Network-Aware Embedding of Virtual Machine Clusters onto Federated Cloud Infrastructure

Atakan Aral*, Tolga Ovatman

Department of Computer Engineering, Istanbul Technical University, Istanbul, Turkey

Abstract

Federated clouds are continuously developing as the demands of cloud users get more complicated. Contemporary cloud management technologies like OpenStack [1] and OpenNebula [2] allow users to define network topologies among virtual machines that are requested. Therefore, federated clouds currently face the challenge of network topology mapping in addition to conventional resource allocation problems. In this paper, topology based mapping of virtual machine clusters onto the federated cloud infrastructures is studied. A novel algorithm is presented to perform the mapping operation that work towards minimizing network latency and optimizing bandwidth utilization. To realize and evaluate the algorithm, a widely used cloud simulation environment, CloudSim [3], is extended to support several additional capabilities in network and cost modeling. Evaluation is performed by comparing the proposed algorithm to a number of conventional heuristics such as least latency first and round-robin. Results under different request characteristics indicate that the proposed algorithm performs significantly better than the compared conventional approaches regarding various QoS parameters such as inter-cloud latency and throughput.

Keywords: Cloud computing, Federated cloud, Infrastructure as a Service, VM cluster embedding, Resource allocation, Subgraph isomorphism

*Corresponding author. Tel.: +90 212 285 6704.

Email addresses: aralat@itu.edu.tr (Atakan Aral), ovatman@itu.edu.tr (Tolga Ovatman)

1. Introduction

Magnitude of the digital data being generated and the speed at which it is aggregating in cloud is enormous. In the not so distant future, even the largest Infrastructure as a Service (IaaS) providers may run into a difficulty in scalability as a result of this enormous increase in cloud service usage. Moreover, cloud users access the data from all around the world which makes it increasingly hard to provide a globally consistent Quality of Service (QoS). Federated cloud [4, 5] is motivated by these dangers and defined as the mechanisms, policies and technologies to coordinate and unite cloud data centers even if they are managed by different vendors. As distinct from multi-cloud where multiple independent clouds are utilized by an application uninformedly, cloud providers voluntarily collaborate in the federated cloud scenario [6].

Federated clouds allow vendors to dispatch Virtual Machine (VM) requests to the other members of the federation, delivering the infinite scalability promise of cloud computing [4]. This improves the QoS by giving cloud vendors the ability to cope with demand peaks as well as to provide complete geographical coverage. Additionally, such an interoperability at the infrastructure level sets cloud users free of vendor lock-in and allows private data center owners to easily hybridize. Finally and more importantly for our work, with federated cloud it is possible to scale VM clusters across multiple vendor clouds [5]. Here, a VM cluster is a group of collaborating VMs that constitute a cloud service. It is a common practice to isolate different components of a service (e.g. storage, application logic, user interface) using distinct VMs that communicate among themselves.

From the point of a cloud based service provider (and an IaaS user), deployment of cooperating VMs on different clouds paves the way for the following advantages.

Availability and Disaster Recovery The effect of a failure or low QoS in a cloud vendor can be easily compensated with minimal damage to the overall service.

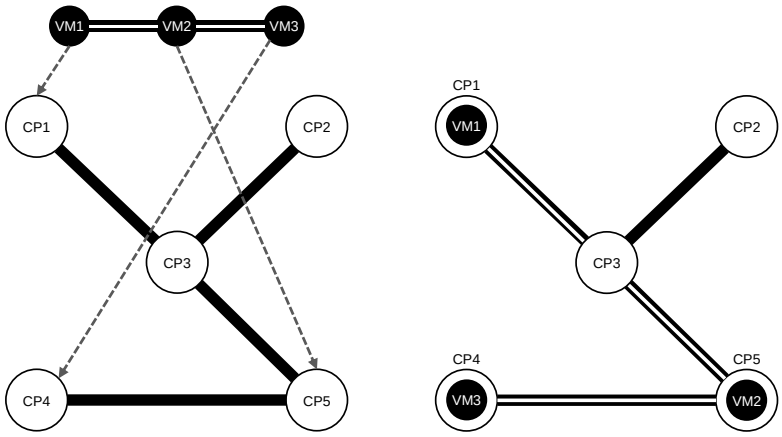
Geographical Coverage Geographically distributed user base of the service can be covered with a high QoS.

No Vendor Lock-in VMs can be migrated easily and quickly between vendors in case of any dissatisfaction.

Cost Reduction Different pricing policies of the vendors can be exploited to reduce infrastructure cost.

However, distributed placement of VMs onto a federated cloud infrastructure also presents new problems that need to be addressed. One of the most significant of these problems is finding an efficient mapping between the physical topology and the user requests in the form of virtual topologies [7]. Here, virtual topology defines the bandwidth requirements for data flows between VMs in the same cluster. When a service provider requests a VM cluster, it characterizes VM capacities as well as the amount of data that will be transferred between each VM pair in terms of bandwidth. On the other hand, physical topology defines the available dedicated network connections between cloud providers with their bandwidth capacities and latencies. Not all cloud provider pairs in the federation may have direct dedicated connections and not all VM pairs in a cluster need to communicate, thus neither of the topologies are complete graphs in general. When adjacent VMs are mapped to nonadjacent clouds, the connection has to be multi-hop.

Figure 1 visualizes the mapping and deployment of a single VM cluster of three VMs onto a federation of 5 cloud providers (CP). Here, physical topology is represented with white circles (clouds) and thick lines (inter-cloud network connections) while virtual topology is represented with black circles (VMs) and double lines (data flows). According to the requested virtual topology, data will flow between pairs VM1 – VM2 and VM2 – VM3 but not VM1 – VM3. Figure 1(a) demonstrates an example mapping between VMs and clouds. Different mappings can be generated via optimization algorithms with different objective functions, however the mapping relation must satisfy the function property (each



(a) Mapping (b) Deployment

Figure 1: VM Cluster Embedding [8].

VM must be mapped to exactly one output). In Figure 1(b) VMs are dispatched to clouds according to the mapping in Figure 1(a) and deployed there. During the execution, data transfer between VM2 and VM3 will be direct, while it will be through CP3 for VM1 and VM2. In a real world scenario with non-trivial number of VM clusters, multiple VMs belonging to different clusters would be hosted at each cloud.

VM cluster embedding (VMCE) problem deals with finding a mapping between inter-connected VMs and clouds, as exemplified by Figure 1(a). The problem is not trivial due to multiple constraints and objectives present [9]. First of all, clouds have limited and heterogeneous capacities in terms of CPU, memory and storage. Similarly, network connections have varied latencies and bandwidth capacities. VMs of different sizes should be placed on clouds respecting such limits and making an efficient use of the resources to increase utilization. A similar problem is referred as Virtual Network Embedding (VNE) in the literature [7, 10, 11]. Our definition of VMCE diverges from VNE as it also involves constraints and requirements for nodes (cloud/VM) in addition to the edges (network). A mathematical definition of the VMCE problem is provided in subsection 3.1.2.

Network plays a key role in the performance of distributed cloud services. Hence, communicating VMs should be placed on clouds that have low latency inter se for high QoS. Another factor is the latency between the user base and selected clouds. In the case of provisional applications such as scientific calculations or MapReduce [12] jobs, high latency also extends the execution time and accordingly increases resource costs. Better latency optimization is vital for distributed, soft real-time services and applications (e.g. video streaming, online gaming) to run on federated cloud. Cloud computing may find a new area of application in real-time software provided that the network related challenges are overcome [13].

