Virtual Network Mapping: A Graph Pattern Matching Approach

Yang Cao [1,2] Wenfei Fan [1,2] Shuai Ma [1]

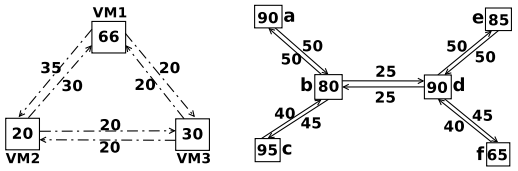
[1]RCBD&SKLSDE Lab, Beihang University, China
[2]University of Edinburgh, UK
{caoyang, mashuai}@act.buaa.edu.cn wenfei@inf.ed.ac.uk

Abstract. Virtual network mapping (VNM) is to build a network on demand by deploying virtual machines in a substrate network, subject to constraints on capacity, bandwidth and latency. It is critical to data centers for coping with dynamic cloud workloads. This paper shows that VNM can be approached by graph pattern matching, a well-studied database topic. (1) We propose to model a virtual network request as a graph pattern carrying various constraints, and treat a substrate network as a graph in which nodes and edges bear attributes specifying their capacity. (2) We show that a variety of mapping requirements can be expressed in this model, such as virtual machine placement, network embedding and priority mapping. (3) In this model, we formulate VNM and its optimization problem with a mapping cost function. We establish complexity bounds of these problems for various mapping constraints, ranging from PTIME to NP-complete. For intractable optimization problems, we further show that these problems are approximation-hard, *i.e.,*NPO-complete in general and APX-hard even for special cases.

1 Introduction

Virtual network mapping (VNM) is also known as virtual network embedding or assignment. It takes as input (1) a *substrate network* (SN, a physical network), and (2) a *virtual network* (VN) specified in terms of a set of virtual nodes (machines or routers, denoted as VMs) and their virtual links, along with constraints imposed on the capacities of the nodes (*e.g.,* CPU and storage) and on the links (*e.g.,* bandwidth and latency). VNM is to deploy the VN in the SN such that virtual nodes are hosted on substrate nodes, virtual links are instantiated with physical paths in the SN, and the constraints on the virtual nodes and links are satisfied.

VNM is critical to managing big data. Big data is often distributed to data centers [23,26]. However, data center networks become *the bottleneck* for dynamic cloud workloads of querying and managing the data. In traditional networking platforms, network resources are manually configured with static policies, and new workload provisioning often takes days or weeks [1]. This highlights the need for VNM, to automatically deploy virtual networks in a data center network in



(a) VN Request (b) SN

Fig. 1. VN requests found in practice

response to real-time requests. Indeed, VNM is increasingly employed in industry, *e.g.,* Amazon's EC2 [2], VMware Data Center [3] and Big Switch Networks [1]. It has proven effective in increasing server utilization and reducing server provisioning time (from days or weeks to minutes), server capital expenditures and operating expenses [1]. There has also been a host of work on virtualization techniques for big data [23] and database systems [7, 24].

Several models have been proposed to specify VNM in various settings:

(1) *Virtual machine placement* (VMP): it is to find a mapping f from virtual machines in a VN to substrate nodes in an SN such that for each VM v, its capacity is no greater than that of $f(v)$, *i.e.,* $f(v)$ is able to conduct the computation of the VM v that it hosts [12].

(2) *Single-path* VN *embedding* ($\mathsf{VNE_{SP}}$): it is to find

(a) an injective mapping f_v that maps nodes in VN to nodes in SN, subject to node capacity constraints; and
(b) a function that maps a virtual link (v, v') in VN to a path from $f_v(v)$ to $f_v(v')$ in SN that satisfies a bandwidth constraint, *i.e.,* the bandwidth of each link in the SN is no smaller than the sum of the bandwidth requirements of all those virtual links that are mapped to a path containing it [20].

(3) *Multi-path* VN *embedding* ($\mathsf{VNE_{MP}}$): it is to find a node mapping f_v as in $\mathsf{VNE_{SP}}$ and a function that maps each virtual link (v, v') to *a set* of paths from $f_v(v)$ to $f_v(v')$ in SN, subject to bandwidth constraints [14, 25].

However, there are a number of VN requests commonly found in practice, which cannot be expressed in any of these models, as illustrated by the following.

Example 1. Consider a VN request and an SN, depicted in Figures 1(a) and 1(b), respectively. The VN has three virtual nodes $\mathsf{VM_1}$, $\mathsf{VM_2}$ and $\mathsf{VM_3}$, each specifying a capacity constraint, along with a constraint on each virtual link. In the SN, each substrate node bears a resource capacity and each connection (edge) has an attribute, indicating either bandwidth or latency. Consider the following cases. *(1) Mapping with latency constraints* ($\mathsf{VNM_L}$). Assume that the numbers attached to the virtual nodes and links in Fig. 1(a) denote requirements on CPUs and latencies for SN, respectively. Then the VNM problem, denoted by $\mathsf{VNM_L}$, aims to map each virtual node to a substrate node with sufficient computational power, and to map each virtual link (v, v') in the VN to a path in the SN such that its latency, *i.e., the sum* of the latencies of the edges on the path, does not exceed the latency specified for (v, v'). The need for studying VNM_L arises from

latency sensitive applications such as multimedia transmitting networks [21], which concern latency rather than bandwidth.

