) is in charge of online resource re-allocations in the substrate network in favor of more efficient resource management. In this paper, the terms reconfiguration and migration of VNs are used interchangeably to refer to runtime reallocation of substrate resources for VNs in online VNE contexts.

The framework's online VN migration process addresses three critical questions: "When to trigger VN migration?", "What virtual resources to migrate?", and "From which substrate resources to which substrate resources to migrate?" The overall steps taken to answer those questions are depicted in Fig. 4.

The third question is a critical one. Its answer implies searching for physical resources that would support the virtual resources to-be-migrated. We name this search problem as VLMP, which consists of finding a physical path from a given source to a given target such that the migration path meets three conditions: availability, feasibility and optimality (described later in this section). In our framework,

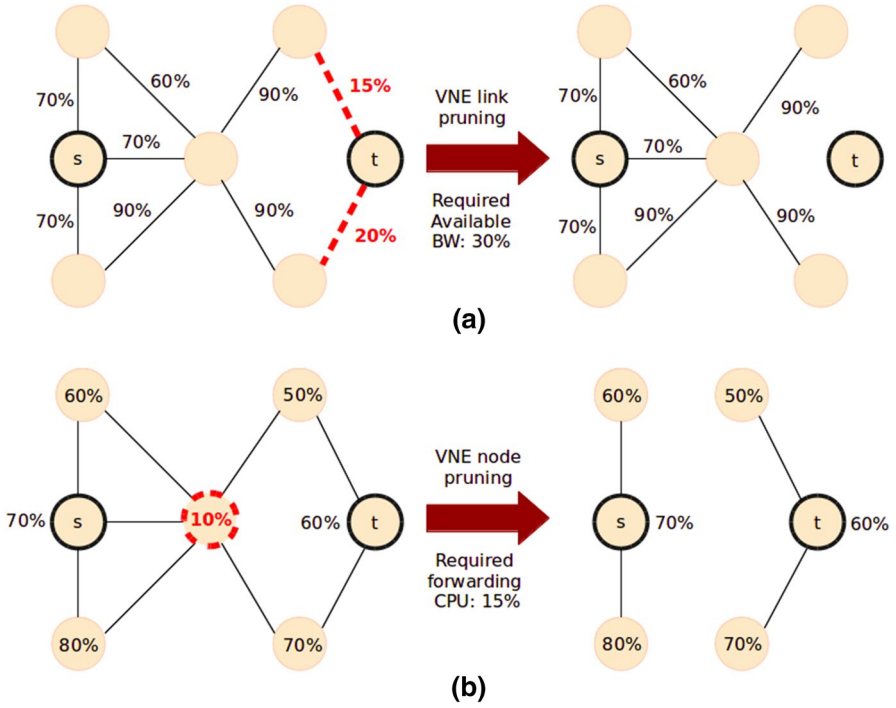

Fig. 3 Physical graph (**a**) node and (**b**) link pruning steps of the link embedding algorithm in the VNE process when trying to embed (**a**) a VL requiring an available bandwidth of 30% and (**b**) a VL requiring an available forwarding CPU of 15%. Percentages denote (**a**) available BW% and (**b**) available CPU%. Both pruning steps break down the graph into two connected components due to bottlenecks (dotted red lines), making the target node t unreachable from source node s. Note that node pruning due to unavailability of CPU resources for forwarding can be especially disruptive, since pruning a single node may implicate the removal of many links (**b**) (Color figure online)

the search process considers both the additive (delay and hop count) and the non-additive (CPU and BW) QoS parameters of the VNs supported by bottleneck resources. A detailed description of how the VN migration process solves each of the posed migration questions is provided hereafter, placing special emphasis on the description of how our approach to answer the question: "From which substrate resources to which substrate resources to migrate?" since it represents the cornerstone of the proposed framework.

4.1 When to Trigger VN Migration?

Defining when the migration should take place implies defining the conditions that must hold to activate the VN migration mechanisms.

A migration trigger is a process that decides whether to trigger or to avoid triggering the migration of VNs, with the expectation of improving the chances of future VNR acceptance and reducing the impact of excessive VNE reconfiguration.

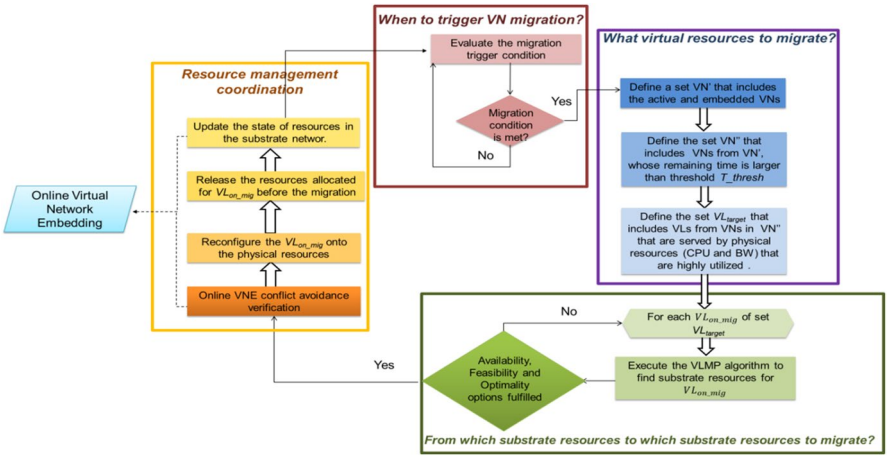

Fig. 4 Online VN migration process. It addresses the following critical questions: When to trigger VN migration? What virtual resources to migrate? From which substrate resources to which substrate resources to migrate?

Current works have addressed the migration trigger problem through approaches such as periodic and reactive triggering. As for periodic triggering, in [6] node remapping stage is triggered for a predefined number of times; making sure that each time a different bottleneck link is migrated. In [37] migration is triggered at fixed periods.

In [7] each existing VN periodically checks, with a given reconfiguration period, if it has been selected by a migration marking process. In [21] migration is triggered every 450 time units, simulating a scenario with hourly reallocation. In [25] VL migration is triggered periodically too, being the period defined in terms of the coexistence time threshold. Similarly, in [27] reconfiguration is invoked periodically for a time interval. An alternative approach to periodic migration is reactive migration, where reconfiguration does not happen periodically but only if a condition or a series of conditions are met.

In [36] a reactive approach is adopted by detecting bottlenecks and trigger migration whenever a VNR is rejected. Similarly, in [35] reconfiguration is triggered when a VNR is rejected. In [24] and [12] migrations are performed only if a substrate node fails. In [22] reconfiguration happens only when the following two conditions are satisfied: i) a VN request cannot be mapped due to lack of available resources, and ii) the cost of reconfiguration does not exceed the gain from mapping the new VN.

In [5] path migration is enabled if one of the following two conditions is satisfied: i) the number of substrate paths selected for migration is larger than the maximum number of multiple substrate paths that a virtual link is allowed to map, or ii) the sum of the resources allocated to the virtual link that is mapped to the substrate path is lower than the requested resources of the virtual link. Araújo et al. [27] also tested two reactive triggers motivated by the rejection of a VNR and the expiration of an embedded VNR.

A brief description of the considered triggers in this paper is given hereafter. A given amount of time [7] and a given amount of VN embeddings [37] are two conditions used earlier by specific VNE approaches. We included these two trigger conditions in our VN migration framework. These conditions are fixed and as such, they are easy to implement as subroutines of the embedding algorithms. However, restricting our framework to the use of these two static conditions could result in unnecessary reallocations when the demand of resources is low or highly variable, or they could produce inefficient resource allocations due to late reconfigurations.

For these reasons, we take a step forward and also include the following two reactive trigger conditions:

- *Substrate network overutilization.* This condition considers the utilization of substrate resources to trigger VN migration. Overutilization is defined through two thresholds: one threshold that defines when a network resource is considered overutilized, and another threshold defines the limit on the percentage of resources (nodes and links) out of the total ones that could be overutilized. An example of this trigger could be as follows: trigger the VN migration process when 25% of the physical nodes and links are utilized above 60% of their capacity. In this example, the threshold to define overutilized resources is set to 60% and the threshold to define the amount of overutilized resources is set to 25%.
- *Low VNE acceptance ratio.* This trigger drives migrations as a function of the overall VN embedding performance. The VNE acceptance ratio is an indicator of how well the embedding is being carried out, namely it considers the actual evaluation of the current state of resources in the substrate at the time of each VN embedding and the release of resources produced by VNs leaving the NV ecosystem. Moreover, as VNRs can be rejected due to the lack of physical resources or due to the poor performance of the embedding algorithm, this trigger condition evaluates the efficiency of the resource allocation as a whole.

Table 2 summarizes the VN migration trigger conditions considered in our framework.

4.2 What Virtual Resources to Migrate?

When a trigger condition is evaluated to true, a selection process that identifies the virtual resources with potential to be migrated is carried out. This selection process executes the following steps. First, the process defines a set VN' of VNs that are active and embedded in the substrate network. From these, the process identifies a set VN'' of VNs whose remaining time is higher than threshold T_{thresh} to discard VNs with little remaining time. This approach increases the chances for VNs with longer remaining time to be reconfigured [6]. The threshold T_{thresh} then limits the number of VNs prone to be re-configured.

Second, from the set VN'', the process defines a set VL_{target} of virtual links (VLs) hosted by highly utilized substrate nodes and links. This step relies on thresholds that define when network resources are considered overutilized. In this regard, the

Table 2 VN migration trigger conditions considered in the VN migration framework

Trigger	Condition	Configurable parameter—values used in our assessment	Features
TC1	A given amount of time [7]	Time to trigger VN migration 25-time units as in [7]	Easy to implement in VNE algorithms. Agnostic to changes in network resources and to all VN embedding aspects
TC2	A given amount of VN embeddings [37]	Every four VN embeddings	Easy to implement in VNE algorithms. Agnostic to changes in network resources and to the efficiency of resource allocations
TC3	Substrate network overutilization	Threshold to define overutilized resources Threshold to define amount of overutilized resources when the 25% of physical nodes or links are highly utilized (utilization above 60%)	Relies heavily on runtime monitoring. Agnostic to the efficiency of resource allocations
TC4	Low VNE acceptance ratio	VNE acceptance ratio threshold when the VNE acceptance ratio is below 80%	Easy to implement. Direct relationship with the efficiency of resource allocations

Trigger conditions determine when to trigger the migration process and at which conditions reconfiguration will be effective in order to avoid intensive SN instability, bandwidth overhead and eventual service disruption of reconfigured VNs

process relies on overutilization thresholds to set limits on CPU and link capacity usage of substrate nodes and links respectively.

The result at this stage is a set VL_{target} that includes virtual links whose remaining time and overutilization of their hosting substrate resources make them prone to being migrated to other substrate resources. It is worth mentioning that the actual reconfiguration (i.e. migration) of the virtual links in VL_{target} is restricted to the availability of substrate resources that can meet the constraint requirements.

4.3 From Which Resources to Which Resources to Migrate?

The next and most complex step of the migration process is to find the appropriate resources where to re-configure VLs while considering multiple constraints and a given forwarding rate. This step relies on our VLMP heuristic [39, 40], which is a linear-time algorithm designed to find a set of substrate resources (nodes and links) that could host a virtual link. The VLMP heuristic finds substrate resources that can meet the constraints of the VL prone to be migrated, and also the operator's constraints about the use of substrate resources.

4.3.1 Problem Formulation

We model a physical network as a weighted undirected graph where each node u_i and link (i,j) have associated parameters. Let $G_{phys} = \left(V_{phys}, E_{phys}\right)$ denote a physical network with physical node set V_{phys} and physical edge set E_{phys}. Let $n_{phys} = \left|V_{phys}\right|$ be the number of physical nodes in G_{phys} and $m_{phys} = \left|E_{phys}\right|$ be the number of physical links in G_{phys}. Let $= \left(u_1, u_2, \ldots, u_i\right) \in V_{phys}$, denote a simple physical path in G_{phys}. Let $d_{phys} \in [0,1]$ denote the density of the physical network G_{phys}, i.e. the ratio of the number of links in the physical network to the maximum possible number of links.

Each physical node $i \in V_{phys}$ is associated with an available CPU percentage $cpu^{avail}_{phys}(i) \in [0,100]$. Likewise, each physical link $(i,j) \in E_{phys}$ is associated with an available BW percentage $bw^{avail}_{phys}(i,j) \in [0,100]$ and a delay $dly_{phys}(i,j)$. A physical link is also associated to a cost $cost_{phys}(i,j)$. Additionally, a physical path p is associated with a hop count $hop_{phys}(p) = |p| - 1$.