Major contributions of this paper can be listed as follows.

- A novel VMCE algorithm for federated cloud, Topology Based Mapping (TBM), is proposed. TBM employs a graph theoretical approach in combination with greedy heuristics. Main objectives of the TBM are to reduce network latency and optimize bandwidth utilization.

- RalloCloud, a framework for the realistic simulation of resource allocation in federated cloud as an extension to CloudSim [3], is presented. It includes topology, network and cost modeling as well as several performance criteria.

- Evaluation of the TBM algorithm as well as baseline heuristics in terms of latency, execution time, throughput, cost, rejection rate, etc. is performed.

TBM algorithm mainly focuses on the bandwidth and latency that is (1) within the VM cluster, and (2) between the VM cluster and the intermediate cloud user who submits it (e.g. a cloud-based service provider or a scientist running a high performance job). Optimization of the latency and bandwidth between the VM cluster and the geographically distributed end users is beyond the scope of this work. Reader may refer to works on replica placement (e.g. [14, 15]) for this kind of optimization.

A preliminary version of the TBM algorithm is published as a work-in-progress paper [8]. Here, we present the complete version of the algorithm and extended evaluation results. Improvements to the TBM algorithm includes validity consideration (see subsection 3.1.2) while the extended evaluation includes new evaluation parameters and heterogeneous infrastructure capacities (see subsection 5.1). Consequently, TBM significantly outperforms its preliminary version in [8]. Moreover, RalloCloud framework is introduced for the first time in this article.

The rest of this paper is organized as follows. Section 2 summarizes the relevant literature. Respectively in Sections 3 and 4, suggested framework (RalloCloud) and algorithm (TBM) are introduced. Section 5 presents evaluation results and their discussion. We conclude the paper in Section 6.

2. Related Work

2.1. Graph and Subgraph Equivalence

Two graphs are called isomorphic if a bijective function that pairs vertices of one graph to the vertices of the other graph with edge preserving property can be defined. Edge preserving property states that two adjacent vertices of one graph can be paired to two vertices in the other if and only if they are adjacent as well. Instead, if a subdivision of a graph is isomorphic to a subdivision of another graph, then these graphs are called homeomorphic. A subdivision of a graph can be generated by replacing an edge with a new vertex which is adjacent to the endpoint vertices of the original edge with two new edges.

In subgraph isomorphism / homeomorphism problems, given two graphs, the objective is to find a subgraph of the larger one that is isomorphic / homeomorphic to the smaller one. Both these problems are shown to be NP-complete [16] and can be solved via evaluating all possible matchings. In order to reduce the search space, several filtering methods are proposed in the literature [17].

2.2. Cloud Interoperability

Ability to dispatch VMs or data between cloud providers is referred as cloud interoperability and it is considered an important challenge in cloud computing [18, 19]. Establishing interoperability is critical for both elimination of vendor lock-in and realization of federated cloud. Thus, several promising approaches for interoperating clouds are present in the literature [20, 21].

An effective approach for interoperability is defining open standards. A comprehensive list of standardization efforts is presented in [22]. Since a universally accepted standard is not available currently, there are also efforts to provide interoperability at the user-level. In Multi-Cloud model, clouds do not directly communicate and the user is responsible for providing an adaptor for their interoperation [23, 24]. A cloud broker who is responsible to negotiate with cloud providers on behalf of the user, may also act as a cloud aggregator and provide a unified interface to nonstandard vendor APIs [25, 26]. In addition to interoperability, cloud brokers can also be employed to ensure QoS with reduced cost in federated cloud [27].

2.3. Cloud Benchmarking and Monitoring

Efficient monitoring and benchmarking of cloud providers is closely linked with the quality of the resource management for federated cloud. This is because VM scheduling, allocation and placement algorithms require up-to-date performance and utilization information to cope with quickly changing conditions and make optimum decisions [28]. Once these measures are collected within a cloud data center, they should also be disseminated through the federation in a secure and efficient way so that the cloud brokers work in harmony. A large number of research studies and tools have been introduced so far that monitor cloud systems [28] and benchmark their services [29] as well as their performance [30].

2.4. VM Cluster Embedding

Several studies have been conducted for the efficient embedding of virtual topologies onto federated cloud infrastructure in the past years [7, 9, 23, 26, 31–37]. Additionally, there exist studies focused on similar embedding problems

onto other systems that are either non-cloud [11, 38–43] or non-federated cloud [44–48]. Table 1 summarizes some selected properties of these studies. Description of each column of the table is given below.

Cloud Whether the placement is made onto a cloud infrastructure with on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service according to the definition in [49].

Federation Whether the VMs or tasks are placed onto a federated, distributed and networked infrastructure (e.g. Federated cloud, Multi-cloud, Inter-cloud, etc.).

Embedding Whether a network or VM cluster embedding is carried out as described in Section 1 and Figure 1. Otherwise, VMs or tasks are individually treated and placed.

Entity The term used in the study to indicate entities that are placed (Virtual Machines (VMs), Virtual Nodes (VNs), Services, Tasks, Jobs or Workflow Engines (WEs)).

Simultaneous Whether the algorithm maps nodes and links simultaneously. Otherwise, there are two phases of the algorithm for mapping them independently and/or sequentially.

Network-aware Whether the algorithm considers network factor during the placement. This includes the improvement of latency, bandwidth utilization, hop count, etc.

For the sake of brevity, we provide comments on a subset of the studies in Table 1. Authors of the paper [38] suggest a mixed integer programming solution with relaxed integer constraints for the network embedding problem. Their approach increases the coordination between node mapping and link mapping phases with the aim of increasing acceptance rate and revenue.

Use of subgraph isomorphism is introduced in [39] for embedding virtual topologies. Later, [40] suggested mapping only the critical nodes via subgraph isomorphism and the rest with heuristics.

Studies introduced so far are designed to embed virtual networks onto single-site substrate networks such as data centers. In that scenario, network latency and bandwidth are not as critical as geographically distributed environments, e.g. federated cloud. Moreover, location of the user base emerges as an important factor for the performance of clusters in the latter case.

In [41] (and in the other works that are discussed henceforth) however, authors consider a similar distributed resource allocation problem to the one we aim to solve. They suggest an exact (integer programing) solution to the network embedding problem. They also prove that grouping incoming requests during a period and embedding them simultaneously yields higher acceptance ratio and lower cost in comparison to the individual handling of each request. Authors of the article [26] employ cloud brokers both to optimize placement and to act as a uniform interface for developers. The method for optimized placement is also integer programming. Even though integer programming can produce optimal results in a centralized case, using a heuristic can simplify the amount of overhead induced by trying to solve large optimization problems.

Both data center selection and VM placement to the physical resources of the selected data centers are considered in [32]. Suggested 2-approximation algorithm first finds a subgraph of the complete data center topology with the smallest diameter. Then, VMs requested by the user are partitioned in such a way that each partition fits to a data center in the subgraph and the inter-data center traffic is minimum.