(2) Priority mapping ($\mathsf{VNM_P}$). Assume that the constraints on the nodes in Fig. 1(a) are CPU capacities, and constraints imposed on edges are bandwidth capacities. Here the VNM problem, denoted by $\mathsf{VNM_P}$, is to map each virtual node to a node in SN with sufficient CPU capacity, and each virtual link (v, v') in the VN to a path in SN such that the *minimum* bandwidth of all edges on the path is no less than the bandwidth specified for (v, v'). The need for this is evident in many applications [4], we want to give different priorities at run time to virtual links that share some physical links, and require the mapping only to provide bandwidth guarantee for the connection with the highest priority.

(3) Mapping with node sharing ($\mathsf{VNE_{SP(NS)}}$). Assume that the numbers attached to the virtual nodes and links in Fig. 1(a) denote requirements on CPUs and bandwidths for SN, respectively. Then $\mathsf{VNE_{SP(NS)}}$ is an extension of the single-path VN embedding ($\mathsf{VNE_{SP}}$) by supporting node sharing, *i.e.,* by allowing mapping multiple virtual nodes to the same substrate node, as needed by X-Bone [6].

There is also practical need for extending other mappings with node sharing, such as virtual machine placement (VMP), latency mapping ($\mathsf{VNM_L}$), priority mapping $\mathsf{VNM_P}$ and multi-path VN embedding ($\mathsf{VNE_{MP}}$). We denote such an extension by adding a subscript NS.

Observe that (a) VNM varies from practical requirements, *e.g.,* when latency, high-priority connections and node sharing are concerned; (b) Existing models are not capable of expressing such requirements; indeed, none of them is able to specify $\mathsf{VNM_L}$, $\mathsf{VNM_P}$ or $\mathsf{VNE_{SP(NS)}}$; And (c) it would be an overkill to develop a model for each of the large variety of requirements, and to study it individually.

As suggested by the example, we need a generic model to express virtual network mappings in various practical settings, including both those already studied (*e.g.,*VMP, $\mathsf{VNE_{SP}}$ and $\mathsf{VNE_{MP}}$) and those being overlooked (*e.g.,* $\mathsf{VNM_L}$, $\mathsf{VNM_P}$ and $\mathsf{VNE_{SP(NS)}}$). The uniform model allows us to characterize and compare VNMs in different settings, and better still, to study generic properties that pertain to all the variants. Among these are the complexity and approximation analyses of VNMs, which are obviously important but have not yet been systematically studied by and large.

Contributions & Roadmap. This work takes a step toward providing a uniform model to characterize VNMs. We show that VNMs, an important problem for managing big data, can actually be tackled by graph pattern matching techniques, a database topic that has been well studied. We also provide complexity and approximation bounds for VNMs. Moreover, for intractable VNM cases, we develop effective heuristic methods to find high-quality mappings.

(1) We propose a generic model to express VNMs in terms of graph pattern matching [18] (Section 2). In this model a VN request is specified as a graph pattern, bearing various constraints on nodes and links defined with aggregation functions, and an SN is simply treated as a graph with attributes associated with its nodes and edges. The decision and optimization problems for VNMs are then simply graph pattern matching problems. We also show that the model is

able to express VNMs commonly found in practice, including all the mappings we have seen so far (Section 3).

(2) We establish complexity and approximation bounds for VNMs (Section 4). We give a uniform upper bound for the VNM problems expressed in this model, by showing that all these problems are in NP. We also show that VNM is polynomial time (PTIME) solvable if only node constraints are present (VMP), but it becomes NP-complete when either node sharing is allowed or constraints on edges are imposed. Moreover, we propose a VNM cost function and study optimization problems for VNM based on the metric. We show that the optimization problems are intractable in most cases and worse still, are NPO-complete in general and APX-hard [10] for special cases. To the best of our knowledge, these are among the first complexity and approximation results on VNMs.

We contend that these results are useful for developing virtualized cloud data centers for querying and managing big data, among other things. By modeling VNM as graph pattern matching, we are able to characterize various VN requests with different classes of graph patterns, and study the expressive power and complexity of these graph pattern languages. The techniques developed for graph pattern matching can be leveraged to study VNMs. Indeed, the proofs of some of the results in this work capitalize on graph pattern techniques. Furthermore, the results of this work are also of interest to *the study of graph pattern matching* [18].

2 Graph Pattern Matching Model

Below we first represent virtual networks (VNs) and substrate networks (SNs) as weighted directed graphs. We then introduce a generic model to express virtual network mapping (VNM) in terms of graph pattern matching [18].