We model a virtual network as a weighted undirected graph, where each virtual node v_i and virtual link (k,l) have associated requirements, namely requested CPU and requested BW, respectively. Let $G_{virt} = \left(V_{virt}, E_{virt}\right)$ denote a VNR with virtual node set V_{virt} and virtual edge set E_{virt}. Let $d_{virt} \in [0,1]$ denote the density of the virtual network G_{virt}. Each virtual node $k \in V_{virt}$ has a requested CPU percentage $cpu^{req}_{virt}(k) \in [0,100]$. Each virtual link $(k,l) \in E_{virt}$ is associated with a requested BW percentage $bw^{req}_{virt}(k,l) \in [0,100]$.

The VLMP is proposed to formulate the problem of finding substrate resources where a given VL can be remapped. Let $VL_{on\_mig}\left(k_{mig}, l_{mig}\right) \in E_{virt}$ be a VL candidate to be migrated, let expressions $cpu_{constr}\left(VL_{on\_mig}\right)$, $bw_{constr}\left(VL_{on\_mig}\right)$, $dly_{constr}\left(VL_{on\_mig}\right)$, and $hop_{constr}\left(VL_{on\_mig}\right)$ be the CPU, BW, delay and path hop

constraints of VL_{on_mig} respectively, and let $C_{VL_{on\_mig}}$=(cpu_{constr}, bw_{constr}, dly_{constr}, hop_{constr}) be the set of VL_{on_mig} constraints.

We define the VLMP as the problem of finding a physical simple path $p_{vlmp} = (s, \ldots, t) \subset V_{phys}$ from a source physical node s to a target physical node t, onto which virtual node k_{mig} is mapped, to a target physical node t, onto which virtual node l_{mig} is mapped, such that the following three conditions are met:

1. *VLMP availability condition.* It is used to search for physical resources that meet the non-additive QoS constraint requirements of VL_{on_mig}. Specifically, it guarantees that the substrate resources for $p_{vlmp} = (s, \ldots, t) \subset V_{phys}$ will have enough BW and enough CPU to forward the required rates of VL_{on_mig}.
2. *VLMP feasibility condition.* It is used to search for migration paths that both meet the VLMP availability condition and whose additive parameters (delay and hop count) do not exceed the limits on delay and hop-count defined by the operator.
3. *VLMP optimality condition.* It is used to search for a migration path that both meets the VLMP feasibility condition and that minimizes the CPU and BW link capacity usage. This condition gives the operator the flexibility to define search criteria considering CPU and BW link utilization levels.

The VLMP is an instance of the Multi-Constrained Optimal Path (MCOP) problem [51]. In general, multi-constrained path selection, with or without optimization, is an NP-hard problem [52] and QoS routing with constraints on multiple additive metrics has been proven to be NP-complete [52]. Nonetheless, the VLMP can be tackled efficiently from a graph theoretical perspective, by exploiting a Dijkstra-based shortest-path approach. In network routing terms, shortest path algorithms are designed to find a shortest path between two specified nodes of a connected weighted graph. The most widely used algorithm for this problem was proposed by Dijkstra in 1959.

In this section we show that it is possible to efficiently solve VLMP by exploiting a Dijkstra-based approach in linear time in the number of physical vertices and edges. We propose a VLMP linear-time heuristic designed as a shortest path algorithm that seeks to minimize both feasibility and optional cost functions in order to find migration paths that meet the three VLMP conditions. Such a linear time algorithm is critical for our framework to find feasible paths with available physical CPU and physical BW in reasonable time.

The three VLMP conditions together with the proposed heuristic to meet the conditions are detailed hereafter.

4.3.2 VLMP Availability Condition

The VLMP availability condition guarantees that all substrate links along $p_{vlmp} = (u, \ldots, t) \subset V_{phys}$ will have enough BW and enough CPU required by VL_{on_mig}. Forwarding rates define the processing capabilities that are needed in intermediate nodes to ensure enough CPU resources to forward data between the source

and the target physical nodes. Let $cpu^{avail}_{phys}(p_{vlmp}) = min_{u \in p_{vlmp}} \left\{ cpu^{avail}_{phys}(u) \right\}$ be the minimum available CPU all the way along $p_{vlmp} = (u, \ldots, t) \subset V_{phys}$, let $cpu_{constr}(VL_{on\_mig}) = cpu^{req}_{VL_{on\_mig}}(p_{vlmp})$ be the CPU needed to achieve the desired forwarding rate. Let $bw^{avail}_{phys}(p_{vlmp}) = min_{(i,j) \in p_{vlmp}} \left\{ bw^{avail}_{phys}(i,j) \right\}$ be the minimum available BW all the way along p_{vlmp}, and let $bw_{constr}(VL_{on\_mig}) = bw^{req}_{VL_{on\_mig}}(k_{mig}, l_{mig})$ be the requested BW by the VL_{on_mig}, $\{k_{mig}, l_{mig}\} \in E_{virt}$.

The VLMP availability condition is formally defined as follows:

$$cpu^{avail}_{phys}(p_{vlmp}) \geq cpu_{constr}(VL_{on\_mig}) \tag{4.1}$$

$$bw^{avail}_{phys}(p_{vlmp}) \geq bw_{constr}(VL_{on\_mig}) \tag{4.2}$$

4.3.3 VLMP Feasibility Condition

The VLMP feasibility condition guarantees that migration paths will comply with the resource constraints specified by the InP. Let $dly_{phys}(p_{vlmp}) = \sum_{(i,j) \in p_{vlmp}} dly_{phys}(i,j)$ be the sum of the physical delays along p_{vlmp}, and let $hop_{phys}(p_{vlmp})$ be the hop count of path $p_{vlmp} = (s, \ldots, t) \in V_{phys}$. Expressions 4.3 and 4.4 are constraints that are considered while evaluating the VLMP feasibility condition of p_{vlmp}:

$$dly_{phys}(p_{vlmp}) \leq dly_{constr}(VL_{on\_mig}) \tag{4.3}$$

$$hop_{phys}(p_{vlmp}) \leq hop_{constr}(VL_{on\_mig}) \tag{4.4}$$

Expression 4.3 evaluates whether the sum of the physical links' delays of all substrate nodes along $p_{vlmp} = (s, \ldots, t) \in V_{phys}$ is not higher than $dly_{constr}(VL_{on\_mig})$, whereas Expression 4.4 evaluates if the hop count of path p_{vlmp} is not higher than $hop_{constr}(VL_{on\_mig})$.

Paths that meet multiple constraints for additive metrics can be found by minimizing a non-linear cost function presented in [53]. This cost function approach can be used to find migration paths that meet Expressions 4.3 and 4.4. In order to enable the InPs to select a feasibility condition tailored to their needs, we designed two *feasibility cost functions*:

$$g^{dly}(p_{vlmp}) = \left(\frac{dly_{phys}(p_{vlmp})}{dly_{constr}(VL_{on\_mig})} \right)^{\lambda} \tag{4.5}$$

$$g^{dual}(p_{vlmp}) = \left(\frac{dly_{phys}(p_{vlmp})}{dly_{constr}(VL_{on\_mig})} \right)^{\lambda} + \left(\frac{hop_{phys}(p_{vlmp})}{hop_{constr}(VL_{on\_mig})} \right)^{\lambda} \tag{4.6}$$

where λ is a constant. The feasibility cost function in Expression 4.5 enables InPs to define the feasibility condition as in terms of the fulfillment of the delay constraint, whereas the feasibility cost function in Expression 4.6 defines feasibility in terms of the fulfillment of both delay and hop count constraints.

4.3.4 VLMP Optimality Condition

The VLMP optimality condition is evaluated to find, from the set of paths P_{vlmp_fea} that meet the VLMP feasibility condition, a path that uses the lowest amount of resources. This condition is formally defined as follows:

$$p_{vlmp\_opt\_cpu} = \underset{p_{vlmp\_fea} \in P_{vlmp\_fea}}{\arg\min} \left\{ h^{cpu}\left(p_{vlmp\_fea}\right) \right\} \tag{4.7}$$

$$p_{vlmp\_opt\_dual} = \underset{p_{vlmp\_fea} \in P_{vlmp\_fea}}{\arg\min} \left\{ h^{dual}\left(p_{vlmp\_fea}\right) \right\} \tag{4.8}$$

where $p_{vlmp\_opt\_cpu} \in P_{vlmp\_fea}$ is a feasible and optimal migration path w.r.t. nodes resources, $p_{vlmp\_opt\_dual} \in P_{vlmp\_fea}$ is a feasible and optimal migration path w.r.t. both nodes and link resources, $h^{cpu}\left(p_{vlmp_{fea}}\right)$ defines the cost of a feasible path p_{vlmp_fea} w.r.t. the substrate node resources' utilization along the path, and $h^{dual}\left(p_{vlmp\_fea}\right)$ defines the cost of a feasible path p_{vlmp_fea} w.r.t. both the substrate node and link resources' utilization along the path.

Let $cpu^{use}_{phys}(i) = 100 - cpu^{avail}_{phys}(i)$ be the percentage of CPU currently used by the substrate node $i \in p_{vlmp\_fea}$ and let $bw^{use}_{phys}(i,j) = 100 - bw^{avail}_{phys}(i,j)$ be the percentage of BW currently in use by the substrate link $(i,j) \in p_{vlmp\_fea}$. In order to enable the InPs to select an optimality condition tailored to their needs, we designed two *optimality cost functions*:

$$h^{cpu}\left(p_{vlmp\_fea}\right) = \sum_{i \in p_{vlmp\_fea}} \left(\frac{cpu^{use}_{phys}(i)}{cpu^{use}_{thresh}} \right)^{\beta} \tag{4.9}$$

$$h^{dual}\left(p_{vlmp\_fea}\right) = \sum_{i,(i,j) \in p_{vlmp\_fea}} \max \left\{ \frac{cpu^{use}_{phys}(i)}{cpu^{use}_{thres}}, \frac{bw^{use}_{phys}(i,j)}{bw^{use}_{thres}} \right\}^{\beta} \tag{4.10}$$

Expression 4.9 is an optimality cost function that models the cost of a feasible path p_{vlmp_fea} as proportional to the sum of the CPU in use of the nodes along the path. The optimality cost function in Expression 4.10 models the cost of a feasible path as a function of both, CPU and BW in use of the nodes and links along p_{vlmp_fea}. The resources in use are normalized by usage percentage thresholds cpu^{use}_{thresh} and bw^{use}_{thresh} to express when the resources are highly-utilized. The constant β highlights paths that contain bottlenecks, i.e. paths having nodes and/or links whose resources in use are higher than the thresholds, penalizing and

avoiding them during the search process. We have observed that a value of $\beta = 5$ gives good performance results.

4.4 The VLMP Heuristic

We now provide an algorithmic description of the proposed VLMP heuristic, which is listed in Algorithm 1. The objective of Algorithm 1 is to find a migration path from a given source s to a given target t in the physical network graph G_{phys} that meet the three VLMP conditions.

First, Expressions 4.1 and 4.2 of the VLMP availability condition are met by pruning the physical network G_{phys} to: (i) discard any physical node whose available CPU is lower than $cpu_{constr}\left(VL_{on\_mig}\right)$; and (ii) discard any physical link whose available BW is lower than $bw_{constr}\left(VL_{on\_mig}\right)$. In Algorithm 1, procedure *Prune* prunes the physical network graph to remove such nodes and links. The result is a pruned physical network $G'_{phys} = \left(V'_{phys}, E'_{phys}\right)$, $V'_{phys} \subseteq V_{phys}$ and $E'_{phys} \subseteq E_{phys}$, where all paths between s and t meet the VLMP availability condition.Next, the VLMP heuristic tries to meet the VLMP feasibility condition by running an adaptation of the Reverse Dijkstra algorithm [52], which searches for a feasible path in the backward direction, i.e. from the target substrate node t to the source substrate node s. For this purpose, our Reverse Dijkstra minimizes the linear version ($\lambda = 1$) of the feasibility cost function selected by the InP from Expressions 4.5 or 4.6 on the pruned graph G'_{phys}.