Virtual topology graph is partitioned into connected components via a k-cut algorithm in [37]. A new graph, the nodes of which represent the connected components and the edges of which aggregate the inter-component links, is generated. Later, this graph is matched to an isomorphic subgraph of the physical topology graph and the VMs in each partition is submitted to the cloud matched to that component. In this work, the mapping between VM and clouds is not

	Cloud	Federation	Embedding	Entity	Simult.	NW-aware
[31]	✓	✓	✓	VMs		
[32]	✓	✓	✓	VMs	✓	✓
[33]	✓	✓	✓	Services	✓	
[34]	✓	✓	✓	Tasks		
[35]	✓	✓	✓	VMs	✓	✓
[36]	✓	✓	✓	VNs		✓
[23]	✓	✓	✓	VMs	✓	
[9]	✓	✓	✓	VMs		✓
[7]	✓	✓	✓	VNs		✓
[26]	✓	✓	✓	VMs	✓	✓
[37]	✓	✓	✓	VNs	✓	✓
[38]			✓	VNs		
[39]			✓	VNs	✓	✓
[40]			✓	VMs	✓	✓
[11]			✓	VNs		✓
[41]		✓	✓	VNs	✓	✓
[42]		✓	✓	VMs	✓	✓
[43]		✓	✓	VNs	✓	✓
[44]	✓		✓	VMs	✓	✓
[45]	✓		✓	VNs		✓
[46]	✓		✓	Tasks	✓	✓
[47]	✓		✓	VNs		✓
[48]	✓		✓	VNs		✓
[50]	✓	✓		VMs	✓	✓
[51]	✓	✓		Jobs		
[52]	✓	✓		WEs	✓	✓

Table 1: Summary of the related literature.

injective (one-to-one). Since the objectives are load balancing and cost reduction, if the utilization of the clouds is low enough, all VMs are mapped to the same cloud without partitioning. Hence the mapping fails to gain the benefit of inter-cloud deployment.

Similarly, an iterated local search based graph partitioning and integer programming based embedding with an objective function that minimizes the cost and number of hops is proposed in [36]. Recently, the same authors suggested a semantic based greedy node mapping algorithm for federated virtual infrastructures [7]. The algorithm defines a upper limit for the number of hops between nodes to increase QoS. However, actual latency incurred by the distribution are not evaluated in neither of the studies.

Other approximation techniques to solve the VMCE problem or similar problems include artificial immune system [11] and markov random walk [45].

TBM algorithm mainly differs from the existing approaches in the following two ways.

- To the best of our knowledge, only one study [37] addresses the use of subgraph isomorphism for resource management in distributed infrastructures, e.g. grids and clouds. Our work is the first attempt to match the exact virtual network request to a subgraph of the physical network, and consequently produce a bijective matching and a fully distributed mapping.

- Although most of the existing VMCE or VNE algorithms are network-aware, none of them explicitly measure inter-VM latency in their evaluation. Only a few studies [7, 9, 32, 47] employ hop count metric which can be deceiving in the case of heterogeneous network infrastructure. We measure both inter-cloud and cloud-to-user latency and show that TBM decreases both.

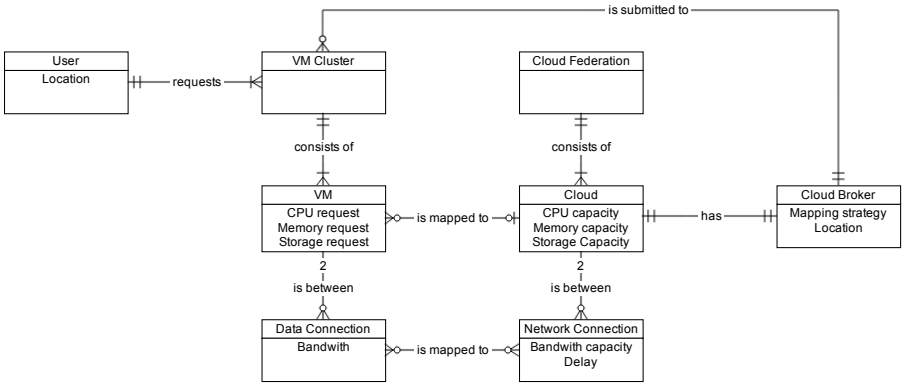


Figure 2: Partial Entity-Relationship Model of RalloCloud.

3. RalloCloud Framework

RalloCloud [53] is a framework for modeling and simulating resource allocation in the federated cloud environment. It is developed on top of the popular cloud simulation toolkit, CloudSim [3]. The rest of this section explains how federated cloud infrastructure, VM clusters and resource allocation problem are modeled as well as the performance criteria to evaluate the allocation.

3.1. Problem Modeling

3.1.1. Entities

Entity-Relationship model of the RalloCloud is given in Figure 2. A **cloud data center** (cloud for short) is defined with its resource capacities (CPU, memory, storage) and its geographical location. **Physical topology** is the network connections between clouds which include bandwidth capacity and latency. Each cloud is attached to a **cloud broker** that is responsible for receiving user requests in the form of VM clusters and allocating resources for them. It is possible for a broker to dispatch VMs to other brokers. This federated environment allows us to model realistic cloud brokerage scenarios.

A **VM cluster**, on the other hand, is composed of **VMs**, a **virtual topology** and the geographical location of requesting user. Each VM is defined with its size and resource requirements (CPU, memory, storage) while the virtual

topology is the collection of required bandwidths between VMs. A VM cluster starts executing when each VM in that cluster is deployed to a cloud and it is terminated when each VM in that cluster completes its execution.

Both the VM cluster and the cloud federation are represented with weighted undirected graphs; $G_C = (V_C, E_C, A_C^V, A_C^E)$ and $G_F = (V_F, E_F, A_F^V, A_F^E)$, respectively. V is the set of vertices and E is set of edges while A^V is the attributes of the vertices and A^E is the attributes of the edges. We use subscripts C and F to distinguish cluster and federation. A^V and A^E consist of the attributes that are as explained above, for instance, A_F^E defines the bandwidth capacity and latency of each network connection.

3.1.2. VM Cluster Embedding Problem

Embedding a user's VM cluster request onto the cloud federation involves mapping each vertex $v \in V_C$ to exactly one vertex $v' \in V_F$ and mapping each edge $e \in E_C$ to exactly one set of edges $E' \subseteq E_F$. Three conditions must hold for these mappings to be considered a valid embedding:

1. Remaining resource capacities of all v' and $e' \in E'$ are greater than or equal to the sum of required resources by all v and e which are mapped to v' and E'.

2. For each e, mapped E' forms a path between the two distinct vertices v'_1 and v'_2 to which endpoint vertices of e (v_1, v_2) are mapped.

3. If multiple vertices $v_1, \cdots, v_n \in V_C$ are mapped to the same v', then any e which has both endpoint vertices in $v_1, \cdots, v_n$ is mapped to $E' = \varnothing$.

In our model, it is possible to map multiple VMs to a single cloud as long as the resource capacity is enough. Similarly, a network connection can be utilized by multiple data connections and multiple VM clusters can be embedded onto the cloud federation.

3.1.3. Network

Network connections in the cloud federation are represented with the bandwidth capacity and latency. Bandwidth is a constraint that limits the number of data connections that can utilize a network connection. If two connected VMs are mapped to two distinct clouds that are connected but not adjacent (i.e. $|E'| > 1$), then the bandwidth of all network connections of the selected path are utilized the same amount. The amount is equal to the requested data connection between these VMs.

