Towards Min-Cost Virtual Infrastructure Embedding

Ruozhou Yu, Guoliang Xue, Xiang Zhang

***Abstract*—Cloud computing has emerged as a prevailing platform for internet service hosting. To best utilize Cloud resources for profit making, Cloud providers rely on intelligent resource allocation algorithms when provisioning the virtualized environments for tenant service hosting. Conventional resource allocation proposals mainly focus on efficient allocation of the computing and storage resources, with little effort on ensuring the network performance of tenant services. To address this issue, a number of recent efforts abstract tenant services in the form of virtual infrastructures for resource allocation. A virtual infrastructure specifies the tenant's demand of both the computing resources for hosting virtual servers, and the network bandwidth for inter-virtual server communications. With the problem of resource allocation for virtual infrastructures being $\mathcal{NP}$-hard in general networks, heuristic algorithms have been proposed for this problem. In this paper, we propose a novel optimization technique, named *sequential rounding*, to tackle the resource allocation problem for virtual infrastructures. The proposed technique extends the rounding technique used for the traditional virtual network embedding problem, while minimizing mapping conflicts introduced by the virtual infrastructure embedding problem. Experiments show that our proposed algorithm outperforms existing algorithms regarding both the acceptance ratio and average embedding cost of virtual requests.**

***Keywords*—Cloud resource allocation, data center virtualization, virtual infrastructure embedding**

1. Introduction

With Cloud computing becoming prevailing as the hosting platform for a large number of internet services, the Cloud resource allocation problem has received extensive attentions from both industry and academia. Cloud infrastructure providers (InPs) adopt intelligent resource allocation algorithms to reduce their operational costs and increase revenue. On the other hand, service providers (SPs) rely on Cloud InPs to provide performance guarantees for their services. These performance guarantees are usually specified by the service level agreements (SLAs) between SPs and InPs. Violation of the SLAs will commonly lead to both degradation to the SPs' services and additional charges to the InPs.

Conventionally, Cloud InPs only offer virtual machine (VM) based service hosting. Resources are allocated in terms of CPU, memory, storage, etc. However, such allocation schemes commonly lead to severe network performance issues, due to the lack of network optimization and bandwidth guarantee. Hence, researchers have proposed a novel service hosting abstraction for Cloud resource allocation, called the *virtual infrastructure* (VI) or the *virtual data center* (VDC) model [1], [2]. A virtual infrastructure consists of both virtual servers (in the form of VMs) and virtual data links that interconnect the virtual servers for communications. By adopting the VI model, SPs can express their demands for network bandwidth explicitly, while InPs are able to conduct network optimization and efficient bandwidth allocation, both to reduce operational costs and to improve quality of the provided services.

However, efficient resource allocation under the VI model is not a trivial task. To ensure performance guarantee for each component (virtual server or link) in a virtual infrastructure, meanwhile to maximize the InP's revenue, the *virtual infrastructure embedding* (VIE) problem seeks a feasible mapping from each virtual component to a hosting substrate component, while minimizing the incurred embedding cost. During the mapping process, not only each virtual server needs to be guaranteed with sufficient computing resources on its substrate host, but also enough bandwidth needs to be reserved in the network to accommodate the high-speed data communications between virtual servers.

The VIE problem (even without considering the virtual link components) can be shown to be $\mathcal{NP}$-hard, by reduction from the *bin-packing* problem [3]. Recent researches have proposed several heuristic solutions to the VIE problem, with different feasibility constraints and optimization objectives [3]–[6].

The VIE problem is similar to the virtual network embedding (VNE) problem in network virtualization [7], which also seeks feasible mappings from virtual networks to substrate networks. One approach for solving the VNE problem is to formulate it as a mixed-integer linear programming (MILP), which generally requires exponential time to solve. To address this issue, in [7] the authors proposed to use relaxation techniques to solve the VNE problem. Efficient VNE heuristics have been developed based on *linear programming (LP) relaxation* and *variable rounding*.

Yet the VIE problem differs from the VNE problem in that it allows consolidation of virtual components, e.g., two virtual servers can be placed on the same substrate host as long as sufficient resources could be reserved. In this case, not only the MILP formulation proposed in [7] is not suitable for the VIE problem, but also applying the same rounding technique would lead to poor performance, due to the possible conflicts of resource allocation between two (potentially) consolidated virtual components. Therefore, in this paper we propose an MILP formulation to model the VIE problem. Based on the formulation, we also propose a novel optimization technique named sequential rounding, which adopts an *iterative* evaluation process to minimize the mapping conflicts between virtual components. Our proposed technique carries out the mapping process by reevaluating the optimal LP relaxation solution after mapping each virtual server.