In Algorithm 1, procedure *ReverseDijkstra* minimizes the feasibility cost function in Expression 4.6 by using the edge relaxation procedure *ReverseDijkstraRelax,* listed in Algorithm 2. For each node $x \in V'_{phys}$, *ReverseDijkstraRelax* maintains the following node labels: $x.r$ denotes the cost of the shortest path from x to t, $x.R_{dly}$ denotes the accumulated delay along the path from x to t, and $x.R_{hop}$ denotes the accumulated hop count along the path from x to t.

```
procedure VLMP_H(G_phys, s, t)
    G'_phys ← Prune(G_phys)
    ReverseDijkstra(G'_phys, t)
    if s.r > 2 then
        return failure
    end if
    LookAheadDijkstra(G'_phys, s)
    if t.G_dly ≤ dly_constr and t.G_hop ≤ hop_constr then
        p_vlmp_opt_dual ← RetrievePath(t.π)
        return p_vlmp_opt_dual
    end if
    return failure
end procedure
```

ALGORITHM 1. The VLMP heuristic.

procedure REVERSEDIJKSTRARELAX(u, v)
  if $u.r > \frac{v.R_{dly}+dly_{phys}(u,v)}{dly_{constr}} + \frac{v.R_{hop}+1}{hop_{constr}}$ **then**
    $u.r \leftarrow \frac{v.R_{dly}+dly_{phys}(u,v)}{dly_{constr}} + \frac{v.R_{hop}+1}{hop_{constr}}$
    $u.R_{dly} \leftarrow v.R_{dly} + dly_{phys}(u, v)$
    $u.R_{hop} \leftarrow v.R_{hop} + 1$
  end if
end procedure

ALGORITHM 2. The relaxation procedure of our Reverse Dijkstra algorithm.

Next, the VLMP heuristic tries again to meet the VLMP feasibility condition, this time by running an adaptation of the Look-ahead Dijkstra algorithm [52], which searches for a feasible path in the forward direction, i.e. from the source substrate node s to the target substrate node t.

Our Look-ahead Dijkstra minimizes the non-linear version ($\lambda > 1$) of the feasibility cost function selected by the InP from Expressions 4.5 or 4.6 on the pruned graph G'_{phys}. Setting $\lambda > 1$ improves the chances of finding a feasible path if Reverse Dijkstra failed to find it [52]. Simultaneously, our Look-ahead Dijkstra tries to meet the VLMP optimality condition. For this purpose, it uses the optimality cost function selected by the InP from Expressions 4.9 or 4.10 in order to find an alternative path that is feasible and that *probably* reduces the resources in use over the path found by Reverse Dijkstra.

In Algorithm 1, procedure *LookAheadDijkstra* minimizes the feasibility cost function in Expression 4.6 by using the edge relaxation procedure *LookAheadDijkstraRelax*, listed in Algorithm 3. For each node $x \in V'_{phys}$ *LookAheadDijkstraRelax* maintains the following node labels: $x.c$ denotes the accumulated optimality cost along the path from s to x, $x.g$ denotes the cost of a foreseen complete path that goes from s to t via x, $x.G_{dly}$ denotes the accumulated delay along the path from s to x, x denotes the accumulated hop count along the path from s to x, and $x.\pi$ denotes the predecessor node of x on the path. Note that the optimality cost function in Expression 4.10 is used to calculate label $x.c$.

In each edge relaxation, the algorithm uses the preference rules of procedure *PreferTheBest*, listed in Algorithm 3, to identify whether there is another foreseeable feasible path with lower resources in use, or at least with more chances to be feasible if the previous found path is not feasible. For this purpose, *PreferTheBest* uses both feasibility and optimality costs calculated so far.

Finally, if a feasible path is found after running *LookAheadDijkstra*, the VLMP heuristic uses the procedure *RetrievePath* to backtrack the final path from the target node t by using the predecessor information stored in label $x.\pi$.

procedure LOOKAHEADDIJKSTRARELAX(u, v)
  $tmp \leftarrow$ new temporary node
  $tmp.c \leftarrow u.c + \max\left\{\frac{cpu^{use}_{phys}(v)}{cpu^{use}_{thresh}}, \frac{bw^{use}_{phys}(u,v)}{bw^{use}_{thresh}}\right\}^{\beta}$
  $tmp.g \leftarrow \left(\frac{u.G_{dly}+dly_{phys}(u,v)+v.R_{dly}}{dly_{constr}} + \frac{u.G_{hop}+1+v.R_{hop}}{hop_{constr}}\right)^{\lambda}$
  $tmp.G_{dly} \leftarrow u.G_{dly} + dly_{phys}(u, v)$
  $tmp.G_{hop} \leftarrow u.G_{hop} + 1$
  $tmp.R_{dly} \leftarrow v.R_{dly}$
  $tmp.R_{hop} \leftarrow v.R_{hop}$
  if PREFERTHEBEST(tmp, v)=tmp **then**
    $v.c \leftarrow tmp.c$
    $v.g \leftarrow tmp.g$
    $v.G_{dly} \leftarrow tmp.G_{dly}$
    $v.G_{hop} \leftarrow tmp.G_{hop}$
    $v.\pi \leftarrow u$
  end if
end procedure

procedure PREFERTHEBEST(a, b)
  if $a.c < b.c$
    and $a.G_{dly} + a.R_{dly} \leq dly_{constr}$
    and $a.G_{hop} + a.R_{hop} \leq hop_{constr}$
    then return a
  if $a.c > b.c$
    and $b.G_{dly} + b.R_{dly} \leq dly_{constr}$
    and $b.G_{hop} + b.R_{hop} \leq hop_{constr}$
    then return b
  if $a.g < b.g$
    then return a
  return b
end procedure

ALGORITHM 3. The relaxation procedure of our Look-ahead Dijkstra algorithm.

Note that the VLMP heuristic described above only requires the invocation of two Dijkstra searches, sequentially. Thus, its time complexity is $O\left(m_{phys} + n_{phys}\log n_{phys}\right)$. In other words, it only takes linear time in the number of physical links to obtain both, available, feasible and optimal migration paths. Due to the time constraints of online resource reconfiguration, having a low time-complexity migration algorithm becomes an important requirement for timely delivering and maintaining appropriate levels of QoS. This is particularly relevant in scenarios that involve large physical networks that serve a large number of VNRs arriving at a high rate.

4.5 Migration Policies

The introduction of the VLMP conditions previously described is intended to give InPs the flexibility to manage QoS levels of the VNs deployed on the substrate, and if needed, to make counter-offers when the required levels of QoS cannot be maintained. The conditions and cost functions of the three VLMP conditions can be combined to define migration policies (MPs) to drive the search of resources where to re-configure VL_{on_mig}. The migration policies represent different search strategies to select the physical resources where virtual elements would be re-allocated. Each policy can have specific policy parameter values that drive the search objectives of the migration process.

We propose 4 migration policies for our framework (MP1 to MP4) which are described hereafter:

- *MP1*. It meets the VLMP availability condition (Expressions 4.1 and 4.2) to guarantee that the substrate resources will meet the non-additive constraint requirements of VL_{on_mig}, namely cpu_{constr}, and bw_{constr}. Its optimality function finds resources with CPU utilization lower than cpu^{use}_{thresh} to minimize CPU overutilization with the cost function in Expression 4.9.
- *MP2*. It meets the VLMP availability condition as MP1. It uses the optimality cost function in Expression 4.10 to find resources with CPU utilization lower than cpu^{use}_{thresh} and link capacity utilization lower than bw^{use}_{thresh}, in order to minimize CPU and link capacity overutilization.
- *MP3*. It meets the VLMP availability condition as MP1. It also uses the feasibility cost function in Expression 4.5 to meet the operator's delay constraint dly_{constr} of VL_{on_mig}. The optimality cost function in Expression 4.10 is used to find resources with both, CPU utilization lower than cpu^{use}_{thresh} and link capacity utilization lower than bw^{use}_{thresh}, in order to minimize CPU and link capacity overutilization.
- *MP4*. It meets the VLMP availability condition as MP1. It also implements the feasibility cost function in Expression 4.6 to meet both delay and hop-count operator's constraints (dly_{constr} and hop_{constr}) of VL_{on_mig}. The optimality cost function in Expression 4.10 is used to find resources with both, CPU utilization lower than cpu^{use}_{thresh} and link capacity utilization lower than bw^{use}_{thresh}, in order to minimize CPU and link capacity overutilization.

A summary of the four migration policies of our framework is given in Table 3.

5 Assessment Methodology and Setup

As we have described earlier, VN migration is a difficult problem that involves several aspects that include features of the substrate and virtual networks, features of the VNRs, VNE algorithms' performance, and migration-oriented parameters defined by the operator. This section describes the methodology we followed to assess the performance of the proposed VN migration framework and to obtain the

Table 3 A summary of the four migration policies of our framework

Policy	Availability	Feasibility	Optimality
MP1	Guarantees fulfillment of non-additive constraints (forwarding CPU and VL's BW) of VL_{on_mig}: cpu_{constr} and bw_{constr}	None	Minimize CPU overutilization optimality parameter cpu^{use}_{thresh}
MP2		None	Minimize CPU and BW overutilization—optimality parameters cpu^{use}_{thresh} and bw^{use}_{thresh}
MP3		Guarantees fulfillment of VL_{on_mig} delay dly_{constr}	Minimize CPU and BW overutilization—optimality parameters cpu^{use}_{thresh} and bw^{use}_{thresh}
MP4		Guarantees fulfillment of VL_{on_mig} delay and hop-count dly_{constr} and hop_{constr}	Minimize CPU and BW overutilization—optimality parameters cpu^{use}_{thresh} and bw^{use}_{thresh}

Migration policies and their parameters considered in the VN migration framework. Four migration policies (MP1 to MP4) are defined with different search criteria in order to drive the performance of VN migration considering different strategies to select the physical resources where virtual elements would be re-allocated

most suitable configuration parameters and settings that produce the most profitable results in terms of the enhancement of resource (re)allocation efficiency.

5.1 Performance Measurement

In our framework the ultimate goal of online VN migration is to enhance the quality of resource allocations to improve the chances of successful VNR embeddings. One of the most commonly used metric for analyzing the performance of online VNE is the *VNE acceptance ratio* [48] [53], which indicates the number of VNs effectively embedded out of the total number of VNRs:

$$VNE\ acceptance\ ratio = \frac{Number\ of\ successfully\ embedded\ VNRs\ by\ the\ VNE\ process}{Total\ number\ of\ VNRs} \tag{5.1}$$

In order to quantify the effectiveness of the migration process of our framework in our assessment, we introduce a metric, namely the relative *VNE acceptance ratio enhancement.* It quantifies the percentage increase of the VNE acceptance ratio while using the migration approach w.r.t. the VNE acceptance ratio without the migration approach (i.e. the default online VNE alone):

$$diff = VNE\ acceptance\ ratio_{With\ migrations} - VNE\ acceptance\ ratio_{Without\ migrations} \tag{5.2}$$

$$VNE\ acceptance\ ratio\ enhancement = \frac{diff \times 100}{VNE\ acceptance\ ratio_{Without\ migrations}} \tag{5.3}$$

We define the relative VNE acceptance ratio enhancement metric taking VNE acceptance ratio as a comparison basis to measure the capacity of the VN migration framework to impact the number of VNs effectively embedded out of the total amount of VNRs. We also define a metric to analyze the performance of the migration process itself. We define the *VN migration success ratio*, an adaption of the success ratio [52], as the fraction of successfully migrated virtual links by the total number of virtual link requests to be migrated:

$$Migration\ success\ ratio = \frac{Number\ of\ successfully\ migrated\ VLs}{Total\ number\ of\ VL\ migration\ requests} \tag{5.4}$$

As it will be further shown in Sect. 6.3, we observed a direct relationship between the migration success ratio and the VNE acceptance ratio.

Comparing the performance of the proposed VN migration framework with previous work is difficult since they either do not start with the same problem formulation or do not address the problem of enhancing the VNE acceptance ratio that we address. For example, Xiao et al. [12] mainly deal with substrate node failures; Tarutani et al. [23] focus on energy-aware reconfigurations; Tran et al. [22] and Wang et al. [9] use VNE migrations to tackle service disruption; Zhu and Ammar [7] and Zhao et al. [25] seek to minimize reconfiguration costs, being the remapping goal in [25] to minimize the chances for covert channel attacks. Yu

et al. [6] formulates a reconfiguration splitting solution that re-runs the embedding algorithm by allowing path splitting, whereas our approach only applies to unsplittable flow VNE environments. The works of Melo et al. [11, 14] propose a re-optimization VNE algorithm with multiple constraints to enhance the acceptance ratios. However, their approach is noticeably different from ours since it is posed as solving a multi-commodity flow constraint problem where solutions are found by means of linear programming.

Considering that our framework embodies many of the key ideas from prior work, we instead present a methodological approach that allowed us to comprehensively assess the performance of the framework while obtaining the most suitable configuration parameters that produced the most profitable results in terms of resource (re)allocation efficiency.