Latency, on the other hand, is not a constraint but rather a factor that affects the performance of VMs and overall cluster. We consider two types of latencies in RalloCloud:

Deployment Latency is correlated to VM size (S) and the sum of the latencies (L) on the shortest path (P) between a VM and the user it is submitted by. It is also inversely correlated to the allocated bandwidth (B) on the path. Basically, deployment latency determines the time it takes to deploy a VM after it is mapped to a certain cloud and it is considered only in the initialization phase of the VM. Even if the selected cloud is in the same location as the user, there is a minimum deployment latency (M).

$$Deployment\ Latency = M + \sum_{i \in P} L_i + \frac{S}{B} \tag{1}$$

Communication Latency is between the communicating VMs that form a cluster. Contrary to deployment latency, it is in effect after the deployment of VMs. It has an impact on the execution time of the VM that is correlated to observed latency and the size of transferred data (D) and inversely correlated to the allocated bandwidth (B). Observed latency between two VMs is the sum of latencies (L) on the path (P) to which their data connection is mapped. If two VMs are deployed to the same cloud or there is no data connection, then the communication latency between

them is negligible.

$$Communication\ Latency = \sum_{i \in P} L_i + \frac{D}{B} \tag{2}$$

3.1.4. Cost

In addition to the fixed pricing policy for IaaS providers, RalloCloud also supports a dynamic pricing policy called Trough Filling [54]. It is based on yield management strategy in economics, and aims to maximize revenue from a limited and perishable resource. In our case, price of a certain cloud resource (CPU, memory, storage or bandwidth) varies directly with its utilization, that is, the less percentage of the resource is utilized, the lower is the unit price of that resource. It should be noted that utilization of a resource in a cloud affects its price only in that cloud, not in the whole federation.

Unit cost of a VM is the sum of its reserved resources multiplied by their unit prices at the time of deployment. Similarly, unit cost of a network connection is the reserved bandwidth multiplied by the bandwidth unit price. Then, total cost of a VM cluster is the sum of unit costs of its VMs and network connections, multiplied by execution time.

3.2. Performance Criteria

RalloCloud framework contains a module for measuring performance criteria to evaluate the quality of embeddings and compare algorithms. We reviewed related literature, identified several criteria, suggested some new ones and categorized them as given in Figure 3. To keep the presentation concise and due to the fact that evaluation results of some criteria came out similar, we choose 8 criteria (marked in bold) to consider in this paper. Our selection includes at least one criterion from each category. Definition of these criteria are given below.

Cloud-to-User Latency is measured between a VM cluster and the user who submits it. It affects the deployment latency of VMs.

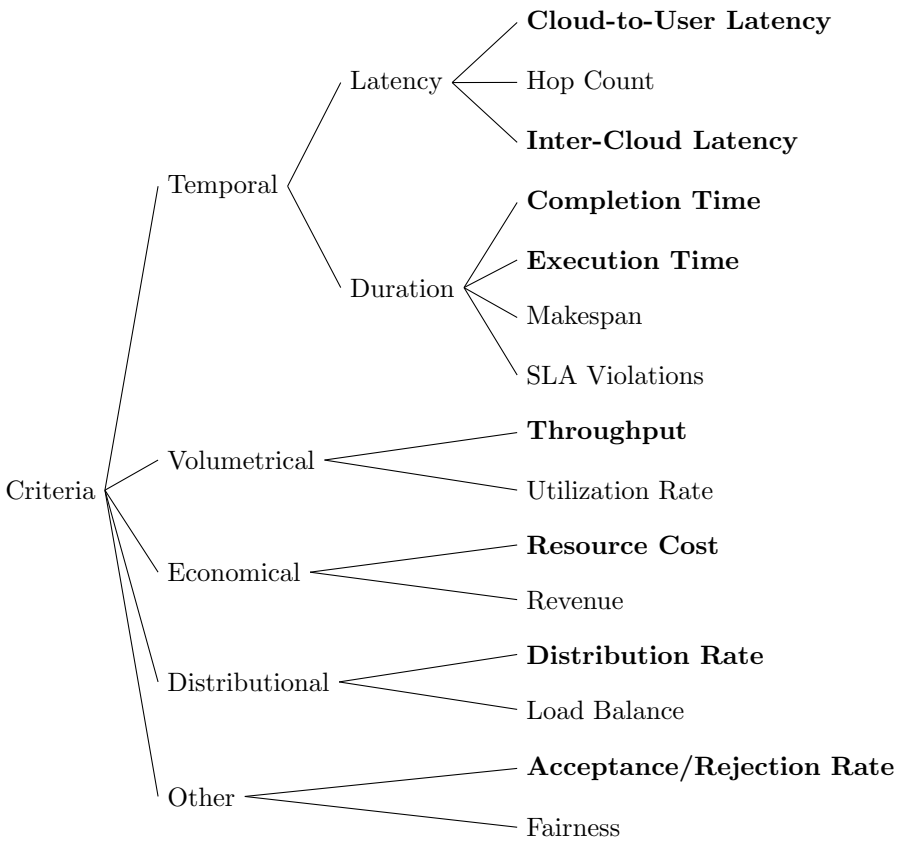


Figure 3: Hierarchical categorization of the performance criteria.

Inter-Cloud Latency is measured among the VMs that constitute a VM cluster. It affects the communication latency of VMs.

Completion Time is the duration between the arrival of a request and its successfully completion. It is normalized by the size of the task executed in million instructions. Completion time is affected by both deployment and communication latency as well as pending time. Pending time is measured when the cluster cannot be deployed immediately because there is no available cloud at that time.

Execution Time is the duration between the deployment of a VM cluster and its completion. Also normalized by task size, it is affected by communica-

tion latency but not the deployment latency or pending time.

Throughput is measured as the millions instructions processed in the whole federation per second (MIPS).

Resource Cost is the total cost of CPU, memory, storage and bandwidth for a VM cluster during its execution time. Calculation of the total cost is described is Section 3.1.4.

Distribution Rate is the extent that the VMs of a cluster are placed to separate clouds. We measure it as the ratio of number of distinct clouds employed for VMs to the number of VMs in the cluster.

Rejection Rate is the percentage of VMs that are submitted to a cloud but failed to be deployed due to lack of resources. Such a case may occur if the embedding or matching algorithm is unaware of cloud providers' utilization or if multiple instances of the algorithm submits VMs concurrently to the same cloud. Depending on the algorithm, rejected VMs may be dispatched to other cloud providers or queued.

4. Topology Based Mapping

Efficient use of the network infrastructure is crucial for the VMCE problem because of two reasons: (i) network latency affects execution time and cost, and (ii) availability of bandwidth affects the acceptance rate of new requests. Moreover, both these factors also contribute to overall utilization, throughput and revenue of the federation. Hence, the main factors that TBM algorithm is built to optimize are latency and bandwidth. We observed that decreasing latency and efficient use of bandwidth are not conflicting objectives, yet they are not completely parallel.

The main idea behind the TBM is to map a VM cluster request to a subset of clouds in the federation that has the same (or at least similar) network topology as the request. To achieve this, TBM algorithm searches for subgraphs of G_F that are isomorphic to G_C. If such a subgraph does not exist or if the mapping is

not valid, it deducts to a heuristic approach to find a homeomorphic one. Each cloud broker in the federation runs the algorithm locally for each incoming VM cluster request. In order to gain the benefits of the distributed VM placement that are explained in Section 1, TBM tries to map each VM to a different cloud, letting multiple VMs from different clusters to be hosted at the same cloud.