2.1 Substrate and Virtual Networks

An SN consists of a set of substrate nodes connected with physical links, in which the nodes and links are associated with resources of a certain capacity, *e.g.,* CPU and storage capacity for nodes, and bandwidth and latency for links. A VN is specified in terms of a set of virtual nodes and a set of virtual links, along with requirements on the capacities of the nodes and the capacities of the links. Both VNs and SNs can be naturally modeled as weighted directed graphs.

Weighted directed graphs. A *weighted directed graph* is defined as $G = (V, E, f_V, f_E)$, where (1) V is a finite set of nodes; (2) $E \subseteq V \times V$ is a set of edges, in which (v, v') denotes an edge from v to v'; (3) f_V is a function defined on V such that for each node $v \in V$, $f_V(v)$ is a positive rational number; and similarly, (4) f_E is a function defined on E.

Substrate networks. A *substrate network* (SN) is a weighted directed graph $G_S = (V_S, E_S, f_{V_S}, f_{E_S})$, where (1) V_S and E_S denote sets of substrate nodes and physical links (directly connected), respectively; and (2) the functions f_{V_S} and f_{E_S} denote resource capacities on the nodes (*e.g.,* CPU) and links (*e.g.,* bandwidth and latency), respectively.

Virtual networks. A *virtual network* (VN) is specified as a weighted directed graph $G_P = (V_P, E_P, f_{V_P}, f_{E_P})$, where (1) V_P and E_P denote virtual nodes and links, and (2) f_{V_P} and f_{E_P} are functions defined on V_P and E_P in the same way as in substrate networks, respectively.

Example 2. The SN depicted in Fig. 1(b) is a weighted graph G_S, where (1) the node set is $\{\mathsf{a}, \mathsf{b}, ..., \mathsf{f}\}$; (2) the edges include the directed edges in the graph; (3) the weights associated with nodes indicate CPU capacities; and (4) the weights of edges denote bandwidth or latency capacities. Figure 1(a) shows a VN, where (1) the node set is $\{\mathsf{VM}_1, \mathsf{VM}_2, \mathsf{VM}_3\}$; (2) the edge set is $\{(\mathsf{VM}_\mathsf{i}, \mathsf{VM}_\mathsf{j}) \mid \mathsf{i}, \mathsf{j} = 1, 2, 3\}$; (3) $f_{V_P}(\mathsf{VM}_1) = 66$, $f_{V_P}(\mathsf{VM}_2) = 20$, $f_{V_P}(\mathsf{VM}_3) = 30$; and (4) the function f_{E_P} is defined on the edge labels. As will be seen when we define the notion of VN requests, the labels indicate requirements on deploying the VN in an SN.

Paths. A *path* ρ from node u_0 to u_n in an SN G_S is denoted as $(u_0, u_1, \ldots, u_n)$, where (a) $u_i \in V_S$ for each $i \in [0, n]$, (b) there exists an edge $e_i = (u_{i-1}, u_i)$ in E_S for each $i \in [1, n]$, and moreover, (c) for all $i, j \in [0, n]$, if $i \neq j$, then $u_i \neq u_j$. We write $e \in \rho$ if e is an edge on ρ. When it is clear from the context, we also use ρ to denote the set of edges on the path, *i.e.,* $\{e_i \mid i \in [1, n]\}$.

2.2 Virtual Network Mapping

Virtual network mapping (VNM) from a VN G_P to an SN G_S is specified in terms of a node mapping, an edge mapping and a VN request. The VN request imposes constraints on the node mapping and edge mapping, defining their semantics. We next define these notions.

A *node mapping* from G_P to G_S is a pair (g_V,r_V) of functions, where g_V maps the set V_P of virtual nodes in G_P to the set V_S of substrate nodes in G_S, and for each v in V_P, if $g_V(v) = u$, $r_V(v, u)$ is a positive number. Intuitively, function r_V specifies the amount of resource of the substrate node u that is allocated to the node v.

For each edge (v, v') in G_P, we use $P(v, v')$ to denote the set of paths from $g_V(v)$ to $g_V(v')$ in G_S. An *edge mapping* from G_P to G_S is a pair (g_E, r_E) of functions such that for each edge $(v, v') \in E_P$, $g_E(v, v')$ is a subset of $P(v, v')$, and r_E attaches a positive number to each pair (e, ρ) if $e \in E_P$ and $\rho \in g_E(e)$. Intuitively, $r_E(e, \rho)$ is the amount of resource of the physical path ρ allocated to the virtual link e.