5.2 Assessment Methodology

The methodological assessment approach is depicted in Fig. 5. It is composed of three main steps:

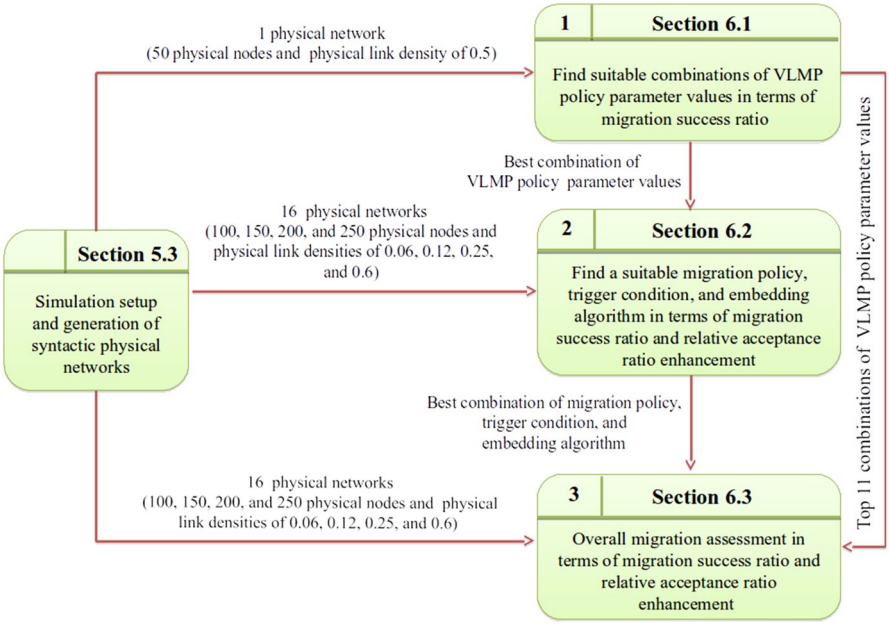

Fig. 5 Methodology for the assessment of the online VN migration framework in different VN environment conditions. In Step 1, we obtain suitable combinations of VLMP policy parameters. In Step 2, we further elaborate on suitable combinations of migration policy, trigger conditions, and embedding algorithms in terms of both migration success ratio and relative acceptance ratio enhancement. Finally, in Step 3, we conduct an assessment of the migration success ratio and relative acceptance ratio enhancement

- *Assessment Step 1*. Find combinations of parameters for the delay and hop-count constraints as well as CPU and BW overutilization thresholds that render the most prominent results in terms of migration success ratio. Choosing suitable combinations of policy values is not trivial. On the one hand, random combinations may result in defective resource (re)allocations and sensible reduction of the VNE acceptance ratio. On the other hand, the impractical number of executions required to get statistically valid results deserves a strategic approach to make the problem more tractable. This step presents such an approach in order to find the combinations of policy values that are most suitable for the VNE environment under evaluation. Section 6.1 elaborates on this step.
- *Assessment Step 2*. Find the most suitable policy, trigger condition and VNE algorithm that produces the most profitable results for the VNE environment under evaluation. For this purpose, we i) evaluate its effectiveness with the policy values obtained in Step 1; and ii) determine the most effective policy, trigger condition, and VNE algorithm. Section 6.2 elaborates on this step.
- *Assessment Step 3*. Evaluate the effects of the migration process in real-world scenarios where virtual networks scale up to large-scale network infrastructures. This step evaluates and analyses the performance of the migration process with the parameters and configuration settings obtained in Steps 1 and 2, respectively, in medium- to large-scale scenarios (100, 150, 200 and 250 substrate nodes) with several physical link densities (0.06, 0.12, 0.25 and 0.6). This step represents the overall assessment of online virtual network migration, for which formal analysis is carried out to determine the effects of the migration process in terms of efficiency of resource (re)allocations. Section 6.3 elaborates on this step.

5.3 Simulation Setup

Following the standard procedure in previous works [6, 7, 14, 37], we developed our own discrete-event software simulation platform that implements the features of the VN migration framework described in Sects. 3 and 4, in order to validate our approach. In our simulations VNRs arrive and leave at different discrete times with different topologies and lifetimes. The platform simulates VNR traffic based on a priority queue in which the priority is the arrival time of each VNR.

Our simulation platform includes an instance generator that produces substrate networks and VNRs. The network topologies (substrate and virtual) are produced with the GT-ITM graph generator. Attributes of the GT-ITM graphs were added to meet the requirements of online VNE environments like CPU processing capacity of the substrate and virtual nodes, BW and delay of the substrate and virtual links, and time of life of VNRs. The values of CPU and BW capacities as well as link delays have been generated using a minimum–maximum range following a uniform distribution.

The time of life of VNRs followed an exponential distribution. VNRs arrive following a Poisson distribution that determines the order and arrival time for each VNR. A summary of online VNE parameters of our simulation setup is shown in Table 4, which has been elaborated taking into account setups from the literature

Table 4 A summary of online VNE parameters of our simulation setup which has been elaborated taking into account setups from the literature for the same purposes [6, 45, 46, 54]

Substrate network								Virtual network requests										
# Nodes	Node connection probability	CPU (UR)		Delay (Msec)		BW (UR)		# VNRs	# Nodes		Node connection probability	CPU (UR)		BW (UR)		Life time (UT)	Arrival rate	Simulation time (UT)
		Min	Max	Min	Max	Min	Max		Min	Max		Min	Max	Min	Max			
100	0.06	50	100	0	150	50	100	1000	2	10	0.5	1	20	1	50	500	4/100	2500
100	0.12	50	100	0	150	50	100	1000	2	10	0.5	1	20	1	50	500	4/100	2500
100	0.25	50	100	0	150	50	100	1000	2	10	0.5	1	20	1	50	500	4/100	2500
100	0.6	50	100	0	150	50	100	1000	2	10	0.5	1	20	1	50	500	4/100	2500
150	0.06	50	100	0	150	50	100	1000	2	10	0.5	1	20	1	50	500	4/100	2500
150	0.12	50	100	0	150	50	100	1000	2	10	0.5	1	20	1	50	500	4/100	2500
150	0.25	50	100	0	150	50	100	1000	2	10	0.5	1	20	1	50	500	4/100	2500
150	0.6	50	100	0	150	50	100	1000	2	10	0.5	1	20	1	50	500	4/100	2500
200	0.06	50	100	0	150	50	100	1000	2	10	0.5	1	20	1	50	500	4/100	2500
200	0.12	50	100	0	150	50	100	1000	2	10	0.5	1	20	1	50	500	4/100	2500
200	0.25	50	100	0	150	50	100	1000	2	10	0.5	1	20	1	50	500	4/100	2500
200	0.6	50	100	0	150	50	100	1000	2	10	0.5	1	20	1	50	500	4/100	2500
250	0.06	50	100	0	150	50	100	1000	2	10	0.5	1	20	1	50	500	4/100	2500
250	0.12	50	100	0	150	50	100	1000	2	10	0.5	1	20	1	50	500	4/100	2500
250	0.25	50	100	0	150	50	100	1000	2	10	0.5	1	20	1	50	500	4/100	2500
250	0.6	50	100	0	150	50	100	1000	2	10	0.5	1	20	1	50	500	4/100	2500

Our assessment setup goes beyond the state of the art in terms of substrate network size and densities as it includes substrate network sizes of 50, 100, 150, 200 and 250 and densities of 0.06, 0.12, 0.25 and 0.6, rendering to a VNE simulation setup that includes the sixteen test instances

for the same purposes [6, 45, 46, 54]. Our assessment setup goes beyond the state of the art in terms of substrate network size and densities as it includes substrate network sizes of 50, 100, 150, 200 nodes and 250 and link densities of 0.06, 0.12, 0.25, 0.5 and 0.6, obtaining a VNE simulation setup that includes a total of seventeen test instances, as shown in Table 4. We consider 50 substrate network nodes as small-scale, 100 and 150 substrate network nodes as medium-scale, and 200 nodes or higher as large-scale.

In our assessment we considered that ISNs should guarantee CPU availability to meet the forwarding rate requirements of the VL subject of migration. For the purposes of this study, we considered a constant CPU utilization of 10% due to forwarding to simulate a traffic rate of near 2 Mbps in both the VNE process and VN migration process. This is the value of the cpu_{constr} constraint. Nonetheless, in the VNE process simulations we allowed for controlled CPU overutilization greater than 100% in physical nodes that simultaneously act as both a source/target in a mapped path and as ISN in order to: (i) allow for the building up of ISN bottlenecks due to forwarding and observe the performance of the VN migration process while solving them, as well as to (ii) study the overall impact of this class of bottlenecks on the framework.

Regarding the migration process, our setup used configurable values for the parameters of the migration policies MP1 to MP4 (see Table 3). For the feasibility cost function parameters, the values for the constraint bw_{constr} are obtained in runtime from the requirements of the virtual link under migration evaluation (VL_{on_mig}). Note that all of the migration policies guarantee that the substrate resources for $p_{vlmp} = (s, \ldots, t) \in V_{phys}$ will have enough BW and enough CPU to forward the required rates of VL_{on_mig}. For the feasibility function parameters our simulation setup considers the following values: (i) four values of the delay constraint (dly_{constr}): 50 ms, 100 ms, 250 ms, and 300 ms; (ii) four values of hop-count constraint (hop_{constr}): 3 hops, 4 hops, 5 hops and 6 hops. Finally, for the optimality cost function parameters our simulation setup considered the following values: (i) for the CPU overutilization threshold (cpu^{use}_{thresh}): 50%, 60%, 70% and 80%; (ii) for the BW overutilization threshold (bw^{use}_{thresh}): 50%, 60%, 70% and 80%.

Considering the possible values for dly_{constr}, hop_{constr}, cpu^{use}_{thresh}, and bw^{use}_{thresh} mentioned earlier, we have a total of $4^4 = 256$ combinations of possible policy values. Our assessment considers four policies (MP1 to MP4), four migration trigger conditions (TC1 to TC4), and four VNE algorithms (GA, PSO, ACO, HS) subject of analysis. Thus, the combination of migration policies, trigger conditions, and VNE algorithms is $4^3 = 64$, and the total number of parameter configurations for our holistic assessment is $256 \times 64 = 16,384$.

Taking into account that we experiment with sixteen test instances in Assessment Steps 2 and 3, an exhaustive assessment would consist of $16,384 \times 16 = 262,144$ evaluation tests, which in turn should be executed a statistically valid number of times to produce statistically valid results. The execution of a statistically valid number of simulations for the 262,144 evaluation tests are impractical even with high-performance computing equipment (the reader should have in mind that each typical simulation mocks an online VNE process with 1000 VNRs, as summarized in Table 4). To make the problem more tractable and in order to reduce the number of

experiments to a subset of a full factorial design while preserving the most important variable interactions, we propose the use of Covering Arrays (CAs) [55] to experiment only with the most important variable configurations.

A CA is a mathematical object used for software testing purposes where testing all possible configurations of a software program can be redundant and it is needed to achieve an adequate simulation of all of the program's behavior with a much smaller number of carefully chosen configurations. The latter number is given by the size of the CA. Specifically, we make use of the conventional $CA(t, q, r)$, which is parameterized by t, q, and r, where t is strength of the CA and measures how "fully" we cover the possible variable interactions in the array, q is the number of possible values that each entry in the array can take, and r is the number of columns in the array.

In our methodological approach, the CA columns represent the different variables of the experiments (for example VNE algorithms, migration policy, trigger condition, delay constraint, etc.) and the CA rows represent the actual value combinations for these variables in the simulation trials.

6 Assessment Results and Analysis

6.1 Step 1: Suitable Combinations of Policy Parameter Values

In this step we determined the combinations of policy parameter values suitable for the VNE environment under evaluation ratio by carrying out the following procedure: (i) we determined a tractable number representative combinations of policy parameter values to evaluate the performance of the migration approach; (ii) we evaluated the performance of the migration approach in terms of its migration success ratio on a physical network of size 50; (iii) we applied statistical significance tests to determine suitable combinations of policy parameter values that provide the best performance of the VN migration process.

To determine a tractable set of representative combinations of policy parameter values, in this assessment step we used a $CA(3, 7, 4)$ in order to cover $r = 7$ variables (VNE algorithms, migration policies, trigger conditions, delay constraints, hop-count constraints, CPU overutilization thresholds, and BW overutilization thresholds) and $q = 4$ different values for each variable. A value of $t = 3$ guaranteed the evaluation of the migration performance at least with all the combinations of VNE metaheuristics, migration policies and trigger conditions. Following on, our test configuration trials were defined via the $CA(3, 7, 4)$ obtained from the National Institute of Standards and Technology (NIST) Covering Array Tables [56], which allowed us to reduce the number of experiments from $16,384$ to only 124 (less than 5% of the full factorial design).