4.1. An Example Use Case Scenario

Consider a small cloud-based photo editing service. The service is deployed on a two-tier architecture where first tier contains the image processing algorithms and the user interface while the second tier is for persistence of user account information and images. The service provider wishes to replicate the first tier and host them in two different providers in order to achieve the following two benefits: (i) since the users of the service will be served by the closest replica, overall latency will be decreased, and (ii) in case of a failure in one of the replicas, the service will continue its execution and the failed replica can be recovered later (possibly in another cloud) without loss of data.

They do not prefer to replicate the second tier since it is not critical for the execution of the service and due to the high cost of ensuring data consistency. However, as the second tier has different resource requirements (see Table 2) than the first one, it may be economically beneficial to host it in a separate cloud provider where the storage pricing is more convenient. Consequently, a cluster of three VMs need to be deployed in three different clouds for the service. Resource requirements of these VMs are provided in Table 2. Here, VM1 and VM3 are for the replicas of the first tier and VM2 is for the second tier. Resource capacities of VMs in this scenario are taken from Amazon Web Services[1] EC2 dedicated instances m3.2xlarge and d2.xlarge.

As seen in the bandwidth column of Table 2, the service needs 500 Mbps dedicated (D) bandwidth between VM1 and VM2 as well as between VM2 and VM3. There is no requirement of bandwidth between VM1 and VM3. Thus, the

[1] https://aws.amazon.com/

VM	CPU Cores	Memory	Storage	Bandwidth
VM1	8	30 GB	2 x 80 GB SSD	1000 Mbps out
				500 Mbps to VM2 (D)
VM2	4	30.5 GB	3 x 2 TB HDD	500 Mbps to VM1 (D)
				500 Mbps to VM3 (D)
VM3	8	30 GB	2 x 80 GB SSD	1000 Mbps out
				500 Mbps to VM2 (D)

Table 2: Resource requirements of the VMs for the photo editing service.

virtual topology of the VM cluster is as given in Figure 1(a). In addition, being the interface of the service, VM1 and VM3 requires 1000 Mbps bandwidth-out each for the user access. Let us also assume that all five cloud providers in Figure 1(a) have enough available resources to accept any of these VMs.

An arbitrary mapping between VMs and clouds would result in high latency data connection between VM1 and VM2 or between VM2 and VM3. Consider the mapping in Figure 1(b), there is no direct and dedicated network connection between CP1 and CP5 so the latency between VM1 and VM2 is high. This would decrease the QoS for the users who are served by VM1 because of the delay perceived when loading and saving edited photos. Moreover, it will increase the execution time of the image processing algorithms and thus the cost of infrastructure. From the cloud providers (CP3) point of view, such a mapping is a waste of bandwidth capacity which could have been leased to another customer.

The TMB algorithm, on the other hand, would decrease the latency between VM1 and VM2 by mapping VM1 to CP3 instead of CP1. Because the subgraph consisting of CP3, CP4, CP5 and their inter-connections is isomorphic to the topology graph of the requested VM cluster as denoted by the bijective function f in Equation 3.

$$f = (\text{VM1} \mapsto \text{CP3}, \text{VM2} \mapsto \text{CP5}, \text{VM3} \mapsto \text{CP4}) \tag{3}$$

Naturally, the algorithm considers several other factors and conditions than this trivial example in mapping VMs to clouds. These are elaborated in the following subsection. Motivations of the TBM algorithm, some of which are demonstrated in this use case, are as follows:

- Reducing the average inter-cloud latency by placing communicating VMs to locations that are connected with dedicated network connections.
- Reducing the average cloud-to-user latency by prioritising subgraphs with low latency to user.
- Improving the QoS and decreasing the execution time as a result of low latency communication.
- Decreasing the resource cost for the service provider and increasing the throughput (and the profit) for the cloud provider.
- Finding effective, "good enough" mappings via heuristics when the optimum solution (isomorphic subgraph) is not available.

4.2. The TBM Algorithm

Figure 4 demonstrates the flow of the algorithm where the main scenario is represented with hollow rectangles and alternative heuristic part with the shaded ones. In addition, a high level pseudo code is given in Algorithm 1.

Inputs to the TBM algorithm are (i) overall utilization and capacity of each resource in each cloud data center, (ii) bandwidth utilization and capacity as well as average latency of each network connection between cloud data centers, (iii) resource requirements of each VM in the requested clusters, (iv) bandwidth requirement between each VM pair, and (v) location of the cloud user.

4.2.1. Subgraph Isomorphism Based Mapping

The first step of the algorithm is to fetch the VM cluster request and start an isomorphic subgraph search on the federation topology. If the search ends up with exactly one valid mapping, each VM is submitted to its matched cloud. If

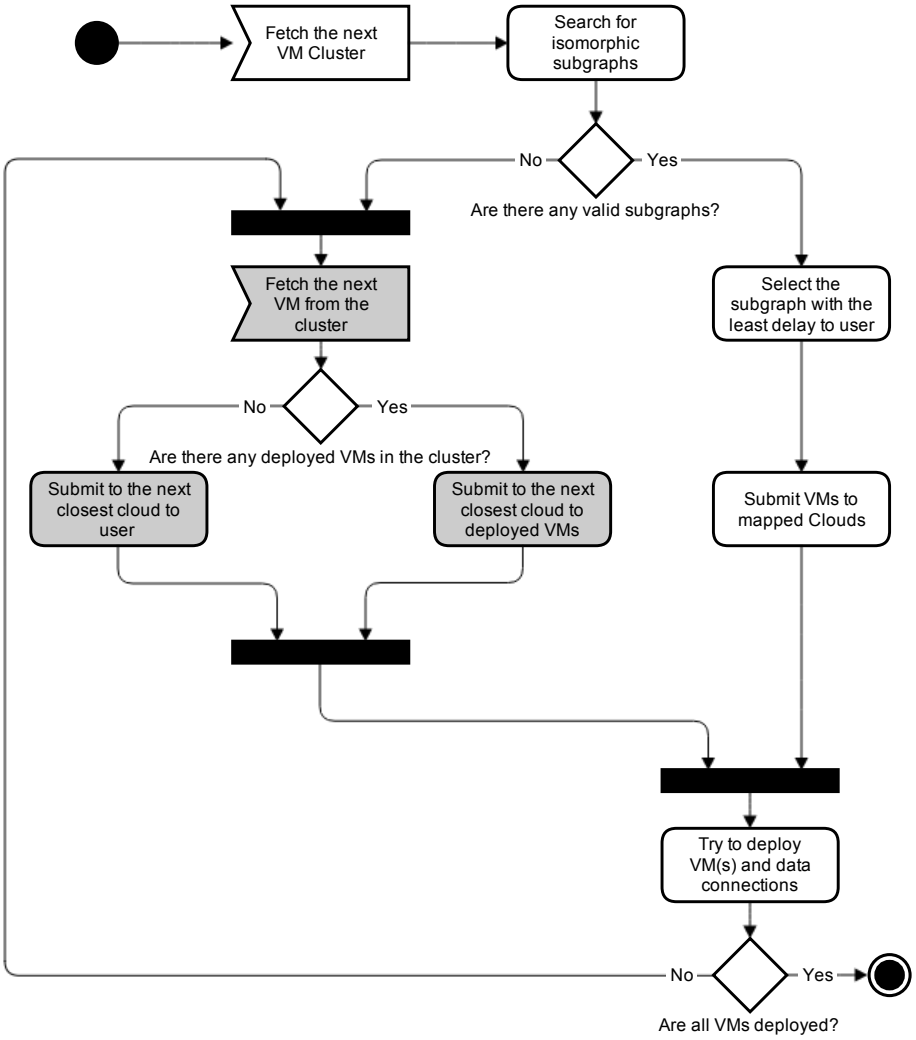


Figure 4: UML activity diagram of the TBM algorithm.

there are multiple candidates, however, they are sorted by average cloud-to-user latency and the VMs are submitted to the clouds in the subgraph with the least average latency. Algorithm is completed if all VMs are successfully deployed to the mapped clouds.