The main contributions of this paper are as follows:

1) An MILP formulation is proposed for the VIE problem with the objective of cost minimization.

Yu, Xue and Zhang ({ruozhouy, xue, xzhan229}@asu.edu) are all with Arizona State University, Tempe, AZ 85287. All correspondences should be addressed to Guoliang Xue. This research was supported in part by NSF grants 1421685, 1457262 and 1461886. The information reported here does not reflect the position or the policy of the federal government.

2) A novel sequential rounding technique is proposed based on LP relaxation of the proposed MILP and rounding. Based on the analysis about conflicts between before-and-after mapped virtual servers, the proposed technique minimizes the mapping conflicts by iteratively reevaluating the program based on previous decisions.
3) Extensive experiments demonstrate that our proposed algorithm greatly improves the number of feasible solutions found and the quality of solutions over existing works.

The rest of this paper is organized as follows. In Section 2, we discuss recent research efforts related to the VIE problem. In Section 3, we introduce the system model, and describe the VIE problem in detail. In Section 4, we present the MILP formulation for the VIE problem. In Section 5, we first analyze the conflicts between the mappings of (potentially) consolidated virtual servers, and then detail the proposed sequential rounding algorithm based on the analysis. In Section 6, we present the performance evaluation of our proposed algorithm against existing algorithms through simulation experiments. Section 7 concludes this paper.

2. Related Work

A. Virtual Infrastructure/Data Center Embedding

More and more recent researches have focused on resource allocation for virtual data centers. SecondNet [1] first advocates to abstract tenant services in the form of VDCs, in which VMs and an inter-VM traffic matrix jointly define a tenant's resource demand. [2] and [8] extend the VDC abstraction to have homogeneous node/link demands, where [8] presents an optimal solution to the embedding problem in tree-like topologies. PROTEUS [9] and ElasticSwitch [10] also extend the VDC abstraction to incorporate time-varying bandwidth demands and work-conservative bandwidth sharing, respectively. VDC Planner [3] leverages live VM migration to improve virtual request acceptance ratio and reduce cost. Recently, CloudMirror [11] presents a new service abstraction called *Tenant Application Graph* (TAG), which clusters virtual components and allocates network bandwidth accordingly.

Most above works focus on VI embedding with specific assumptions, including all-to-all communication demands [1], [10], specific substrate topologies [2], [8], specific request models [9], [11], or low VM migration costs [3]. Few works have investigated the general VI embedding problem with no assumptions on node/link demands and the topologies. Xu *et al.* [4] studied the general VI embedding problem with survivability against virtual machine failures, and proposed two heuristic algorithms for the problem. Rabbani *et al.* [6] and Zhang *et al.* [5] also proposed heuristic algorithms based on the failure characteristics of the substrate components. These works focus on providing survivability guarantee for the virtual infrastructures rather than minimizing their embedding costs.

B. Virtual Network Embedding

The VIE problem is similar to the traditional VNE problem. Chowdhury *et al.* [7] first proposed a programming-based solutions coordinating the node and link mappings. Yu *et al.* [12] proposed a survivable solution for VNE. A number of works [13]–[17] have studied virtual network embedding through topology-aware approaches. A comprehensive survey for VNE can be found in [18], listing significant proposals for the problem in recent years. However, VIE differs from VNE in that virtual server consolidation is allowed in VIE, *i.e.*, multiple virtual servers can be mapped to the same host; in VNE, each virtual node needs to be mapped to a distinct substrate node. Therefore, most existing algorithms yield poor performance when applied to the VIE problem.

C. VM Management

VM management is another problem related to the VIE problem. A number of early works have studied the VM management problem with network-related constraints. For example, Meng *et al.* [19] studied the traffic-aware VM placement (VMP) problem in data center networks, with the objective of minimizing communication costs. Wang *et al.* [20] studied VM consolidation with dynamic bandwidth demands, and proposed approximation solutions to the consolidation problem. However, the VM management algorithms mostly assume static and single path routing in the underlying network, which greatly limits their application in the VIE problem.

3. System Model and Problem Description

A. Substrate Infrastructure

The substrate infrastructure topology is modeled as an undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{L}, \mathcal{R}, \mathcal{B})$. $\mathcal{N} = \mathcal{N}_H \cup \mathcal{N}_S$ is the substrate node set, where $\mathcal{N}_H$ is the set of substrate hosts (physical servers) that are capable of hosting virtual servers, and $\mathcal{N}_S$ is the set of substrate network switches that cannot host virtual servers. $\mathcal{L}$ is the set of substrate links that interconnect substrate nodes. For substrate host $h \in \mathcal{N}_H$, $\mathcal{R}_h$ denotes the computing resource capacity of the host. In this paper, we consider the CPU capacity as the measure of computing resource of a host. For substrate link $l \in \mathcal{L}$, $\mathcal{B}_l$ denotes the bandwidth capacity of the link.