Next, we evaluated the performance of our migration approach by executing a statistically valid number of simulations for each of the CA-defined 124 configuration combinations. Following the rule of thumb of the Central Limit Theorem [57], we executed thirty-one online simulations for each of the 124 configuration

combinations. In this step we used a 50-node substrate network with the VNE environment shown in the first row of Table 4.

Finally, we applied statistical significance tests to identify the combinations with the best migration performance in terms of the migration success ratio. For this purpose, we applied Wilcoxon's test [58] to our data to detect the combinations whose migration success ratios do not statistically differ from the combination with the best migration success ratio. The eleven combinations in Table 5 are the best in terms of the migration success ratio considering the 124 CA-based combinations representative of the VNE environment, covering all 3-interactions of VNE metaheuristics (GA, PSO, ACO, HS), all migration policies and all trigger conditions. The migration success ratio on the rightmost column in Table 5 is averaged from the 31 execution runs of the 124 combinations. For each constraint and threshold, we selected from Table 5 the values with the highest frequency in order to determine the most dominant values among the QoS parameters. We found the values $dly_{constr} = 300$ ms, $hop_{constr} = 3$ hops, $cpu^{use}_{thresh} = 60\%$, and $bw^{use}_{thresh} = 50\%$ to be the most suitable to configure the VN migration process in the next step of the assessment.

6.2 Step 2: Suitable Policy, Trigger Condition, and Embedding Algorithm

In this step we determined the policy, trigger condition and VNE algorithm that better suits the environments under evaluation. For this purpose, we carried out the following procedure: (i) we evaluated the performance of the migration framework configured with the most suitable policy parameter values obtained in Assessment Step 1 on an array of 16 physical networks; and (ii) we determined the most effective policy, trigger condition, and VNE algorithm in terms of the VNE acceptance ratio enhancement.

Table 5 The most suitable combinations in terms of the migration success ratio considering the 124 CA-based combinations representative of the VNE environment, covering all 3-interactions of VNE metaheuristics (GA, PSO, ACO, HS), all migration policies and all trigger conditions

Performance rank	$dly_{constr}(ms)$	$hop_{constr}(hops)$	$cpu^{use}_{thresh}(\%)$	$bw^{use}_{thresh}(\%)$	Migration success ratio (%)
Combination 1	50	3	70	80	49.99
Combination 2	50	4	60	50	45.35
Combination 3	50	5	60	60	35.95
Combination 4	50	6	60	50	40.08
Combination 5	250	3	50	60	47.23
Combination 6	300	3	50	70	38.29
Combination 7	300	3	60	50	39.57
Combination 8	300	3	70	50	46.99
Combination 9	300	4	70	80	41.27
Combination 10	300	4	80	50	41.22
Combination 11	300	6	50	50	41.32

First, in order to evaluate the migration effectiveness, we use the dominant policy values from the previous assessment step. Second, our intention is to evaluate the effectiveness of the approach in medium- to large-scale scenarios (100, 150, 200 and 250 substrate nodes), with several densities (0.06, 0.12, 0.25 and 0.6), with the online VNE simulation parameters shown in Table 4.

In this step, we chose the CA(3,5,4) [56] to consider $r = 5$ variables (metaheuristic, policy, trigger, physical network size, and density), and $q = 4$ different values for each variable. A value of $t = 3$ means that we experimented with all the value combinations of any three variables (including every combination of metaheuristic, policy and trigger condition).

Our intention was to get a non-exhaustive but representative evaluation of the behavior of our migration approach with emphasis on these three variables. The CA prescribed experiments with 94 combinations of our five variables. For each of the 94 combinations we executed 31 simulations to obtain statistical valid results. We also executed the same simulations without the migration process to compare the performance of the migration framework.

From the evaluations, we found that migration policy MP4, trigger condition TC4 and the Genetic Algorithm (GA) were the most suitable configuration settings in terms of relative VNE acceptance ratio enhancement. Particularly, we observed that TC4 produced more than 15% relative online VNE acceptance ratio enhancement on the network with 250 substrate nodes networked with density 0.6. The full set of observed results when evaluating the migration process with different trigger conditions can be found in our previous work [40].

6.3 Step 3: Overall Migration Assessment

For each of the eleven combinations of policy parameter values identified in the Assessment Step 1, we carried out one experiment. Each experiment consisted of thirty-one simulation runs of the migration approach by using MP4, TC4 and the GA algorithm, as determined in the Assessment Step 2. In each experiment the thirty-one simulations were run for each row of the setup shown in Table 4 for medium- to large-scale scenarios (100, 150, 200 and 250 substrate nodes) with several densities (0.06, 0.12, 0.25 and 0.6), namely with sixteen test instances.

In order to compare the migration approach with the traditional VNE approach (without migration), we also executed the above procedure without the migration approach (only online VNE) to compute the relative VNE acceptance ratio enhancement metric. Figure 6 shows the average values of the thirty-one simulations for the relative VNE acceptance ratio enhancement metric computed for all combinations, for all network densities and sizes of the VNE environment under evaluation.

From the overall assessment we conclude that the relative VNE acceptance ratio enhancement that our VN migration framework scales well with the network density and size. Specifically, for substrate networks of very low density (0.06), the enhancement ranges from 2% for network sizes of 100 nodes up to 4.5% of enhancement for network sizes of 250 nodes. For mid-sized networks, the largest enhancements range

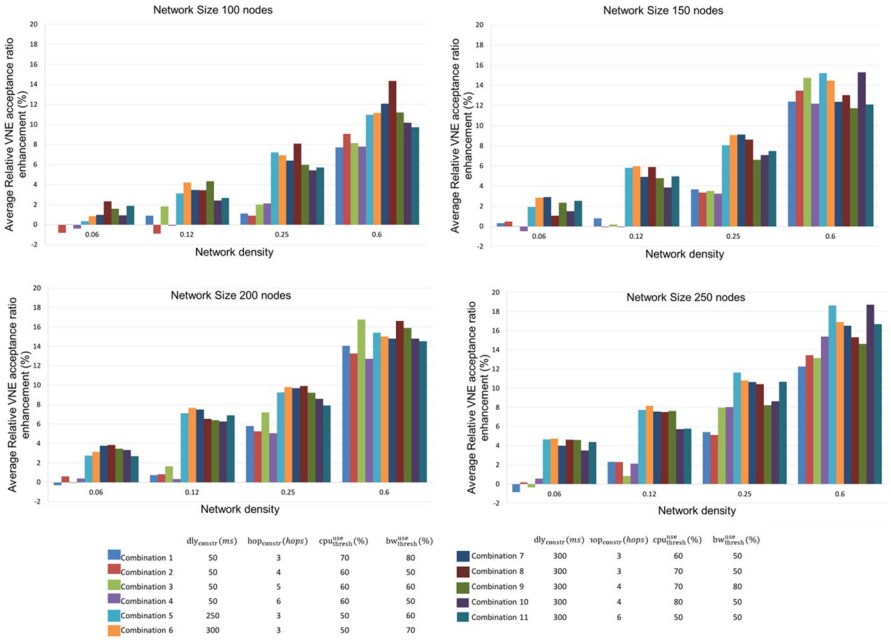

Fig. 6 Average on-line VNE acceptance ratio enhancement for medium- to large-scale scenarios (100, 150, 200 and 250 substrate nodes) with several densities (0.06, 0.12, 0.25 and 0.6), i.e. with sixteen test substrate network instances. We show the average values of thirty-one simulations for the relative VNE acceptance ratio enhancement metric computed for all 11 VLMP parameter combinations (Color figure online)

from 6% to 10%. For substrate networks with density of 0.6, the enhancements go from 14.5% for 100-node networks up to 18.7% for 250-node networks.

This indicates that the advantages of online VN migration are more accentuated in large-sized physical networks. This aspect is of particular relevance considering that a recent analysis demonstrated that the performance of VNE solutions in key metrics like VN acceptance ratio decreases drastically [8] with large sizes of the substrate and virtual networks. The lowest impact of online VN migration was observed on the smallest and sparsest physical networks. However, even in these cases, the benchmark provided in Fig. 6 allows to identify the most suitable combination of policy parameter values in such specific network conditions. Online virtual network migration has been proved to be an effective management tool to increase the efficiency of resource allocations. Figure 7 suggests a correlation between the migration success ratio and the VNE acceptance ratio enhancement for the online migration assessment presented in this paper. Figure 7 provides the average values of the eleven combinations for both, migration success ratio (right part of Fig. 7) and relative VNE acceptance ratio enhancement (left part of Fig. 7). The highest averaged migration success ratio was above 80%, which resulted in an average relative VNE acceptance ratio enhancement above 15% for the largest and highly dense networks considered in our assessment.

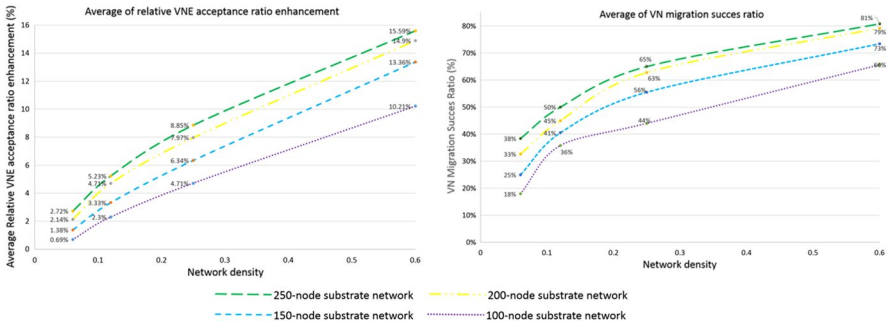

Fig. 7 Potential correlation between the migration success ratio (left) and the VNE acceptance ratio enhancement (right) (vertical axes) for all combinations in each substrate network size as the density of the SN scales (horizontal axis). Each curve represents a different SN size

Moreover, for the same type of networks, the lowest averaged migration success ratio was about 68%, which resulted in an average relative VNE acceptance ratio enhancement above 10%. Finally, the lowest migration success ratios were observed on the smallest and sparsest networks. In these cases, the highest averaged migration success ratio was about 38%, which resulted in a modest average relative VNE acceptance ratio enhancement above 2.7%. This might be a consequence of a reduced number of paths of resources available while evaluating the suitability of resources for migration on these networks. In summary, the results suggest that there is a direct effect of the migration success w.r.t the VNE acceptance ratio enhancements in our framework, in that the higher the migration success ratio, the higher the VNE acceptance ratio enhancement.

6.4 Statistical and Comparative Analysis

The purpose of this section is two-fold. First, we present a formal statistical significance analysis of the results obtained in our assessment to determine the policy parameter values that have higher impact on the performance of the migration process. Later, we provide a comparative study of the migration framework performance against state-of-the-art mechanisms that improve VNE performance while looking for VNE solutions.

6.4.1 Statistical Analysis

In order to carry out an appropriate significance test of the observed results, we initially applied the Shapiro–Wilk normality test [59] to determine, for each combination of policy parameter values in Table 5, if the results of the online VNE acceptance ratio enhancement for all physical network sizes and densities are normally distributed or not. This is necessary to determine the type of significance test required to analyze the results.

The normality test of our simulations determined that results for Combinations 1 to 4 come from normal distributions whereas for Combinations 5 to 11 the results

Table 6 Shapiro–Wilk test results: each row consists of the most suitable combination of policy parameter values, the calculated p value, and average relative VNE acceptance ratio enhancement for each physical network size and density

Combinations#	dly_{constr} (ms)	hop_{constr} (hops)	cpu^{use}_{thresh} (%)	bw^{use}_{thresh} (%)	P-value	AVG-VNE enhancement (%)	Shapiro–Wilk test results
Combination 1	50	3	70	80	0.0082	4.1463	×
Combination 2	50	4	60	50	0.0032	4.1498	×
Combination 3	50	5	60	60	0.0057	4.8465	×
Combination 4	50	6	60	50	0.0069	4.3070	×
Combination 5	250	3	50	60	0.6910	8.1090	✓
Combination 6	300	3	50	70	0.8541	8.2329	✓
Combination 7	300	3	60	50	0.7438	7.9142	✓
Combination 8	300	3	70	50	0.6710	8.2179	✓
Combination 9	300	4	70	80	0.4250	7.4112	✓
Combination 10	300	4	80	50	0.1185	7.2618	✓
Combination 11	300	6	50	50	0.2403	7.2855	✓

In addition, if the data are normally distributed the symbol $\surd$ is used and if the data are not normally distributed the symbol × is used. A significance level of 0.05 was used in all statistical tests as suggested in the literature

come from asymmetric distributions. For the sake of completeness, the results of the Shapiro–Wilk test are shown in Table 6. In summary, the normality test showed that some results come from asymmetric distributions, and hence the significance tests to analyze the overall performance of the eleven combinations should be non-parametric.