For the efficient discovery of isomorphic subgraphs, we implement the Local All Different (LAD) filtering procedure [17]. It helps to reduce the search space

Algorithm 1: Pseudo code of the TBM algorithm.

```
foreach cluster request G_C in queue do
    subgraphs[ ] ← SearchIsomorphicSubgraph(G_F, G_C)
    if size(subgraphs[ ])>0 then
        chosenSubgraph ← argmin_x(AvgLatency(subgraph[x], user))
        map each VM in G_C to the corresponding node in chosenSubgraph
    else
        foreach virtual machine VM in G_C do
            deployedVMs[ ] ← deployed(G_C)
            if size(deployed[ ])>0 then
                chosenNode ← argmin_x(AvgLatency(V_F[x], deployed[ ]))
            else
                chosenNode ← argmin_x(AvgLatency(V_F[x], user))
            map VM to chosenNode
    Try to deploy VMs at mapped nodes and allocate data connections
```

by pruning branches that do not contain solutions. Given a pair of nodes (n_p, n_t) from the pattern and target graphs, LAD filtering builds a bipartite graph where the two sets are the neighbours of n_p and n_t. Two nodes from each set are adjacent if the matching of these two nodes are not previously pruned. On that graph, it checks whether a bipartite matching exists such that all neighbours of n_p are covered. If there does not exist such a matching, then n_p and n_t are incompatible and all the branches that match them are pruned from the search tree.

The dominant part of the TBM algorithm in terms of time complexity is the isomorphic subgraph search. Thus, the complexity of TBM is equal to the complexity of the method used for this search. In the case of LAD filtering, time complexity is $\mathcal{O}(|N_p| \cdot |N_t| \cdot d^4)$. Here $|N_p|$ and $|N_t|$ are the number of nodes in the pattern and target graphs, respectively while d is the greater of the maximal degrees of these two graphs. Considering the relatively small number

of nodes in the pattern and target graphs, the complexity is not a major challenge in application of our approach. On the other hand, it is clear that TBM might become infeasible for cases where densely connected large networks are being utilized to handle requests where small scale VMs are interconnected via complicated graphs.

4.2.2. Heuristic Mapping

In two exceptional cases in the main mapping scenario, TBM fails to deploy VM cluster to an isomorphic subgraph. These are:

No mapping: There does not exist any isomorphic subgraphs of clouds that are valid (holding enough available resources).

Partial deployment: There exists at least one valid subgraph but some of the VMs are rejected by the clouds they are mapped to. Since the algorithm has multiple instances running in each broker, two or more brokers may concurrently decide to submit VMs to the same cloud provider. If the cloud does not hold enough available resources to deploy all these VMs, it rejects the VMs arriving after its full utilization. Consequently, not all VMs of the cluster can be deployed.

In both cases, TBM switches to the heuristic mode and aims to find a homeomorphic graph with low latency. In this mode, non-deployed VMs of the cluster are considered individually. Heuristic fetches an arbitrary VM from the cluster and checks whether there exists any other VM in the cluster that is successfully deployed to a cloud. If that is the case, the VM is submitted to the cloud that is available and would result in least average inter-cloud latency. However, if all VMs of the cluster are yet to be deployed, then the VM is submitted to the cloud with the least cloud-to-user latency.

Let us consider that a cluster with 3 VMs to illustrate with an example. If no mappings can be found for the requested topology, these 3 VMs will be matched to the clouds separately. The VM that is considered in the first place will be submitted to the an available cloud with the least latency to the user. The

next one will be submitted to an available cloud with the least latency to the first VM and finally the third VM will be submitted to an available cloud with the least average latency to the first two VMs. Since available clouds may be multiple hops away, the subgraph of the federation topology to which the VMs are mapped is homeomorphic but not necessarily isomorphic to the requested VM topology.

5. Evaluation

5.1. Experimental Setup

The simulations are carried out on the RalloCloud framework which is an extension to CloudSim.

VM cluster topologies and sizes may vary greatly in cloud systems [6]. In complicated cases, number of VMs can easily reach to factors of ten. On the other hand, if a medium size web application is considered, it may consist of a few VMs for persistence layer, user interface, etc. Even smaller applications may need only one VM. Without loss of generality, we generate VM clusters with VM count based on a Poisson distribution with a mean of three. Generated clusters may have data connections with one of the four topologies: complete, linear, circular and star, chosen uniformly at random.

Federation topology, on the other hand, is taken from a real-world example. We simulate the experimental network infrastructure explained in [55]. Implemented version of the topology (Figure 5) includes 14 point of presences across Europe and up to 4 virtualization servers at each location. The architecture follows the IaaS paradigm and the cloud capacities are heterogeneous in proportion to the number of virtualization servers at each point of presence.

Four simulation parameters and their effect on the embedding quality are monitored. Although RalloCloud and TBM support CPU, memory and storage resources, **VM size** is represented with only memory requirement of VMs for simplicity. It determines the number of VMs that can be deployed to a single cloud. In our simulation, we assign $64x$ of memory to each virtualization server

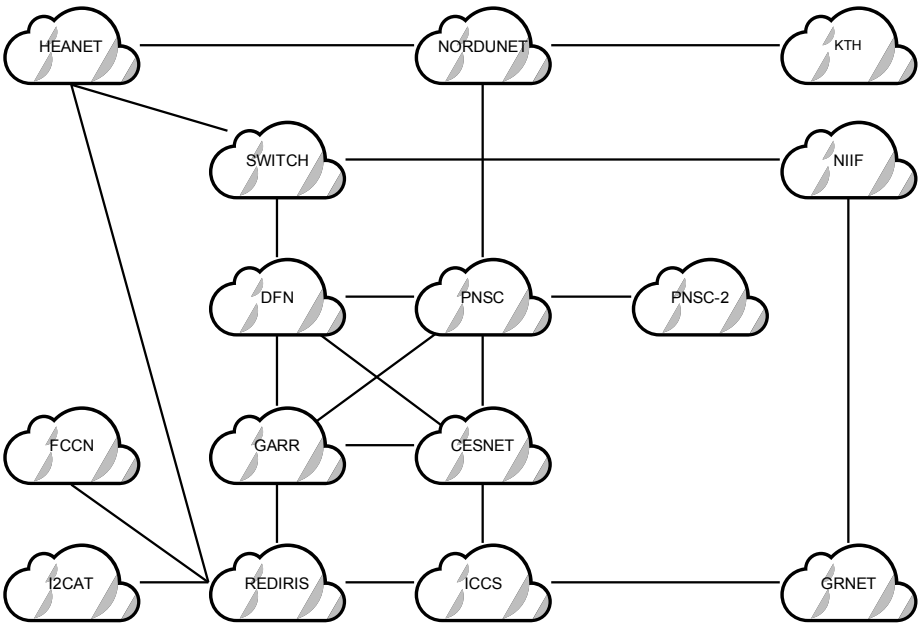


Figure 5: Physical Infrastructure Topology [55].

and $1x$ to $8x$ of memory to each VM. Then, **bandwidth size** of data connections between VMs is up to $8y$ and each network connection has a bandwidth capacity of $80y$. Another simulation parameter, **I/O data length** is the amount of data that is transferred between communicating VMs in a cluster. It is between $1z$ and $8z$ where the length of all data processed in a VM is $10z$. That means up to 80% of the processed data may be communicated between VMs.