B. Virtual Infrastructure Request

Similarly, a virtual infrastructure request is denoted by an undirected graph $G = (V, E, R, B)$. V is the set of virtual servers in the virtual infrastructure, and E is the set of virtual links, denoting the communication patterns between virtual servers. Each virtual server $v \in V$ is associated with a demand for substrate computing resource, denoted by R_v. Each virtual link $e \in E$ is associated with a demand for substrate network bandwidth, denoted by B_e.

C. Virtual Infrastructure Embedding

The VIE problem seeks a mapping $\mathcal{M}$ from all virtual components (virtual servers and links) in request G to a subset of the substrate components (substrate servers and paths) in $\mathcal{G}$, such that the total demand of virtual servers mapped to the same substrate host does not exceed its resource capacity, and the total bandwidth reserved on each substrate link does not exceed its bandwidth capacity:

$$\mathcal{M} : (V, E, R, B) \mapsto (\mathcal{N}'_H, P', \mathcal{R}', \mathcal{B}') \quad (1)$$

where $\mathcal{N}'_H \subseteq \mathcal{N}_H$ is a subset of substrate hosts, $P' \subseteq \mathcal{P}(\mathcal{L})$ is a subset of all substrate paths, $\mathcal{R}'$ denotes the computing resource allocated on substrate hosts to virtual servers, and $\mathcal{B}'$ denotes the bandwidth allocated on substrate to virtual links.

Finding a feasible mapping $\mathcal{M}$ involves two stages: *virtual server mapping* (VSM) and *virtual link mapping* (VLM). The VSM stage maps each virtual server in V to a substrate host in $\mathcal{N}_H$. To reduce in-network communications, multiple virtual servers can be mapped to one host, as long as their computing resource demands does not exceed the resource capacity of the substrate host. A VSM scheme is denoted by

$$\mathcal{M}_V : (V,\, R) \mapsto (\mathcal{N}'_H,\, \mathcal{R}') \tag{2}$$

Similarly, The VLM stage maps each virtual link in E to one of the following: 1) an intra-host communication channel (if both of its end points are mapped to the same substrate host), 2) a substrate path (if its end points are separately embedded, and the substrate network adopts *unsplittable* flow routing), or 3) a substrate network flow (if its end points are separately embedded, and the substrate network adopts *splittable* flow routing). A VLM scheme is denoted by

$$\mathcal{M}_E : (E,\, B) \mapsto (P',\, \mathcal{B}') \tag{3}$$

The three choices of mapping a virtual link have different impacts on the embedding of the request. If a virtual link can be mapped to an intra-host communication channel, no network bandwidth is consumed, and thus no communication cost is incurred by embedding the corresponding virtual link. We assume the intra-host communication channel has unlimited bandwidth on each host. If the two end points are mapped to different substrate hosts, network bandwidth needs to be reserved for the data communications between the two virtual servers, incurring cost for bandwidth consumption.

On the other hand, the splittable flow model yields less constrained feasible VLM solutions. This is because a path with enough bandwidth is naturally a network flow with the same flow value, but a flow that satisfies the demand does not necessarily lead to a substrate path with the same bandwidth capacity. In this paper, we assume that the substrate network is capable of splittable flow routing.

D. Objective

We mainly focus on the objective of minimizing the embedding cost incurred by the virtual request. The embedding cost can be calculated in terms of different measurements, such as resource consumption, energy consumption, etc. In this paper, we measure the embedding cost as the weighted sum of overall computing resource and bandwidth consumed for the request, which is formally defined in the next section.

However, the proposed approach in this paper can be easily extended to other optimization objectives, for example, minimizing energy consumption, load balancing, minimizing resource fragmentation, etc. By adjusting the objective function proposed in the next section, various objectives can be optimized during the embedding process.

4. MILP Formulation

In this section, we propose the MILP formulation for the minimum cost VIE problem.