VN requests. A VN *request* to an SN G_S is a pair $(G_P, \mathcal{C})$, where G_P is a VN, and $\mathcal{C}$ is a set of constraints such that for a pair $((g_V, r_V), (g_E, r_E))$ of node and edge mappings from G_P to G_S, each constraint in $\mathcal{C}$ has one of the forms below:

(1) for each $v \in V_P$, $f_{V_P}(v) \leq r_V(v, g_V(v))$;
(2) for each $u \in V_S$, $f_{V_S}(u) \geq \mathsf{sum}(N(u))$, where $N(u)$ is $\{\!|r_V(v, u) \mid v \in V_P,\ g_V(v) = u|\!\}$, a bag (an unordered collection of elements with repetitions) determined by virtual nodes in G_P hosted by u;
(3) for each $e \in E_P$, $f_{E_P}(e)$ op $\mathsf{agg}(Q(e))$, where $Q(e)$ is $\{\!|r_E(e, \rho) \mid \rho \in g_E(e)|\!\}$, a bag collecting physical paths ρ that instantiate e; here op is either the comparison operator $\leq$ or $\geq$, and $\mathsf{agg}()$ is one of the aggregation functions min, max and sum;

(4) for each $e' \in E_S$, $f_{E_S}(e') \geq \mathsf{sum}(M(e'))$, where $M(e')$ is $\{\!|r_E(e,\rho) \mid e \in E_P, \rho \in g_E(e), e' \in \rho|\!\}$, a bag collecting those virtual links that are instantiated by a physical link ρ containing e'; and
(5) for each $e \in E_P$ and $\rho \in g_E(e)$, $r_E(e,\rho)$ op $\mathsf{agg}(U(\rho))$ where $U(\rho)$ is $\{\!|f_{E_S}(e') \mid e' \in \rho|\!\}$, a bag of all edges on a physical path that instantiate e.

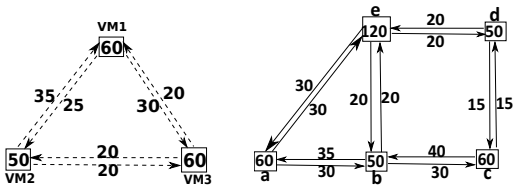
Constraints in a VN request are classified as follows.
Node constraints: Constraints of form (1) or (2). Intuitively, a constraint of form (1) assures that when a virtual node v is hosted by a substrate node u, u must provide adequate resource. A constraint of form (2) asserts that when a substrate node u hosts (possibly multiple) virtual nodes, u must have sufficient capacity to accommodate all those virtual nodes. When u hosts at most one virtual node, *i.e.,* if node sharing is not allowed, then $|N(u)| \leq 1$, where we use $|N(u)|$ to denote the number of virtual nodes hosted by u.
Edge constraints: Constraints of form (3), (4) or (5). Constraints of form (3) assure that when a virtual link e is mapped to a set of physical paths in the SN, those physical paths together satisfy the requirements (on bandwidths or latencies) of e. We denote by $|Q(e)|$ the number of physical paths to which e is mapped. Those of form (4) assert that for each physical link e', it must have sufficient bandwidth to accommodate those of all the virtual links that are mapped to some physical path containing e'. Those of form (5) assure that when a virtual link e is mapped to a set of paths, for each ρ in the set, the resource of ρ allocated to e must be consistent with the capacities of the physical links on ρ, *e.g.,* may not exceed the minimum bandwidth of the physical links on ρ.

VNM. We say that a VN request $(G_P, \mathcal{C})$ can be *mapped to* an SN G_S, denoted by $G_P \rhd_{\mathcal{C}} G_S$, if there exists a pair $((g_V, r_V), (g_E, r_E))$ of node and edge mappings from G_P to G_S such that all the constraints of $\mathcal{C}$ are satisfied, *i.e.,* the functions g_V and g_E satisfy all the inequalities in $\mathcal{C}$.

The VNM *problem* is to determine, given a VN request $(G_P, \mathcal{C})$ and an SN G_S, whether $G_P \rhd_{\mathcal{C}} G_S$.

3 Case study

All the VNM requirements in the Introduction (Section 1) can be expressed in our model, by treating VN request as a pattern and SN as a graph. Below we present a case study.
Case 1: Virtual machine placement. VMP can be expressed as a VN request in which only node constraints are present. It is to find an injective mapping (g_V, r_V) from virtual nodes to substrate nodes (hence $|N| \leq 1$) that satisfies the node constraints, while imposing no constraints on edge mapping.
Case 2: Priority mapping. VNM_P can be captured as a VN request specified as $(G_P, \mathcal{C})$, where $\mathcal{C}$ consists of (a) node constraints of forms (1) and (2), and (b) edge constraints of form (3) when op is $\leq$ and agg is max, and form (5) when op is $\leq$ and agg is min. It is to find an injective node mapping (g_V, r_V) and an edge mapping (g_E, r_E) such that for each virtual link e, $g_E(e)$ is a single path (hence $|Q(e)| = 1$). Moreover, it requires that the capacity of each virtual node v does



(a) VN Request (b) SN

Fig. 2. VN request and SN for case study

not exceed the capacity of the substrate node that hosts v. When a virtual link e is mapped to a physical path ρ, the *bandwidth* of each edge on ρ is no less than that of e, *i.e.,* ρ suffices to serve any connection *individually*, including the one with the highest priority when ρ is allocated to the connection.