Bandwidth size specifies required bandwidth allocation for each communicating VM pair while I/O data length specifies the size of the data that will be transferred on that allocated bandwidth. As the bandwidth size decreases and the data length increases, data transfer duration will extend. Another difference is in that the bandwidth size affects the subgraph search while I/O data length only affects VM's runtime performance.

Cloud locations are assumed to be user bases with varying **VM cluster demand** according to the human population density at that location. The demand is measured as the number of requests received from a user base. The

Simulation Parameter	Range	Default Value
VM memory size	$[x, 8x]$	$4x$
Inter-VM bandwidth size	$[y, 8y]$	$4y$
VM cluster demand per broker	$[16, 128]$	64
I/O data length	$[z, 8z]$	$4z$

Table 3: Ranges and default values of simulation parameters.

most demanding user base requests 16 to 128 VM clusters of varying sizes during the simulation period of 50 hours. Arrival times of the requests are selected uniformly at random within the range of $[0, 50)$ hours.

At each evaluation, only one of these four parameters varies while others are assigned their default values. Ranges and default values of simulation parameters is given in Table 3. The parameters in our experiments are defined as relative factors. For instance, in order to indicate that the VM requests generated during the simulations may contain $\frac{1}{64}$ to $\frac{1}{8}$ of a fixed cloud data center memory, we indicate VM memory range as $[x, 8x]$ and data center memory as $64x$. Same convention holds for other parameters except VM cluster demand. For each combination of parameters, the simulation is run 30 times.

Upper limits of the ranges (i.e. $8x$, $8y$, $8z$ and 128) are selected for two reasons: (i) higher values correspond to unrealistic cases such as a cloud data center that can only host a few VMs, and (ii) simulations result in halt because overall cloud capacity runs short to answer demand.

Finally, the unit prices of resources are taken from Amazon Web Services. At the time of our evaluation, the price of a 50 Mbps AWS Direct Connect Port is \$0.03 per hour and the on-demand price of an EC2 instance with a 1 GB memory (t2.micro) is \$0.013 per hour. These rates are directly proportional to the port speed and memory size, respectively.

5.2. Baseline Heuristics

TBM algorithm is evaluated against VMCE heuristics that have separate node and link mapping stages. Heuristics mainly differ for their node map-

ping strategies. For all heuristics, link mapping simply attempts to utilize the network connections on the shortest path between communicating VMs. We haven't included the linear programming techniques which are widely discussed in the literature since they are not feasible except the trivial cases of a few nodes.

Random (RAN) Mapping Brokers submit VMs to random available clouds known to them.

Round-Robin (RBN) Mapping Each broker has an arbitrarily ordered list of known clouds and it probes them in a circular fashion.

Least-Utilized-First (LUF) Mapping In order to exploit dynamic pricing strategy of the clouds and to balance their load, LUF mapping always submits a VM to the least utilized cloud which would have the lowest unit resource cost.

Least-Latency-First (LLF) Mapping LLF mapping is the same as RBN mapping except the list of clouds is sorted in ascending order of cloud-to-user latency instead of an arbitrary order. Assuming clouds that are close to the user in terms of latency would also be close to each other, objective is to reduce latency and increase VM performance.

5.3. Results and Discussion

Selected results out of 32 performance criteria – evaluation parameter combinations are discussed in this section. We excluded the heavily deadlocked simulations from the charts presented in this section. The deadlock situations are explained later in this section.

5.3.1. Inter-Cloud Latency

Figure 6 demonstrates the evaluation results with increasing size of VM. TBM has by far the least inter-cloud latency. Since RAN, RBN and LUF do not consider latency at all, they have the highest latency while LLF is around

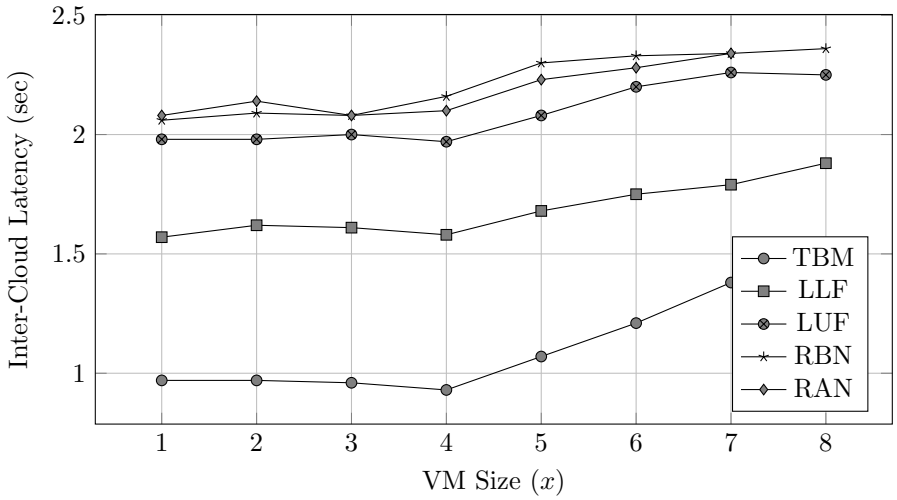


Figure 6: Evaluation results in terms of inter-cloud latency with varying VM size.

the middle. This indicates that inter-cloud latency assumption of LLF is more or less correct, but not enough to reach TBM. As expected, latency increases as the VMs gets bigger. This is because rejection rate of the latency-optimized mappings increases.

An interesting result occurs when the variable parameter is the bandwidth (Figure 7). TBM performs around the same while the baseline methods seem to yield lower latencies as the bandwidth request increases. The reason behind this is the scarcity of bandwidth in the federation. When users request more bandwidth, utilization of network connections increases. As a result, link mappings with multiple hops are likely to be rejected. After several rejections, even randomized heuristics come up with closely located VMs.