Formally, given substrate $\mathcal{G} = (\mathcal{N}, \mathcal{L}, \mathcal{R}, \mathcal{B})$ and virtual request $G = (V, E, R, B)$, the minimum cost VIE problem is formulated into **VIE-MILP** as follows:

Variables:

- x_v^h: Binary indicator that equals 1 if virtual server $v \in V$ is embedded on substrate host $h \in \mathcal{N}_H$, and 0 otherwise;
- f_e^l: Real-value variable denoting the flow value reserved for virtual link $e \in E$ on substrate link $l \in \mathcal{L}$;

Objective:

$$\text{minimize} \qquad \alpha \sum_{h \in \mathcal{N}_H} \sum_{v \in V} x_v^h \cdot R_v + \beta \sum_{l \in \mathcal{L}} \sum_{e \in E} f_e^l \tag{4}$$

Constraints:

- **Capacity constraints**:

$$\sum_{v \in V} x_v^h \cdot R_v \leqslant \mathcal{R}_h, \;\; \text{for } h \in \mathcal{N}_H \tag{5}$$

$$\sum_{e \in E} \left(f_e^{(m,n)} + f_e^{(n,m)} \right) \leqslant \mathcal{B}_{(m,n)}, \;\; \text{for } (m,n) \in \mathcal{L} \tag{6}$$

- **Flow conservation constraints**:

$$\sum_{(m,h) \in \mathcal{L}} f_{(u,v)}^{(m,h)} - \sum_{(h,n) \in \mathcal{L}} f_{(u,v)}^{(h,n)} = \left(x_v^h - x_u^h\right) \cdot B_{(u,v)}, \quad \text{for } h \in \mathcal{N}_H, (u,v) \in E \tag{7}$$

$$\sum_{(m,s) \in \mathcal{L}} f_{(u,v)}^{(m,s)} - \sum_{(s,n) \in \mathcal{L}} f_{(u,v)}^{(s,n)} = 0, \quad \text{for } s \in \mathcal{N}_S, (u,v) \in E \tag{8}$$

- **Embedding constraints**:

$$\sum_{h \in \mathcal{N}_H} x_v^h = 1, \;\; \text{for } v \in V \tag{9}$$

- **Variable constraints**:

$$x_v^h \in \{0,1\}, \;\; \text{for } h \in \mathcal{N}_H, v \in V \tag{10}$$

$$f_e^l \geq 0, \;\; \text{for } l \in \mathcal{L}, e \in E \tag{11}$$

The above precisely define the minimum cost VIE problem:

- Objective (4) represents cost minimization that considers both host resource (the first term) and link bandwidth (the second term). α and β denote the weights that host and link consumptions contribute to the final cost respectively. Note that the first term is only determined by the virtual request itself. Therefore, it is sufficient to solely minimize the second term.
- Constraints (5) and (6) jointly define the capacity constraints on all substrate hosts and links. In Constraint (6), bidirectional flow values on a substrate link are summed up to account for undirected bandwidth consumption.
- Constraints (7) and (8) guarantee flow conservation of each virtual link at each substrate node. The difference between (7) and (8) is that when computing the flow through a host node, the bandwidth demand of each virtual link is introduced. Through defining the flow of virtual link (u, v) around a host node h by the embedding of the two end-points u and v, the formulation builds the relationship between the VSM and VLM stages.
- Constraint (9) guarantees that each virtual server is embedded onto one and only one substrate host.
- Constraints (10) and (11) specify the ranges of variables.

This formulation is similar to the MILP formulation proposed in [7] for the VNE problem. However, there are several differences. First, the one-to-one virtual node mapping is relaxed in the VIE problem to enable consolidation of virtual

servers. Due to the same reason, the load balancing objective in [7] is modified to purely minimize the embedding cost. Second, as stated in Subsection 3-C, in VIE a virtual link can be mapped onto not only a substrate network flow, but also the intra-host communication channel if the two end-points are consolidated. Moreover, the latter choice is superior to the former one since the intra-host channel has unlimited bandwidth, yet incurs no network cost. Therefore we modify the flow conservation constraints on hosts as Constraint (7) to preferably choose the intra-host channel if the two end points of a virtual link are mapped to the same host. Last but not least, our formulation reduces the number of integral variables by removing the redundant indicators on substrate links. It reduces the time complexity of solving the program directly.

5. VIE through Sequential Rounding

A. LP Relaxation and Rounding

The MILP formulation directly gives an optimal solution to the cost minimization problem of VIE. However, solving an MILP is too expensive even with small cases. A common practice to address the MILP intractability is to relax the integral variables to take continuous values, which is called an *LP relaxation* of the original formulation. The LP relaxation of **VIE-MILP** is given by modifying Constraint (10) to be

$$0 \leq \bar{x}_v^h \leq 1, \text{ for } h \in \mathcal{N}_H, v \in V \tag{12}$$

where $\bar{x}_v^h$ is a continuous variable that substitutes all the appearances of x_v^h in the original formulation. This LP relaxation is named **VIE-LP**.