Example 3. Consider the VN given in Fig. 1(a) and the SN of Fig. 1(b). Constraints for priority mapping can be defined as described above, using the node and edge labels (on bandwidths) in Fig. 1(a). There exists a priority mapping from the VN to the SN. Indeed, one can map $\mathsf{VM}_1, \mathsf{VM}_2$ and VM_3 to b, a and d, respectively, and map the virtual links to the shortest physical paths uniquely determined by the node mapping, *e.g.,* $(\mathsf{VM}_1, \mathsf{VM}_2)$ is mapped to (b, a).

Case 3: Single-path VN *embedding.* A $\mathsf{VNE}_{\mathsf{SP}}$ request can be specified as $(G_P, \mathcal{C})$, where $\mathcal{C}$ consists of (a) node constraints of forms (1) and (2), and (b) edge constraints of form (3) when op is $\leq$ and agg is sum, and edge constraints of forms (4) and (5) when op is $\leq$ and agg is min. It differs from $\mathsf{VNM}_{\mathsf{P}}$ in that for each physical link e', it requires the *bandwidth* of e' to be no less than *the sum of* bandwidths of *all* those virtual links that are instantiated via e'. In contrast to VNM_P that aims to serve the connection with the highest priority at a time, $\mathsf{VNE}_{\mathsf{SP}}$ requires that each physical link has enough capacity to serve *all connections* sharing the physical link at the same time.

Similarly, multi-path VN embedding (denoted by $\mathsf{VNE}_{\mathsf{MP}}$) can be expressed as a VN request. It is the same as $\mathsf{VNE}_{\mathsf{SP}}$ except that a virtual link e can be mapped to *a set* $g_E(e)$ of physical paths. When taken together, the paths in $g_E(e)$ provide sufficient bandwidth required by e.

When node sharing is allowed in $\mathsf{VNE}_{\mathsf{SP}}$, *i.e.,* for single-path embedding with node sharing ($\mathsf{VNE}_{\mathsf{SP(NS)}}$), a VN request is specified similarly. Here a substrate node u can host multiple virtual nodes (hence $|N(u)| \geq 0$) such that the sum of the capacities of all the virtual nodes does not exceed the capacity of u. Similarly, one can also specify multi-path VN embedding with node sharing ($\mathsf{VNE}_{\mathsf{MP(NS)}}$).

Example 4. Consider the VN of Fig. 2(a), and the SN of Fig. 2(b). There is a $\mathsf{VNE}_{\mathsf{SP}}$ from the VN to the SN, by mapping $\mathsf{VM}_1, \mathsf{VM}_2, \mathsf{VM}_3$ to $\mathsf{a}, \mathsf{b}, \mathsf{e}$, respectively, and mapping the VN edges to the shortest paths in the SN determined by the node mapping. There is also a multi-path embedding $\mathsf{VNE}_{\mathsf{MP}}$ from the VN to the SN, by mapping $\mathsf{VM}_1, \mathsf{VM}_2$ and VM_3 to a, c and e, respectively. For the virtual links, $(\mathsf{VM}_1, \mathsf{VM}_2)$ can be mapped to the physical path $(\mathsf{a}, \mathsf{b}, \mathsf{c})$, $(\mathsf{VM}_1, \mathsf{VM}_3)$ to (a, e), and $(\mathsf{VM}_3, \mathsf{VM}_2)$ to two paths $\rho_1 = (\mathsf{e}, \mathsf{b}, \mathsf{c})$ and $\rho_2 = (\mathsf{e}, \mathsf{d}, \mathsf{c})$ with

$r_E((\mathsf{VM}_3, \mathsf{VM}_2), \rho_1) = 5$ and $r_E((\mathsf{VM}_3, \mathsf{VM}_2), \rho_2) = 15$; similarly for the other virtual links.

One can verify that the VN of Fig. 2(a) allows no more than one virtual node to be mapped to the same substrate node in Fig. 2(b). However, if we change the bandwidths of the edges connecting a and e in SN from 30 to $f_{V_S}(\mathsf{a}, \mathsf{e}) = 40$ and $f_{V_S}(\mathsf{e}, \mathsf{a}) = 50$, then there exists a mapping from the VN to the SN that supports node sharing. Indeed, in this setting, one can map both $\mathsf{VM}_1, \mathsf{VM}_2$ to e and map VM_3 to a; and map the virtual edges to the shortest physical paths determined by the node mapping; for instance, both $(\mathsf{VM}_1, \mathsf{VM}_3)$ and $(\mathsf{VM}_2, \mathsf{VM}_3)$ can be mapped to (e, a).

Case 4: Latency constrained mapping. A VNM_L request is expressed as $(G_P, \mathcal{C})$, where $\mathcal{C}$ consists of (a) node constraints of forms (1) and (2), and (b) edge constraints of form (3) when op is $\geq$ and agg is min, and of form (5) when op is $\geq$ and agg is sum. It is similar to VNE_SP except that when a virtual link e is mapped to a physical path ρ, it requires ρ to satisfy the *latency* requirement of e, *i.e.,* the sum of the latencies of the edges on ρ does not exceed that of e.