To analyze statistically the patterns observed in the eleven combinations of policy parameter values, and to identify the parameters that provoke such behavior, we initially performed the Spearman correlation test, which is a nonparametric measure of the monotonicity of the relationship between datasets [60]. Like other correlation coefficients, this one varies between -1 and +1 with 0 implying no correlation. Correlations of -1 or $+1$ imply an exact monotonic relationship. The results of the Spearman correlation test are shown in Fig. 8. The test reveals that Combinations 5 to 11 (highlighted in yellow) are highly correlated (correlation values greater than 0.96), while the remaining combinations are divided into two subgroups.

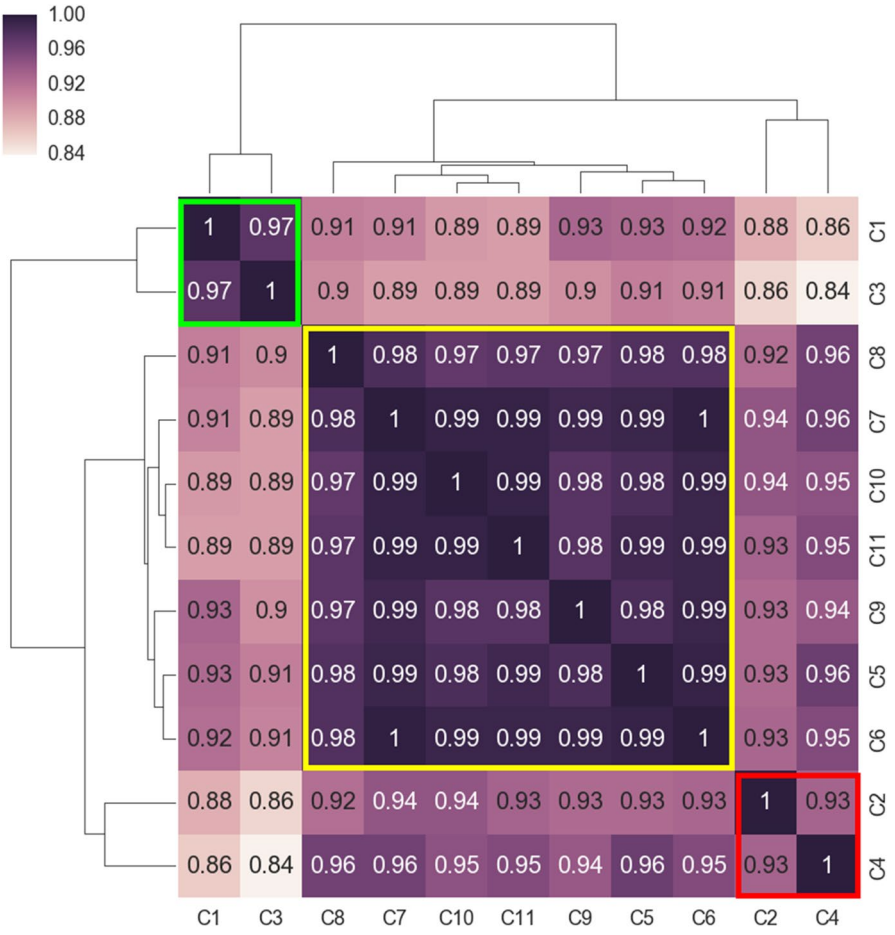

Fig. 8 Spearman correlation tests. The value of each cell represents the correlation value between the combinations of the i-th row and the j-th column, where cells with a darker tone have a higher correlation value. The values in the three highlighted blocks correspond to clusters of highly-correlated combinations

The first group consists of Combinations 1 and 3 (highlighted in green), and the other group consists of Combinations 2 and 4 (highlighted in red). Based on the results of the Spearman test and the parameters of each combination, we conclude that the main difference between the group of combinations highlighted in yellow and the remaining combinations is the delay constraint value. Namely, Combinations 5 to 11 have a higher dly_{constr} value than the rest combinations.

This suggests that the delay constraint is a critical parameter that affects the efficiency of migration-aware resource allocations. However, the remaining of this section analyses the effect of other policy parameter values.

In order to determine the order of influence of policy value parameters in the efficiency of online VN migration, we applied the Friedman test [61], which is a non-parametric statistical test that detects differences in treatments across multiple test attempts and ranks the overall performance of each test.

The objective is to find out the groups of combinations for which the correlation between substrate network size, density and acceptance ratio enhancement is statistically significant as well as to determine the critical parameter values in each combination group.

For this purpose, we performed a series of Friedman tests that have produced the results are shown in Fig. 9:

- *Friedman Test 1*. The result of Friedman test among the eleven combinations proves that the VNE acceptance ratio enhancement pattern is statistically significant for all substrate network sizes and densities when Combinations 1 to 11 are evaluated altogether. Note that Combinations 1 to 11 does not share the same delay.
- *Friedman Test 2*. The VNE acceptance ratio enhancement pattern is not statistically significant for all substrate network sizes and densities among Combinations 1 to 4. Note that Combinations 1 to 4 share the same delay (very-low).
- *Friedman Test 3*. The VNE acceptance ratio enhancement pattern is statistically significant for all substrate network sizes and densities when Combinations 5 to 11 are evaluated altogether. Note that Combinations 5 to 11 do not share the same hop count.
- *Friedman Test 4*. The VNE acceptance ratio enhancement pattern is not statistically significant for all substrate network sizes and densities among Combinations 5 to 8. Note that Combinations 5 to 8 share a very low hop count constraint.
- *Friedman Test 5*. The VNE acceptance ratio enhancement pattern is not statistically significant for all substrate network sizes and densities among Combina-

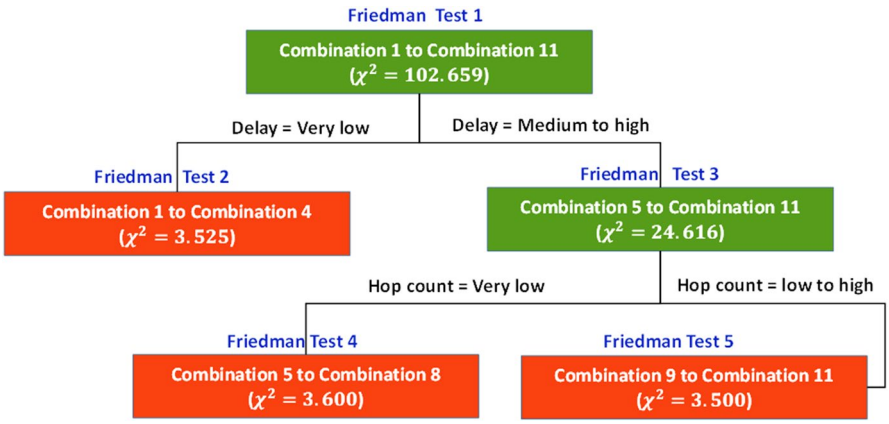

Fig. 9 Results of the Friedman tests on the eleven combinations of policy parameter values. Green boxes denote groups of combinations that did pass the Friedman test. Red boxes denote groups of combinations that failed the Friedman test (Color figure online)

tions 9 to 11. Note that the hop count constraint for Combinations 9 to 11 is low to high here.

From the statistical analysis results, we have drawn the following conclusions. First, the delay constraint (dly_{constr}) is the most important parameter that drives the efficiency of resource (re)allocation.

Very low delay (e.g. 50 ms) renders to very poor migration success ratios and consequently very low (if any) enhancements of VNE acceptance ratios. More relaxed delay constraint values (e.g. 250–300 ms) produce more efficient resource (re)allocations. After delay, the second most influential constraint is the hop count constraint (hop_{constr}), where again, more relaxed values allow increasing the online VNE acceptance ratio via online VN migration.

6.4.2 Comparative Analysis

The ultimate goal of our migration-based approach is to enhance the performance of online VNE. As such, a holistic migration approach like the one described in this paper is missing in the literature to compare its performance. However, as discussed in Sect. 2.1, the research community has developed mechanisms targeting this high-level objective without reallocating resources. In this section, we compare the performance of our migration-based approach with two state-of-the-art mechanisms aimed at enhancing the performance of online VNE. The two mechanisms used for comparison in this section are meant to modify the performance of online VNE while looking for online VNE solutions.

The first mechanism used for comparison is the initialization function L2S2 (large to large and small to small), which was proposed in [46]. The main idea of using this function in online VNE is to increase the possibility to satisfy the virtual node's connectivity constraints in the linking mapping stage when a virtual node is embedded to a substrate node with more bandwidth resources.

This function has been demonstrated to be efficient in the vast majority of metaheuristics used for online VNE [20]. The second mechanism is based on a penalty function proposed in [62]. The core idea of using this penalty function in online VNE is to grade the quality of VNE candidate solutions according to the degree of fulfillment of their constraints as to exploit this information to drive the metaheuristics to more promising regions of the search process, enhancing their performance.

The penalty function-based online VNE approach is developed in a previous work [20], where the reader can find comprehensive descriptions, pseudocodes and reviews of the online VNE process enhanced with the initialization function L2S2 as well as with the penalty function.

In order to compare our migration-based online VNE approach we produced two VNE algorithms based on the genetic metaheuristic algorithm, one including the L2S2 function (identified as L2S2) and one including the penalty function (identified as PF). For comparative purposes, we have chosen the instance number 2 of Table 4. This instance includes a substrate network with 100 nodes with a density of 0.06 and a set of 1000 virtual network requests.

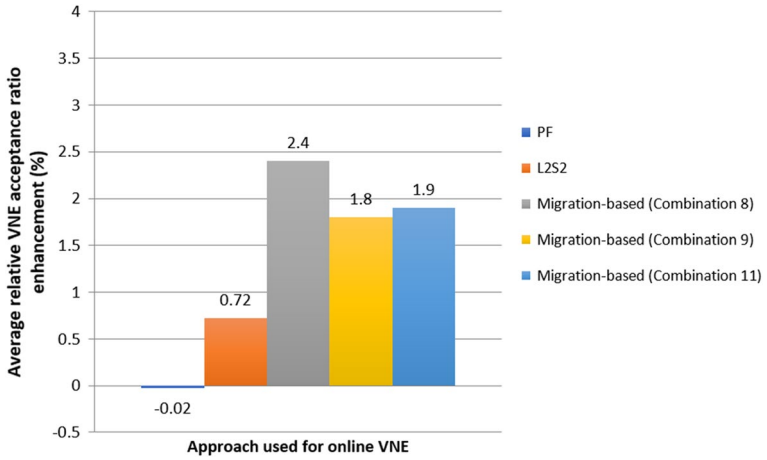

Fig. 10 Relative enhancements for PF- and L2S2-based VNE, and migration-based VNE (Color figure online)

Figure 10 shows the average values of thirty-one simulations for the relative VNE acceptance ratio enhancement metric computed for each approach. From the figure, we can see that the relative VNE acceptance ratio enhancements of the Combinations 8, 9 and 11 of our migration-based approach (relative enhancement of 2.4%, 1.8% and 1.9%respectively) are higher than the ones obtained by the L2S2 mechanism (enhancement of 0.72% relative). On the other hand, we can observe that the PF-based approach produces a relative decrease in the VNE performance of around 0.02% for the instance under evaluation.

It is important to mention that we chose the substrate network with 100 nodes with a density of 0.06 for testing purposes because with such an instance our migration-based approach presented the lowest enhancements of all our test instances. Even in such conditions, our migration- based approach outperformed the two mechanisms used for comparison (PF and L2S2).

Since the enhancements obtained by our approach scale well with the network density and size, the difference in the performance between our approach and the mechanisms would be higher in larger networks with higher density. That is, our migration-based approach would outperform the two state of the art approaches in a much higher degree because all metaheuristic-based approaches decrease drastically their performance with large scale networks [20], contrary to our migration-based approach, which produces much better results in large scale networks (average relative enhancements around 18.7% for 250-node networks with 0.6 density as graphically shown in Fig. 6.