Physical meaning: By relaxing the integral constraint on the embedding indicator of each virtual server, the relaxation actually allows the entity of a virtual server to be split into multiple pieces, each having a resource demand proportional to the corresponding variable value $\bar{x}_v^h$, and being mapped onto a different host. Moreover, due to Constraint (7), the bandwidth demand of a virtual link (u, v) is also split proportional to the fraction of each piece of virtual node u (or v).

The relaxed formulation can be solved in polynomial time using standard optimization techniques. However, the optimal solution of the relaxed formulation may not be a feasible one to the original problem, due to the possible splitting of virtual server mappings. To address this issue, the technique of LP rounding could be applied [7], which enforces the integral constraints of VSM based on the variable assignments of the optimal LP. Each virtual server is mapped to either the substrate host that holds the largest fraction of the virtual server (with the largest value $\bar{x}_v^h$ for $\forall h \in \mathcal{N}_H$ in this case) (deterministic rounding), or every substrate host with probability equal to the corresponding $\bar{x}_v^h$ values (randomized rounding). In this manner, each virtual server is mapped to the substrate host that potentially leads to the optimal LP.

However, the rounding algorithm proposed in [7] is designed for the VNE problem. When the problem is generalized to the VIE problem where virtual server consolidation is enabled, the same intuition may lead to *conflicts* between before-and-after mapping decisions, resulting in degraded performance of the algorithm. Assuming two virtual servers u and v with bandwidth demand $B_{(u,v)}$ are to be mapped, two conflicting scenarios may happen during the embedding:

1) For host X, assume $\bar{x}_u^X = \max\{\bar{x}_u^h \,|\, h \in \mathcal{N}_H\} < 1$ and $\bar{x}_v^X = \max\{\bar{x}_v^h \,|\, h \in \mathcal{N}_H\} < 1$. Therefore u and v are mapped to the same host X after deterministic rounding. However, if $\mathcal{R}_X < R_u + R_v$, the embedding will immediately fail as the resource capacity on X is insufficient to host both u and v.
2) For hosts X and Y, assume $\bar{x}_u^X = \bar{x}_v^X$ and $\bar{x}_u^Y = \bar{x}_v^Y$ are the largest two mapping variables of u and v respectively. According to Objective (4) and Constraint (7), the flow of virtual link (u, v) on either host X or host Y is minimized to 0. Assume $\max\{R_u, R_v\} < \mathcal{R}_X < R_u + R_v$ and $\max\{R_u, R_v\} < \mathcal{R}_Y < R_u + R_v$, and therefore without loss of generality u is mapped to X and v is mapped to Y with high probability after randomized rounding. However, if $\mathcal{B}_{(X,Y)} < B_{(u,v)}$ ($\mathcal{B}_{(X,Y)}$ is the maximum bisectional bandwidth between X and Y), the embedding fails as the bandwidth between X and Y is insufficient.

Both scenarios fail because the later-made mapping decisions have conflicts with the decisions made before them. For example, in Case 1, after embedding the first node (assume it is u), the LP solution should be reconsidered, since node u consumes more resource than assigned on the host X. In this case node v should be mapped to a different host that has sufficient resource capacity, rather than the same host X although it holds the largest piece of v in the LP solution. Similarly, in Case 2, after embedding the first node fully on one host (assume it is $u \mapsto X$), the LP solution should be reconsidered to account for the newly incurred network bandwidth of virtual link (u, v), and choose some substrate host other than Y for v.

B. LP Sequential Rounding

To tackle the aforementioned conflicts, we propose a novel technique named *sequential rounding*.

The intuitive idea of sequential rounding is to minimize the conflicts between before-and-after VSM decisions. To achieve this, the optimal LP solution is reevaluated after each mapping decision is made, taking into consideration the mappings of previously determined virtual servers. The new optimal LP solution will consist of all already fixed virtual server mappings in previous rounds, and the optimal assignments of the rest virtual servers in the relaxed form. This process of (re)evaluating and rounding will continue until all virtual servers are feasibly mapped, after which the VLM stage can be solved optimally as the *Multi-Commodity Flow* (MCF) problem, given the fixed VSM scheme. By continuously reevaluating the LP solution based on previous decisions, it trades only linear time with much improved embedding performance.

The detailed VIE algorithm based on sequential rounding, named **VIE-SR**, is proposed in Algorithm 1.