Example 5. One can verify that there is no latency mapping of the VN shown in Fig. 1(a) to the SN in Fig. 1(b). However, if we change the constraints on the virtual links of the VN request to: $(\mathsf{VM}_1, \mathsf{VM}_2) = 50$, $(\mathsf{VM}_2, \mathsf{VM}_1) = 55$, $(\mathsf{VM}_1, \mathsf{VM}_3) = (\mathsf{VM}_3, \mathsf{VM}_1) = 120$ and $(\mathsf{VM}_2, \mathsf{VM}_3) = (\mathsf{VM}_3, \mathsf{VM}_2) = 60$, then there is a mapping from the VN to the SN. We can map $\mathsf{VM}_1, \mathsf{VM}_2, \mathsf{VM}_3$ to c, b, a, respectively, and map the edges to the shortest physical paths determined by the node mapping.

4 Complexity and Approximation

We next study fundamental issues associated with virtual network mapping. We first establish the complexity bounds of the VNM problem in various settings, from PTIME to NP-complete. We then introduce a cost metric for virtual network mapping, formulate optimization problems based on the function, and finally, give the complexity bounds and approximation hardness of the optimization problems. Due to the space constraint, we defer the detailed proofs to [5].

4.1 The Complexity of VNM

We provide an upper bound for the VNM problem in the general setting, by showing it is in NP. We also show that the problem is in PTIME when only node constraints are present. However, when node sharing or edge constraints are imposed, it becomes NP-hard, even when both virtual and substrate networks are directed acyclic graphs (DAGs). That is, node sharing and edge constraints make our lives harder.

Theorem 1. *The virtual network mapping problem is*
(1) in NP regardless of what constraints are present;

(2) in PTIME when only node constraints are present, without node sharing, i.e.,VMP *is in PTIME; However,*
(3) it becomes NP-complete when node sharing is requested, i.e., $\mathsf{VMP}_{(\mathsf{NS})}$, $\mathsf{VNM}_{\mathsf{P}(\mathsf{NS})}$, $\mathsf{VNM}_{\mathsf{L}(\mathsf{NS})}$, $\mathsf{VNE}_{\mathsf{SP}(\mathsf{NS})}$ *and* $\mathsf{VNE}_{\mathsf{MP}(\mathsf{NS})}$ *are NP-complete; and*
(4) it is NP-complete in the presence of edge constraints; i.e., $\mathsf{VNM}_{\mathsf{P}}$, $\mathsf{VNM}_{\mathsf{L}}$, $\mathsf{VNE}_{\mathsf{SP}}$ *and* $\mathsf{VNE}_{\mathsf{MP}}$ *are intractable.*

All the results hold when both VNs *and* SNs *are* DAG*s.*

4.2 Approximation of Optimization Problems

In practice, one typically wants to find a VNM mapping with "the lowest cost". This highlights the need for introducing a function to measure the cost of a mapping and studying its corresponding optimization problems.

A Cost Function. Consider an SN $G_S = (V_S, E_S, f_{V_S}, f_{E_S})$, and a VN request $(G_P, \mathcal{C})$, where $G_P = (V_P, E_P, f_{V_P}, f_{E_P})$. Assume a positive number associated with all nodes v and links e in G_S, denoted by $w(v)$ and $w(e)$, respectively, that indicates the price of the resources in the SN.

Given a pair $((g_V, r_V), (g_E, r_E))$ of node and edge mappings from $(G_P, \mathcal{C})$ to G_S, its *cost* $c((g_V, r_V), (g_E, r_E))$ is defined as

$$c((g_V, r_V), (g_E, r_E)) = \sum_{v \in V_P} h_V(g_V, r_V, v) \cdot w(g_V(v)) + \sum_{e' \in E_S} h_E(g_E, r_E, e') \cdot w(e'),$$

where (1) $h_V(g_V, r_V, v) = r_V(v, g_V(v))/f_{V_S}(g_V(v))$,
(2) $h_E(g_E, r_V, e') = \sum_{e \in E_P, \rho \in g_E(e), e' \in \rho} r_E(e, \rho)/f_{E_S}(e')$ when the resource of physical links is bandwidth, and
(3) when latency is concerned, $h_E(g_E, r_V, e')$ is 1 if there exists $e \in E_P$ such that $e' \in g_E(e)$, and 0 otherwise.

Intuitively, h_V indicates that the more CPU resource is allocated, the higher the cost it incurs; similarly for h_E when bandwidth is concerned. When latency is considered, the cost of the edge mapping is determined only by g_E, whereas the resource allocation function r_E is irrelevant.

The cost function is motivated by economic models of network virtualization [13]. It is justified by Web hosting and cloud storage [11], which mainly sell CPU power or storage services of nodes, and by virtual network mapping, which also sells bandwidth of links [14]. It is also to serve cloud provision in virtualized data center networks [19], for which dynamic routing strategy (latency) is critical while routing congestion (bandwidth allocation) is considered secondary.