7 Discussion

7.1 Main Contributions

This paper has presented the first attempt to propose an integrated framework to address online virtual network migration holistically, assessed under different virtual network environment conditions addressing three critical questions: (i) "When to trigger resource reallocations?"; (ii) "What to reallocate?"; and (iii) "From which physical resources to which physical resources should be reallocated?". The innovative aspects while addressing these three critical aspects are summarized as follows:

1. We have evaluated state of the art and novel conditions that must hold to trigger VN migration. For the holistic environment conditions evaluated in this paper, a trigger condition driven by a threshold of the VNE acceptance ratio (TC4) produced the most suitable results in terms of relative VNE acceptance ratio enhancement. This trigger condition is more efficient because it drives migrations as a function of the overall VN embedding performance, which in turn can be affected by the lack of physical resources or by the poor performance of the embedding algorithm in use. In this regard, our assessment demonstrates that VNE acceptance ratio thresholds can be exploited to trigger virtual resources reconfigurations in favor of more efficient allocations.
2. A virtual link-oriented migration approach to select the resources with potential to be re-allocated has been analyzed. In this regard, our assessment has demonstrated that substrate resource overutilization is a criterion that can be exploited to select the resources that could be considered for efficient online resource re-allocations.
3. We have analyzed the performance of the VLMP approach under a wide range of network virtualization environment conditions and we have demonstrated its suitability to find appropriate resources where to re-configure virtual links in online VN migration scenarios. In this regard, our assessment describes an approach to define the most appropriate parameters that drive the VLMP search criteria. For the holistic environment conditions evaluated in this paper, the most suitable migration policy (MP4) guarantees that the substrate resources meet the non-additive constraint requirements (CPU and BW) of the virtual links subject to-be-migrated. Also, we demonstrated that the delay and hop-count operator's constraints on the use of the substrate resources have a direct effect on the efficiency of the resource (re)allocation. Namely, we empirically demonstrated that more relaxed constraints produce higher enhancements on the VNE acceptance ratio, whereas strict constraints render to lower enhancements of the VNE acceptance ratio. The VLMP approach has been demonstrated to be a tool that can be further exploited to find tradeoffs between QoS constraint fulfillment, substrate resource utilization directives (through the feasibility and optimality cost functions) and online VNE acceptance ratio enhancement.
4. We have analyzed the performance of online migration configurations in a wide scope of VNE environment conditions, going beyond the state of the art in terms

of substrate network size (up to 250 physical nodes) and substrate link densities (sparse to dense). We have provided an approach to make the assessment tractable due to a large number of possible combinations subject to analysis. Following on, we have demonstrated that it is possible to achieve enhancements that range from 6 to 10% of relative VNE acceptance ratio for mid-size networks, and from 14.5 to 18.8% for mid- to large-scale networks. This is an important achievement considering that previous work in literature has demonstrated that the performance of VNE solutions in key metrics like VN acceptance ratio decreases drastically with large sizes of the substrate and virtual networks.

5. We have compared the performance of the framework against two VNE algorithms based on the GA equipped with state-of-the-art mechanisms aimed at enhancing the performance of online VNE: one includes the L2S2 initialization function and the other includes a quality grading penalty function. We observed acceptance ratio enhancements up to 3x higher than the GA equipped with L2S2 when using our framework on a physical network instance with 100 nodes and a density of 0.06.

Our results also suggest that there is a proportional impact of the migration success ratio on VNE acceptance ratio enhancement in our migration framework. Moreover, due to the nature of the process, we have shown that the migration success ratio depends on non-additive and additive QoS parameters that should be carefully defined.

VN migration is costly in terms of migration time, service disruption, resource utilization and so on. One limitation of the work in this paper is the lack of consideration of the cost involved in the migration. An aspect that deserves special attention is the introduction of a method to detect the critical times to trigger VN migration to reduce migration costs while keeping good VNE acceptance ratios. In this regard, our work in [38] demonstrates that the VN migration framework presented in this paper provides enough flexibility to designers and InPs to be extended in different directions.

7.2 Deployment on SDN Environments

The presented VN migration framework can be deployed in Software Defined Networking (SDN) environments. For this purpose, consider the SDN concepts introduced in [63, 64], enhancing them with the VN migration approach described in this paper. An architectural view of the VN migration-oriented SDN approach is graphically shown in Fig. 11.

The proposed SDN VN migration architecture consists of a centralized control plane with a southbound API such as OpenFlow for communication with the physical infrastructure (data plane) and a northbound API for communication with the service layer (network applications). The service layer receives virtual network requests and it handles them to construct virtual networks to offer end-to-end services to the users. The SDN controller plays the role of a global resource control [63] and it includes the three main components of the VN migration framework: (i)

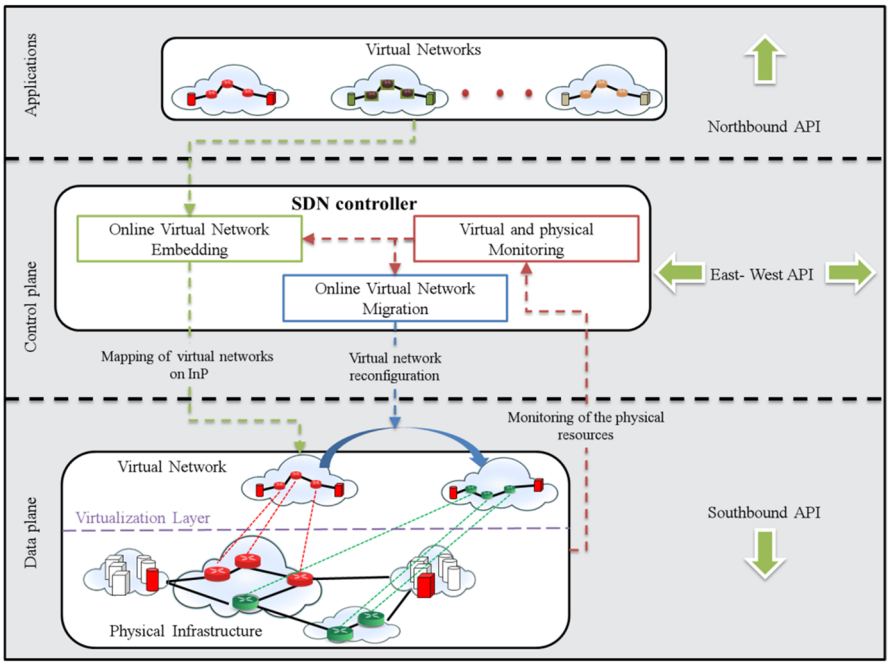

Fig. 11 An architectural view of an online VN migration-oriented SDN approach. The architecture considers a centralized control plane with a southbound API such as OpenFlow for communication with the physical infrastructure (data plane) and a northbound API for communication with the service layer (network applications)

Online VNE; (ii) Online VN migration; and (iii) Virtual and physical network monitoring. The SDN controller can use the VNE algorithms enumerated in Sect. 3.2 to map VNs onto the physical infrastructure.

As for online VN migration, the SDN controller can use the VLMP heuristic approach described in Sect. 4 in which the virtual path reconfigurations are driven by the VLMP process. The choice of optimal path is determined by VLMP considering the QoS requirements of VNRs. Finally, the monitoring component can provide information about the current state of physical and virtual resources. The co-existence of these three components is feasible in SDN environments where the control plane is centralized. When more than one SDN domain is necessary, the east–west API can extend the management functions to other controllers. The lifecycle of such a SDN VN migration platform can be summarized in the following steps:

1. The service provider constructs the virtual network topology according to demands of end users and sends it to our online VN embedding process.
2. Online VN embedding process located in the SDN controller will execute the VNE algorithm to embed VN onto the physical infrastructure.

3. If VNE succeeds, our online VN embedding process will notify physical infrastructure to allocate physical resources in terms of CPU and BW to the VN through the southbound interface.
4. Otherwise, an online VN embedding process will return a REJECT message.
5. When VN migration is required, the SDN controller triggers the VLMP algorithm described in Sect. 4, resulting in appropriate resource reallocations, keeping acceptable VNE acceptance rates.
6. When the lifetime of a VN is over, the resources are released.

8 Conclusions

In this work we have developed a holistic VN migration framework together with a methodological approach that allows obtaining the most suitable configuration parameters that produce the most profitable results in terms of resource (re)allocation efficiency in different online VNE environments. The proposed VN migration framework has been proven to enhance VN embedding acceptance ratios on medium- and high-scale physical networks with different sizes and densities, considering different migration policies and different migration triggers.

We have validated our methodology with a variety of additive and non-additive QoS constraints. Our assessment demonstrates that the VN migration process can serve as a key mechanism to enhance the performance of VNE processes. We observed VNE acceptance ratio enhancements of up to 18.7% on large scale physical networks. Finally, we compared the performance of the framework against GA VNE algorithms equipped with two state-of-the-art VNE enhancement mechanisms and observed acceptance ratio enhancements up to 3× higher when using our framework on a physical network instance with 100 nodes and a density of 0.06.

Future work will be directed to develop more advanced approaches where the configuration parameters would be automatically differentiated depending on QoS-aware requirements and administrative directives. Future work also involves its extension to splittable flow online VNE environments. We expect these two important aspects of future work to have a direct impact on the dynamics of the management tasks, which may possibly require the elaboration of self-adapting mechanisms that will contribute to enhancing, even more, the efficiency of resource allocation in NV environments.

The online VN migration problem is still at its initial stage. We hope that the ideas presented in this paper may encourage practitioners and researchers to address the online migration in other NV and software-defined networking environments.

Acknowledgements This work was supported by the National Council of Research and Technology (CONACYT) through Grant FONCICYT/272278.

References

1. Fischer, A., Botero, J.F., Beck, M.T.: Virtual network embedding: a survey. Surv. Tutor. **15**, 1888–1906 (2013)
2. Chowdhury, N.M.M.K., Rahman, M.R., Boutaba, R.: Virtual network embedding with coordinated node and link mapping. In: IEEE INFOCOM 2009—The 28th Conference on Computer Communications, pp. 783–791. IEEE, New York (2009)
3. Chowdhury, N.M.M.K., Boutaba, R.: Network virtualization: state of the art and research challenges. IEEE Commun. Mag. **47**, 20–26 (2009)
4. Alshaer, H.: An overview of network virtualization and cloud network as a service. Int. J. Netw. Manag. **25**, 1–30 (2015)
5. Hsu, W.-H., Shieh, Y.-P., Wang, C.-H., Yeh, S.-C.: Virtual network mapping through path splitting and migration. In: 2012 26th International Conference on Advanced Information Networking and Applications Workshops, pp. 1095–1100. IEEE, Nwe York (2012)
6. Yu, M., Yi, Y., Rexford, J., Chiang, M.: Rethinking virtual network embedding: substrate support for path splitting and migration. SIGCOMM Comput. Commun. Rev. **38**, 17 (2008)
7. Zhu, Y., Ammar, M.: Algorithms for assigning substrate network resources to virtual network components. In: Proceedings IEEE INFOCOM 2006. 25th IEEE International Conference on Computer Communications, pp. 1–12. IEEE, New York (2006)
8. Shahin, A.A.: Memetic multi-objective particle swarm optimization-based energy-aware virtual network embedding. Int. J. Adv. Comput. Appl. **6**, (2015)
9. Wang, Y., Keller, E., Biskeborn, B., van der Merwe, J., Rexford, J.: Virtual routers on the move. SIGCOMM Comput. Commun. Rev. **38**, 231 (2008)
10. Lo, S., Ammar, M., Zegura, E.: Design and analysis of schedules for virtual network migration. IFIP Netw. Conf. **2013**, 1–9 (2013)
11. Melo, M., Sargento, S., Carapinha, J.: Optimal virtual network migration: a step closer for seamless resource mobility. J. Netw. Comput. Appl. **64**, 124–136 (2016)
12. Xiao, A., Wang, Y., Meng, L., Qiu, X., Li, W.: Topology-aware remapping to survive virtual networks against substrate node failures. In: Network Operations and Management Symposium (APNOMS), 2013 15th Asia-Pacific, pp. 1–6. (2013)
13. Bradford, R., Kotsovinos, E., Feldmann, A., Schiöberg, H.: Live wide-area migration of virtual machines including local persistent state. In: Proceedings of the 3rd International Conference on Virtual Execution Environments—VEE'07, p. 169. ACM Press, New York, New York, USA (2007)
14. Melo, M., Carapinha, J., Sargento, S., Killat, U., Timm-Giel, A.: A Re-optimization Approach for Virtual Network Embedding. In: Timm-Giel, A., Strassner, J., Agüero, R., Sargento, S., and Pentikousis, K. (eds.) Mob. Netw. Manag. pp. 271–283. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
15. Aguilar-Fuster, C., Zangiabady, M., Zapata-Lara, J., Rubio-Loyola, J.: Online virtual network embedding based on virtual links' rate requirements. IEEE Trans. Netw. Serv. Manag. **15**, 1630–1644 (2018)
16. Herker, S., Khan, A., An, X.: Survey on survivable virtual network embedding problem and solutions. In: The 9th International Conference on Networking and Services (ICNS), pp. 99–104 (2013)
17. Hou, W., Ning, Z., Guo, L., Chen, Z., Obaidat, M.S.: Novel framework of risk-aware virtual network embedding in optical data center networks. IEEE Syst. J. **12**, 2473–2482 (2018)
18. Zhang, P., Yao, H., Liu, Y.: Virtual network embedding based on the degree and clustering coefficient information. IEEE Access. **4**, 8572–8580 (2016)
19. Caggiani Luizelli, M., Richter Bays, L., Salete Buriol, L., Pilla Barcellos, M., Paschoal Gaspary, L.: How physical network topologies affect virtual network embedding quality: a characterization study based on ISP and datacenter networks. J. Netw. Comput. Appl. **70**, 1–16 (2016)
20. Rubio-Loyola, J., Aguilar-Fuster, C., Toscano-Pulido, G., Mijumbi, R., Serrat-Fernandez, J.: Enhancing metaheuristic-based online embedding in network virtualization environments. IEEE Trans. Netw. Serv. Manag. **15**, 200–216 (2018)
21. Chowdhury, S.R., Ahmed, R., Shahriar, N., Khan, A., Boutaba, R., Mitra, J., Liu, L.: ReViNE: Reallocation of Virtual Network Embedding to eliminate substrate bottlenecks. In: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), pp. 116–124. IEEE, New York (2017)