Following the above intuition, the **VIE-SR** algorithm makes the mapping decision of each virtual server by solving an updated LP relaxation based on previous decisions. The initial LP relaxation is built as proposed in Section 4 and Subsection 5-A in Line 2. In the iterative loop in Line 3–11, first the LP formulation is solved to obtain the optimal solution with possible fractional variable assignments for VSM. Then, the pair of (unmapped) virtual server and substrate host with the maximum value of $\bar{x}_v^h$ over all combinations is selected and

Algorithm 1: VIE-SR $(\mathcal{G}, G)$

1 Initialize $\mathcal{M} \leftarrow \emptyset$;
2 Build the LP relaxation formulation $\mathcal{LP}$;
3 **while** V *is not empty* **do**
4 Solve $\mathcal{LP}$ using standard LP technique;
5 **if** $\mathcal{LP}$ *is infeasible* **then**
6 **return** *Reject*.
7 **end**
8 $h^*, v^* \leftarrow \arg\max_{h,v}\{\bar{x}_v^h \mid v \in V, h \in \mathcal{N}_H, R_v \leq \mathcal{R}_h\}$;
9 $\mathcal{M}_N(v^*) \leftarrow h^*$, $\mathcal{R}_{h^*} \leftarrow \mathcal{R}_{h^*} - R_{v^*}$, $V \leftarrow V \setminus \{v^*\}$;
10 Update $\mathcal{LP}$ with $\bar{x}_{v^*}^{h^*} = 1$ and $\bar{x}_{v^*}^{h'} = 0$ for $h' \neq h^*$;
11 **end**
12 Solve $\mathcal{LP}$ using standard LP technique;
13 **if** $\mathcal{LP}$ *is infeasible* **then**
14 **return** *Reject*.
15 **end**
16 Derive link mappings $\mathcal{M}_E$ based on LP variables f_e^ls;
17 **return** $\mathcal{M}$.

mapped in Line 8 and 9. In the mean time, the LP formulation $\mathcal{LP}$ is also updated with the new mapping $v^* \mapsto h^*$ in Line 10, so that later iterations will take it into consideration.

After all the virtual servers are mapped, the LP formulation is solved again with all fixed virtual server mappings, in order to obtain the optimal VLM scheme (Line 12–16). Note that when all VSM indicators are assigned in the MILP/LP formulation, the formulation turns into exactly the LP formulation for the MCF problem. If all steps output feasible solutions, the algorithm returns with the mapping $\mathcal{M}$; if at any step no feasible solution is found, the algorithm returns "Reject" to indicate an embedding failure.

6. Performance Evaluation

In this section, we evaluate the performance of our proposed algorithm. In the evaluation, we mainly focus on measuring the number and quality of solutions found by each algorithm. We also compare our proposed algorithm with existing algorithms modified to solve the same problem.

A. Experiment Setup

Experiments were conducted on random graphs. By default, the substrate topology was generated with 80 nodes: 30 routers and 50 hosts. To simulate real-world networks, each host was only connected through its edge router, which was randomly chosen. Routers, on the other hand, were connected as a mesh network with 0.5 probability of connection between each pair. The resource/bandwidth capacity on each substrate host/link was uniformly generated between 50 and 100. Unless otherwise specified, each virtual request was generated with 6 virtual servers, connected with probability of 0.5 between each pair. The resource/bandwidth demand on each virtual server/link was uniformly generated between 0 and 50.

A set of algorithms were compared including our proposed algorithm. Notations of these algorithms are listed in Table 1. To evaluate among algorithms, we used the same substrate state information and the same virtual infrastructure request, and compared the solution for applying each algorithm. For this reason, the host and network resources were not reserved after the embedding of each request.

All algorithms were implemented in C++. The Gurobi Optimizer [21] was used to solve the intermediate LP formulations. All experiments were conducted on a Linux PC with 3.4GHz Quad-Core CPU and 16GB memory. Each single case of experiment was repeated for 50 times with the same experiment settings (number of nodes in substrate and virtual topologies and average demands) but differently generated substrate infrastructure and virtual request instances.

TABLE 1: Compared Algorithms

Algorithm	Description
VIE-SR	The proposed sequential rounding algorithm in this paper, as shown in Algorithm 1.
D-ViNE	Deterministic rounding algorithm in [7], with the LP relaxation modified as in Section 4 of this paper.
R-ViNE	Randomized rounding algorithm in [7], with the LP relaxation modified as in Section 4 of this paper.

B. Performance Metrics

The performance of an algorithm is measured in two metrics. The static *acceptance ratio* defines the percentage of repeated experiments with the same experiment settings that succeed, which reflects the capability of an algorithm to find *feasible* solutions. The *average embedding cost* is measured by the objective function defined in Eq. (4) in Section 4, which reflects the capability of an algorithm to find *good* solutions. Note that when some algorithms fail to find feasible solutions while some other succeeds, only the costs for the virtual requests for which every algorithm finds feasible solutions are accounted. The weights in Eq. (4) are set to $\alpha = \beta = 1$.