Minimum Cost Mapping. We now introduce optimization problems for virtual network mapping.

The *minimum cost mapping* problem is to find, given a VN request and an SN, a mapping $((g_V, r_V), (g_E, r_E))$ from the VN to the SN such that its cost based on the function above is minimum among all such mappings.

The decision problem for minimum cost mapping is to decide, given a number (bound) K, a VN request and an SN, whether there is a mapping $((g_V, r_V), (g_E, r_E))$ from the VN to the SN such that its cost is no larger than K.

We shall refer to the minimum cost mapping problem and its decision problem interchangeably in the sequel.

Complexity and Approximation. We next study the minimum cost mapping problem for all cases given before. Having seen Theorem 1, it is not surprising that the optimization problem is intractable in most cases. This motivates us to study their efficient approximation algorithms with performance guarantees.

Unfortunately, the problem is hard to approximate in most cases. The results below tell us that when node sharing is requested or edge constraints are present, minimum cost mapping is beyond reach in practice for approximation.

Theorem 2. *The minimum cost mapping problem is*
(1) in PTIME for VMP *without node sharing; however, when node sharing is requested,* i.e., *for* $\mathsf{VMP}_{(\mathsf{NS})}$*, it becomes NP-complete and is APX-hard;*
(2) NP-complete and NPO-complete for VNM_P*,* $\mathsf{VNE}_{\mathsf{SP}}$*,* $\mathsf{VNE}_{\mathsf{MP}}$*,* VNM_L*,* $\mathsf{VNM}_{\mathsf{P}(\mathsf{NS})}$*,* $\mathsf{VNE}_{\mathsf{SP}(\mathsf{NS})}$*,* $\mathsf{VNE}_{\mathsf{MP}(\mathsf{NS})}$*,* $\mathsf{VNM}_{\mathsf{L}(\mathsf{NS})}$*; And*
(3) APX-hard when there is a unique node mapping in the presence of edge constraints. In particular, $\mathsf{VNM}_{\mathsf{P}}$ *does not admit* $\ln(|V_P|)$*-approximation, unless* $P = NP$*.*

The NPO-hardness results remain intact even when both VNs *and* SNs *are* DAG*s.*

Here NPO is the *class* of all NP optimization problems (cf. [10]). An NPO-complete problem is NP-hard to optimize, and is among the hardest optimization problems. APX is the *class* of problems that allow PTIME approximation algorithms with a constant approximation ratio (cf. [10]).

Heuristic algorithms. These above results tell us that it is beyond reach in practice to find PTIME algorithms for VNMs with edge constraints such as $\mathsf{VNM}_{\mathsf{P}}$ and $\mathsf{VNE}_{\mathsf{SP}}$, or to find efficient approximation algorithms with decent performance guarantees. In light of these, we study heuristic algorithms.

We develop heuristic algorithms for priority mapping $\mathsf{VNM}_{\mathsf{P}}$, with node sharing or not [5]. We focus on $\mathsf{VNM}_{\mathsf{P}}$ since it is needed in, *e.g.,* internet-based virtualized infrastructure computing platform (iVIC [4]). Our algorithm reduces unnecessary computation by minimizing VNs requests and utilizing auxiliary graphs of SNs [5]. While several algorithms are available for VN embedding (*e.g.,* [20]), no previous work has studied algorithms for $\mathsf{VNM}_{\mathsf{P}}$. We encourage interested readers to look into [5] for the detailed introduction and experimental study of these algorithms.

5 Related Work

Virtualization techniques have been investigated for big data processing [23] and database applications [7,8,24]. However, none of these has provided a systematic study of VNM, by modeling VNM as graph pattern matching. The only exception is [20], which adopted subgraph isomorphism for VNM, a special case of the generic model proposed in this work. Moreover, complexity and approximation

analyses associated with VNM have not been studied for cloud computing in database applications.

Several models have been developed for VNM. (a) The VM placement problem (VMP, [12]) is to map a set of VMs onto an SN with constraints on node capacities. (b) Single-path VN embedding ($\mathsf{VNE}_{\mathsf{SP}}$, [22]) is to map a VN to an SN by a node-to-node injection and an edge-to-path function, subject to constraints on the CPU capacities of nodes and constraints on the bandwidths of physical connections. (c) Different from $\mathsf{VNE}_{\mathsf{SP}}$, multi-path embedding ($\mathsf{VNE}_{\mathsf{MP}}$, [14, 25]) allows an edge of a VN to be mapped to multiple parallel paths of an SN such that the sum of the bandwidth capacities of those paths is no smaller than the bandwidth of that edge. (d) While graph layout problems are similar to VN mapping, they do not have bandwidth constraints on edges but instead, impose certain topological constraints (see [15] for a survey). In contrast to our work, these models are studied for specific domains, and no previous work has studied generic models to support various VN requests that commonly arise in practice.