22. Tran, P.N., Timm-Giel, A.: Reconfiguration of virtual network mapping considering service disruption. In: 2013 IEEE International Conference on Communications (ICC), pp. 3487–3492. IEEE, New York (2013)
23. Tarutani, Y., Ohsita, Y., Murata, M.: Virtual network reconfiguration for reducing energy consumption in optical data centers. J. Opt. Commun. Netw. **6**, 925 (2014)
24. Qiang, Z., Qiang, W., Sheng, F., Wu, L.: Heuristic survivable virtual network embedding based on node migration and link remapping. In: 2014 IEEE 7th Joint International Information Technology and Artificial Intelligence Conference, pp. 181–185. IEEE, New York (2014)
25. Zhao, S., Ji, X., Cheng, G., Hu, H.: Dynamic migration of virtual links. In: 2017 3rd IEEE International Conference on Computer and Communications (ICCC), pp. 12–16. IEEE, New York (2017)
26. Gao, L., Rouskas, G.N.: Virtual network reconfiguration with load balancing and migration cost considerations. In: IEEE INFOCOM 2018—IEEE Conference on Computer Communications, pp. 2303–2311. IEEE, New York (2018)
27. Araújo, S.M.A., Guidoni, D.L., de Souza, F.S.H., Mateus, G.R.: Managing virtual network embedding through reconfiguration and expansion. Simulation **95**, 1113–1125 (2019)
28. Deric, N., Varasteh, A., Basta, A., Blenk, A., Pries, R., Jarschel, M., Kellerer, W.: Coupling VNF orchestration and SDN virtual network reconfiguration. In: 2019 International Conference on Networked Systems (NetSys), pp. 1–3. IEEE, New York (2019)
29. Kellerer, W., Kalmbach, P., Blenk, A., Basta, A., Reisslein, M., Schmid, S.: Adaptable and data-driven softwarized networks: review, opportunities, and challenges. Proc. IEEE **107**, 711–731 (2019)
30. Al-Tam, F., Correia, N.: On load balancing via switch migration in software-defined networking. IEEE Access. **7**, 95998–96010 (2019)
31. AL-Tam, F., Ashrafi, M., Correia, N.: On Controllers' Utilization in Software-defined Networking by Switch Migration. In: Sucasas, V., Mantas, G., and Althunibat, S. (eds.) In: Broadband Communications, Networks, and Systems: 9th International EAI Conference, Broadnets 2018, Faro, Portugal, September 19–20, 2018, Proceedings. pp. 52–61. Springer International Publishing, Cham (2019)
32. Charmet, F., Blanc G., and Kiennert, C.,: Optimizing resource allocation for secure SDN-based virtual network migration. In: 2019 IEEE 18th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, 2019, pp. 1–10 (2019)
33. Guan, Y., Zong, Y., Liu, Y., Guo, L., Ning, Z., Rodrigues, J.J.P.C.: Virtual network embedding supporting user mobility in 5G metro/access networks. In: ICC 2019—2019 IEEE International Conference on Communications (ICC), pp. 1–7. IEEE, New York (2019)
34. Zhang, Z., Cao, H., Su, S., Li, W.: Energy aware virtual network migration. IEEE Trans. Cloud Comput. 1–1 (2020)
35. Phuong Nga Tran, Casucci, L., Timm-Giel, A.: Optimal mapping of virtual networks considering reactive reconfiguration. In: 2012 IEEE 1st International Conference on Cloud Networking (CLOUDNET), pp. 35–40. IEEE, New York (2012)
36. Farooq Butt, N., Chowdhury, M., Boutaba, R.: Topology-awareness and reoptimization mechanism for virtual network embedding. In: Crovella, M., Feeney, L.M., Rubenstein, D., Raghavan, S.V. (eds.) Networking 2010, pp. 27–39. Springer, Berlin Heidelberg (2010)
37. Masti, S.B., Raghavan, S.V.: Simulated annealing algorithm for virtual network reconfiguration. In: Proceedings of the 8th Euro-NF Conference on Next Generation Internet NGI 2012, pp. 95–102. IEEE, New York (2012)
38. Zangiabady, M., Garcia-Robledo, A., Gorricho, J., Serrat-Fernandez, J., Rubio-Loyola, J.: Self-adaptive Online Virtual Network Migration in Network Virtualization Environments. Trans. Emerg. Telecommun. Technol. **30**, 1–29 (2019)
39. Zangiabady, M., Rubio-Loyola, J.: Towards a QoS-Oriented Migration Management Approach for Virtualized Networks. In: Badonnel, R., Koch, R., Pras, A., Drašar, M., Stiller, B. (eds.) Management and Security in the Age of Hyperconnectivity, pp. 57–61. Springer International Publishing, Cham (2016)
40. Zangiabady, M., Aguilar-Fuster, C., Rubio-Loyola, J.: A virtual network migration approach and analysis for enhanced online virtual network embedding. In: 2016 12th International Conference on Network and Service Management (CNSM), pp. 324–329. IEEE, New York (2016)
41. Black, D., Fang, L., Kreeger, L., Napierala, M.: Problem statement: overlays for network virtualization. RFC Editor (2014)

42. Ando, T., Shimokuni, O., Asano, K.: Network virtualization for large-scale data centers. Fujitsu. Sci. Tech. J. **49**, 292–299 (2013)
43. Clayman, S., Galis, A., Mamatas, L.: Monitoring virtual networks with Lattice. In: 2010 IEEE/IFIP Network Operations and Management Symposium Workshops, pp. 239–246. IEEE, New York (2010)
44. Aceto, G., Botta, A., de Donato, W., Pescapè, A.: Cloud monitoring: A survey. Comput. Netw. **57**, 2093–2115 (2013)
45. Chang, X.L., Mi, X.M., Muppala, J.K.: Performance evaluation of artificial intelligence algorithms for virtual network embedding. Eng. Appl. Artif. Intell. **26**, 2540–2550 (2013)
46. Zhang, Z., Cheng, X., Su, S., Wang, Y., Shuang, K., Luo, Y.: A unified enhanced particle swarm optimization-based virtual network embedding algorithm. Int. J. Commun. Syst. **26**, 1054–1073 (2013)
47. Lee, K.S., Geem, Z.W.: A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Comput. Methods Appl. Mech. Eng. **194**, 3902–3933 (2005)
48. 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). pp. 1–6. IEEE, New York (2011)
49. Richter Bays, L., Ruas Oliveira, R.S., Buriol, L., Barcellos, M., Paschoal, Gaspary L.: A toolset for efficient privacy-oriented virtual network embedding and its instantiation on SDN/OpenFlow-based substrates. Comput Commun. **82**, 13–27 (2016)
50. Fajjari, I., Aitsaadi, N., Pujolle, G., Zimmermann, H.: VNR algorithm: a greedy approach for virtual networks reconfigurations. In: 2011 IEEE Global Telecommunications Conference—GLOBECOM 2011, pp. 1–6. IEEE, New York (2011)
51. Smith, D.K., Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network flows: theory, algorithms, and applications. J. Oper. Res. Soc. **45**, 1340 (1994)
52. Korkmaz, T., Krunz, M.: Multi-constrained optimal path selection. In: Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213). pp. 834–843. IEEE, New York (2001)
53. Raddwan, B., Al-Wagih, K., Al-Baltah, I.A., Alrshah, M.A., Al-Maqri, M.A.: Path mapping approach for network function virtualization resource allocation with network function decomposition support. Symmetry. **11**, 1173 (2019)
54. Jiang, Y., Lan, J., Wang, Z., Deng, Y.: Embedding and reconfiguration algorithms for service aggregation in network virtualization. Int. J. Commun. Syst. **29**, 33–46 (2016)
55. Hartman, A.: Software and hardware testing using combinatorial covering suites. In: Golumbic, M.C., Hartman, I.B.-A. (eds.) Graph Theory, Combinatorics and Algorithms, pp. 237–266. Springer-Verlag, New York (2005)
56. National Institute of Standards, (NIST), T.: Covering Array Tables
57. Hogg, R., Elliot, T.: Probability and statistical inference. (1988)
58. Bland, J.M., Altman, D.G.: Multiple significance tests: the Bonferroni method. BMJ **310**, 170 (1995)
59. Shapiro, S.S., Wilk, M.B.: An analysis of variance test for normality (complete samples). Biometrika **52**, 591–611 (1965)
60. Shong, N.: Pearson's versus Spearman's and Kendall's correlation coefficients for continuous data, (2010)
61. Friedman, M.: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J. Am. Stat. Assoc. **32**, 675 (1937)
62. Landa Becerra, R., Coello, C.A.C.: Cultured differential evolution for constrained optimization. Comput. Methods Appl. Mech. Eng. **195**, 4303–4322 (2006)
63. Han, P., Guo, L., Liu, Y.: Virtual network embedding in SDN/NFV based fiber-wireless access network. In: 2016 International Conference on Software Networking (ICSN), pp. 1–5. IEEE, New York (2016)
64. Jain, R., Paul, S.: Network virtualization and software defined networking for cloud computing: a survey. IEEE Commun. Mag. **51**, 24–31 (2013)

Mahboobeh Zangiabady holds a Ph.D. in computer science from the Centre for Research and Advanced Studies of the National Polytechnic Institute. Her research interests cover network virtualization, QoS, resource management, machine learning, and Network Functions Virtualization, Software-Defined Networks (NFV/SDN).

Alberto Garcia-Robledo is a Computer Science Ph.D. and Fellow Researcher of the National Council for Science and Technology of Mexico. He has collaborated with the Geospatial Data Center of the Massachusetts Institute of Technology. His research interests include Network Science, Data Analytics, High Performance Computing and Applied Graph Theory.

Christian Aguilar-Fuster received his M.Sc. degree in computer science from the Center for Research and Advanced Studies of the National Polytechnic Institute of Mexico, where he is currently pursuing a Ph.D. degree. His research interests focus on network management, network virtualization, wireless networks, and evolutionary computation.

Javier Rubio-Loyola is a research scientist and Director of the Campus Tamaulipas of the Centre for Research and Advanced Studies of the National Polytechnic Institute of Mexico. His research interests focus on network management, Network Functions Virtualization, Software-Defined Networks (NFV/SDN) and beyond 5G networks and services.

Affiliations

Mahboobeh Zangiabady[1] · Alberto Garcia-Robledo[2] · Christian Aguilar-Fuster[1] · Javier Rubio-Loyola[1]

[1] Centre for Research and Advanced Studies of the National Polytechnic Institute, Carretera Victoria-Soto la Marina Km. 5.5, 87130. Cd. Victoria, Tamaulipas, Mexico

[2] Cátedras CONACyT, Center for Research in Geography and Geomatics (CentroGeo), Av. Insurgentes Sur 1582, Col. Crédito Constructor, Alcaldía Benito Juárez, 03940 Ciudad de México, Mexico