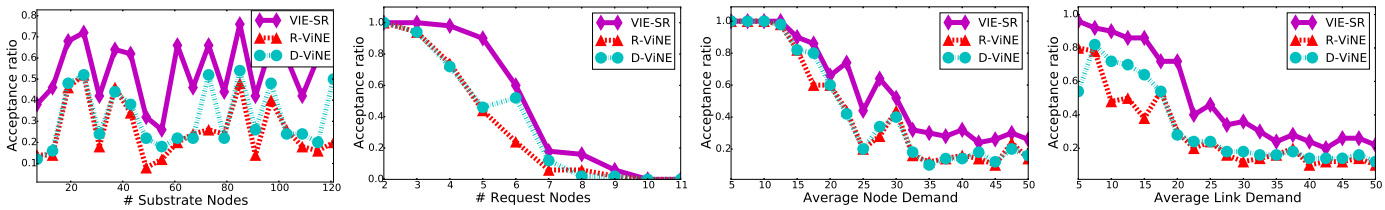
C. Evaluation Results

Figs. 1 and 2 show the acceptance ratio and the average embedding cost respectively, under varying experiment settings.

Fig. 1 shows that **VIE-SR** yields higher acceptance ratio over existing deterministic and randomized rounding algorithms, with varying substrate/request sizes and node/link demands. With increasing number of request nodes or average node/link demands, although all algorithms decrease in the ratio of accepted requests, **VIE-SR** has a much more graceful degradation in acceptance ratio, and can accept more requests under the same situation. **D-ViNE** and **R-ViNE** both rounds with the same relaxed program. Hence their performance is similar, with slight edge on deterministic algorithm **D-ViNE**.

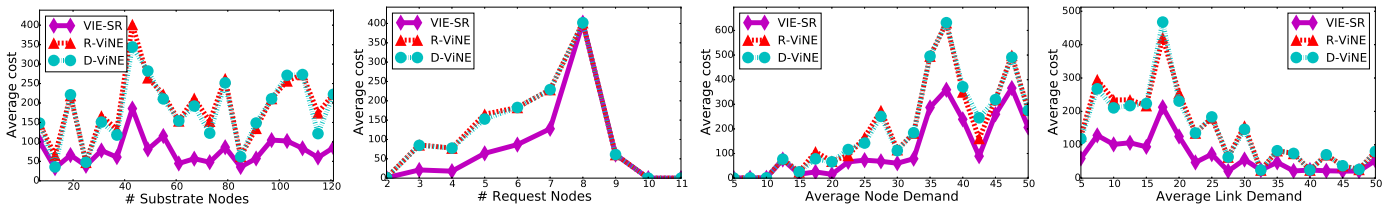
Fig. 2 further shows that even for requests that are accepted by all algorithms, the embedding costs incurred by **VIE-SR** are usually lower than the costs incurred by existing algorithms. In Figs. 2b, 2c and 2d, the cost generally first increases with number of request nodes or average node/link demands due to the increase in the overall resource/bandwidth demands for accepted requests, and then decreases, due to the decrease in acceptance ratio. For example, with more than 8 virtual nodes requested, almost no solution can be found by all three algorithms, and thus their total costs drop drastically. For requests with large demands or number of request nodes, normally not all algorithms can find a feasible solution. Hence their embedding costs are not accounted in the average, which leads to the irregular curves in the cost figures.

In summary, **VIE-SR** has a great performance gain over the existing solutions, regarding both request acceptance ratio



(a) Acceptance ratio vs. substrate size (b) Acceptance ratio vs. request size (c) Acceptance ratio vs. node demand (d) Acceptance ratio vs. link demand

Fig. 1: Acceptance ratio with varying substrate and substrate sizes, and mean request node and link demands, respectively.



(a) Embedding cost vs. substrate size (b) Embedding cost vs. request size (c) Embedding cost vs. node demand (d) Embedding cost vs. link demand

Fig. 2: Average embedding cost with varying substrate and substrate sizes, and mean request node and link demands, respectively.

and embedding cost. While minimizing the cost does not necessarily yield maximized revenue due to the online nature of the problem, it can be expected that the revenue would increase due to more accommodated requests and the reduced cost. Moreover, objectives other than minimizing total resource consumption can also be enforced in the same algorithm, with minimum modification to the programming formulation.