Very few complexity results are known for VNM. The only work we are aware of is [9], which claimed that the testbed mapping problem is NP-hard in the presence of node types and some links with infinite capacity. Several complexity and approximation results are established for graph pattern matching (see [18] for a survey). However, those results are for edge-to-edge mappings, whereas VNM typically needs to map virtual links to physical paths. There have been recent extensions to support edge-to-path mappings for graph pattern matching [16,17], with several intractability and approximation bounds established there. Those differ from this work in that either no constraints on links are considered [17], or graph simulation is adopted [16], which does not work for VNM. The complexity and approximation bounds developed in this work are among the first results that have been developed for VNM in cloud computing.

6 Conclusion

We have proposed a model to express various VN requests found in practice, based on graph pattern matching, and we have shown that that the model is able to express VNMs commonly found in practice. We have also established a number of intractability and approximation hardness results in various practical VNM settings. These are among the first efforts to settle fundamental problems for virtual network mapping. A few topics are targeted for future work. We are developing practical heuristic algorithms and optimization techniques for VNM. We are also exploring techniques for processing VN requests for different applications, as well as their use in graph pattern matching.

Acknowledgments. Fan and Cao are supported in part by NSFC 61133002, 973 Program 2014CB340302, Shenzhen Peacock Program 1105100030834361, Guangdong Innovative Research Team Program 2011D005, EPSRC EP/J015377/1 and EP/M025268/1, and a Google Faculty Research Award. Ma is supported in part by 973 Program 2014CB340304, NSFC 61322207 and the Fundamental Research Funds for the Central Universities.

References

1. http://www.bigswitch.com/.
2. http://aws.amazon.com/ec2/.
3. http://www.vmware.com/solutions/datacenter/.
4. http://frenzy.ivic.org.cn/.
5. http://homepages.inf.ed.ac.uk/s1165433/papers/vnm-full.pdf.
6. http://www.isi.edu/xbone/.
7. A. Aboulnaga, C. Amza, and K. Salem. Virtualization and databases: state of the art and research challenges. In *EDBT*, 2008.
8. A. Aboulnaga, K. Salem, A. Soror, U. Minhas, P. Kokosielis, and S. Kamath. Deploying database appliances in the cloud. *IEEE Data Eng. Bull*, 32(1):13–20, 2009.
9. D. Andersen. Theoretical approaches to node assignment. *Unpublished Manuscript*, 2002.
10. G. Ausiello. *Complexity and approximation: Combinatorial optimization problems and their approximability properties*. Springer Verlag, 1999.
11. A. C. Bavier, N. Feamster, M. Huang, L. L. Peterson, and J. Rexford. In VINI veritas: realistic and controlled network experimentation. In *SIGCOMM*, 2006.
12. N. Bobroff, A. Kochut, and K. Beaty. Dynamic placement of virtual machines for managing sla violations. In *IM*, 2007.
13. N. Chowdhury and R. Boutaba. A survey of network virtualization. *Computer Networks*, 54(5):862–876, 2010.
14. N. Chowdhury, M. Rahman, and R. Boutaba. Virtual network embedding with coordinated node and link mapping. In *INFOCOM*, 2009.
15. J. Díaz, J. Petit, and M. Serna. A survey of graph layout problems. *CSUR*, 34(3):313–356, 2002.
16. W. Fan, J. Li, S. Ma, N. Tang, Y. Wu, and Y. Wu. Graph pattern matching: From intractable to polynomial time. In *VLDB*, 2010.
17. W. Fan, J. Li, S. Ma, H. Wang, and Y. Wu. Graph homomorphism revisited for graph matching. *VLDB*, 2010.
18. B. Gallagher. Matching structure and semantics: A survey on graph-based pattern matching. *AAAI FS.*, 2006.
19. C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. Bcube: A high performance, server-centric network architecture for modular data centers. In *SIGCOMM*, 2009.
20. J. Lischka and H. Karl. A virtual network mapping algorithm based on subgraph isomorphism detection. In *SIGCOMM workshop VISA*, 2009.
21. W. Reinhardt. Advance reservation of network resources for multimedia applications. In *IWACA*, 1994.
22. R. Ricci, C. Alfeld, and J. Lepreau. A solver for the network testbed mapping problem. *SIGCOMM CCR*, 33:81, 2003.
23. O. Trelles, P. Prins, M. Snir, and R. C. Jansen. Big data, but are we ready? *Nature reviews Genetics*, 12(3), 2011.
24. P. Xiong, Y. Chi, S. Zhu, H. J. Moon, C. Pu, and H. Hacigümüs. Intelligent management of virtualized resources for database systems in cloud environment. In *ICDE*, 2011.
25. M. Yu, Y. Yi, J. Rexford, and M. Chiang. Rethinking virtual network embedding: Substrate support for path splitting and migration. *SIGCOMM CCR*, 38(2):17–29, 2008.
26. B. Zong, R. Raghavendra, M. Srivatsa, X. Yan, A. K. Singh, and K. Lee. Cloud service placement via subgraph matching. In *ICDE*, 2014.