7. Conclusions

In this paper, we studied a crucial problem in Cloud resource management: efficient resource allocation for virtual infrastructures, or namely the minimum cost virtual infrastructure embedding problem. The detailed problem model was proposed, followed by a mixed-integer linear programming formulation for the problem. Based on the MILP formulation, the novel sequential rounding technique was proposed to tackle the new challenge of mapping conflicts introduced by the VIE problem. Simulation experiments showed that our proposed algorithm greatly improves the number and quality of feasible solutions over existing algorithms regarding both the number of requests accepted and their embedding costs.

References

[1] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang. SecondNet: A Data Center Network Virtualization Architecture with Bandwidth Guarantees. In *ACM CoNEXT*, 2010.

[2] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. Towards predictable datacenter networks. In *ACM SIGCOMM*, pp. 242–253, 2011.

[3] M. F. Zhani, Q. Zhang, G. Simon, and R. Boutaba. VDC Planner: Dynamic migration-aware virtual data center embedding for clouds. In *IEEE IM*, pp. 18–25, 2013.

[4] J. Xu, J. Tang, K. Kwiat, W. Zhang, and G. Xue. Survivable Virtual Infrastructure Mapping in Virtualized Data Centers. In *IEEE CLOUD*, pp. 196–203, 2012.

[5] Q. Zhang, M. F. Zhani, M. Jabri, and R. Boutaba. Venice: Reliable virtual data center embedding in clouds. In *IEEE INFOCOM*, pp. 289–297, 2014.

[6] M. G. Rabbani, M. F. Zhani, and R. Boutaba. On Achieving High Survivability in Virtualized Data Centers. *IEICE Trans. Commun.*, E97-B(1):10–18, 2014.

[7] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba. Virtual Network Embedding with Coordinated Node and Link Mapping. In *IEEE INFOCOM*, pp. 783–791, 2009.

[8] J. Zhu, D. Li, J. Wu, H. Liu, Y. Zhang, and J. Zhang. Towards bandwidth guarantee in multi-tenancy cloud computing networks. In *IEEE ICNP*, pp. 1–10, 2012.

[9] D. Xie, N. Ding, Y. C. Hu, and R. Kompella. The only constant is change: incorporating time-varying network reservations in data centers. In *ACM SIGCOMM*, pp. 199–210, 2012.

[10] L. Popa, P. Yalagandula, S. Banerjee, J. C. Mogul, Y. Turner, and J. R. Santos. ElasticSwitch: practical work-conserving bandwidth guarantees for cloud computing. In *ACM SIGCOMM*, pp. 351–362, 2013.

[11] J. Lee, Y. Turner, M. Lee, L. Popa, S. Banerjee, J.-M. Kang, and P. Sharma. Application-driven bandwidth guarantees in datacenters. In *ACM SIGCOMM*, pp. 467–478, 2014.

[12] H. Yu, C. Qiao, J. Wang, L. Li, V. Anand, and B. Wu. Regional Failure-Resilient Virtual Infrastructure Mapping in a Federated Computing and Networking System. *J. Opt. Commun. Netw.*, 6(11):997–1007, 2014.

[13] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang. Virtual network embedding through topology-aware node ranking. *ACM SIGCOMM CCR*, 41(2):38–47, 2011.

[14] S. Zhang, Z. Qian, J. Wu, and S. Lu. An Opportunistic Resource Sharing and Topology-Aware mapping framework for virtual networks. In *IEEE INFOCOM*, pp. 2408–2416, 2012.

[15] Z. Wang, Y. Han, T. Lin, H. Tang, and S. Ci. Virtual network embedding by exploiting topological information. In *IEEE GLOBECOM*, pp. 2603–2608, 2012.

[16] J. Wang and J. Liao. Topology-aware virtual network embedding through bayesian network analysis. In *IEEE GLOBECOM*, pp. 2621–2627, 2012.

[17] L. Gong, Y. Wen, Z. Zhu, and T. Lee. Toward profit-seeking virtual network embedding algorithm via global resource capacity. In *IEEE INFOCOM*, pp. 1–9, 2014.

[18] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach. Virtual Network Embedding: A Survey. *IEEE Commun. Surv. Tutorials*, 15(4):1888–1906, 2013.

[19] X. Meng, V. Pappas, and L. Zhang. Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement. In *IEEE INFOCOM*, pp. 1–9, 2010.

[20] M. Wang, X. Meng, and L. Zhang. Consolidating virtual machines with dynamic bandwidth demand in data centers. In *IEEE INFOCOM*, pp. 71–75, 2011.

[21] Gurobi Optimizer. http://www.gurobi.com/products/gurobi-optimizer.