5.3.2. Cloud-to-User Latency

All parameters yield similar results for the average cloud-to-user latency, so we present only bandwidth size results in Figure 8 for brevity. Understandably, as the requests gets larger, possibility of deploying them close to their user decreases, thus latency increases. Although LLF makes decisions based on only cloud-to-user latency, TBM performs slightly better. This is due to the fact that LLF ignores efficient utilization of bandwidth capacity so the clouds with less

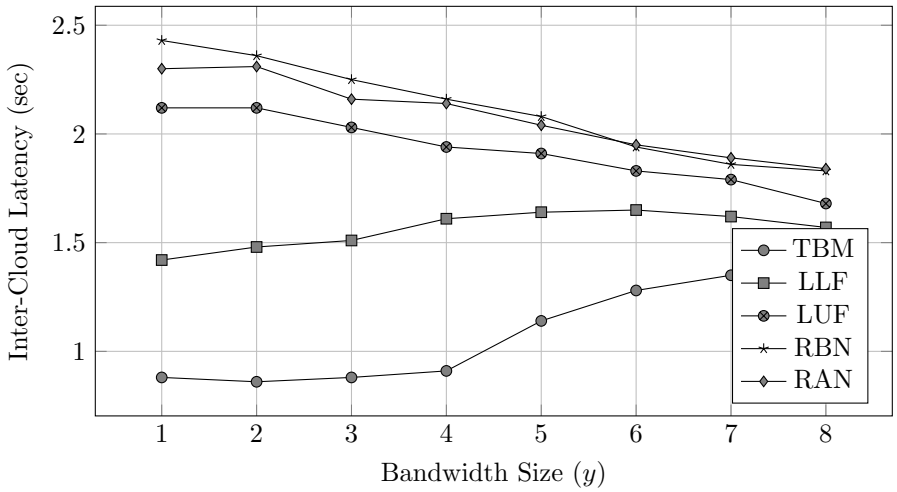


Figure 7: Evaluation results in terms of inter-cloud latency with varying bandwidth size.

latency cannot be utilized even when they have enough computing resources.

Cloud-to-user latency results indicate that, optimizing a single performance criterion and ignoring others in VMCE causes sub-optimal performance even for that criterion. Multi-objective approaches such as TBM are more suitable for the problem.

5.3.3. Completion Time and Throughput

Impact of the TBM's latency reduction can be observed on the execution and completion times of the VM clusters as well as overall system throughput. Figure 9 show the average completion times according to the VM size. We omit the execution time results since they are directly proportional to inter-cloud latency (Figure 6) while completion time also includes deployment latency and pending time. TBM reduces execution time up to 26% and deployment time up to 34% in comparison to the best performing heuristic (LLF).

Overall throughput of the system increases until a threshold (around 64 in x axis) as demand increases (shown in Figure 11). After that threshold however, brokers start to fail finding valid mappings and most new requests get rejected. Although this is unlikely in a real life scenario, we keep increasing the demand to test robustness of the algorithms. In that case, a deadlock may occur since some

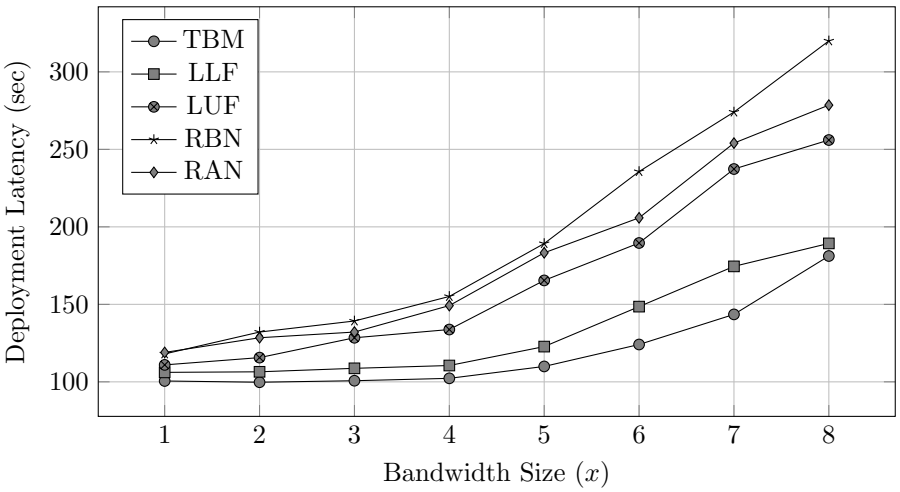


Figure 8: Evaluation results in terms of cloud-to-user latency with varying bandwidth size.

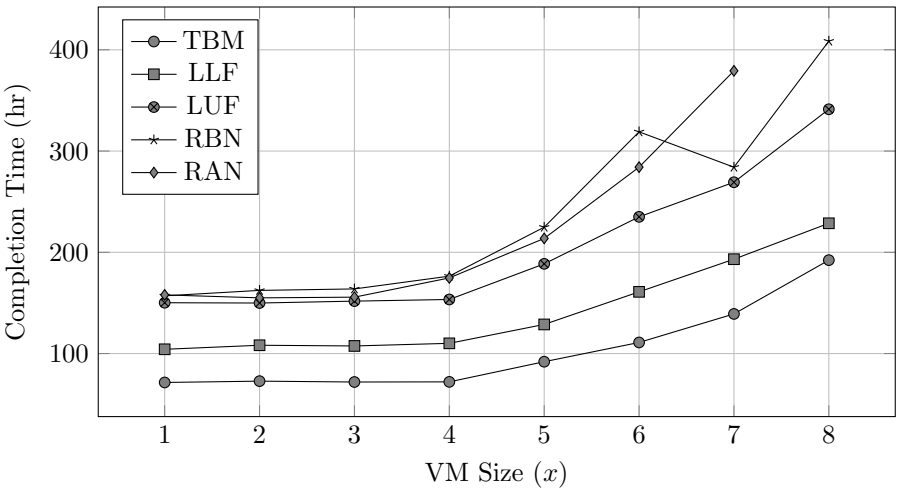


Figure 9: Evaluation results in terms of completion time with varying VM size.

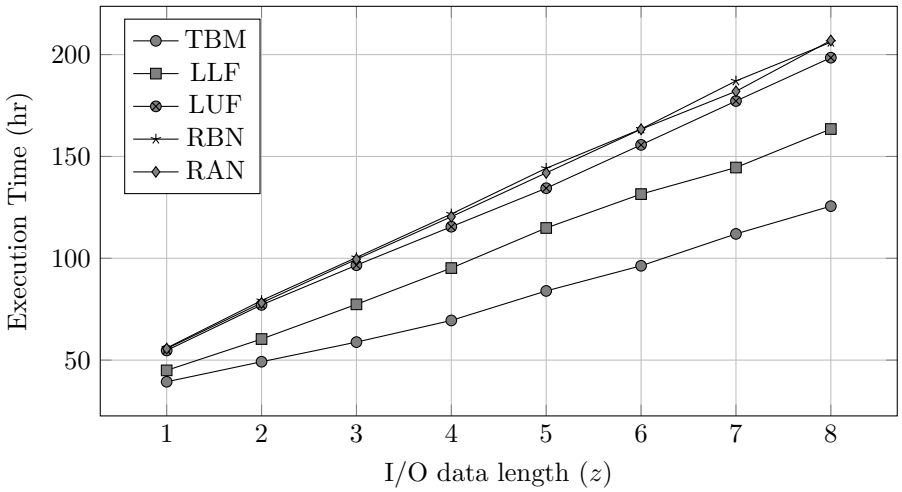


Figure 10: Evaluation results in terms of execution time with varying I/O data length.

VMs of clusters are deployed and other VMs are waiting for resources utilized by the VMs of other clusters (a case of dining philosophers problem). Missing data points and decreasing throughput in Figure 11 is due to such deadlocks. It is obvious that TBM algorithm can better utilize the resources of the system and it is more robust to demand peaks. A deadlock resolution technique is outside the scope of this paper and it is not needed for realistic demands.

As seen in Figure 12, the rate of rejected VM gets higher for increasing load while lowest rate is achieved by TBM. Here, we would like to remind that a rejection in RalloCloud is not permanent. It simply means current available resources at the matched cloud does not allow to deploy the VM thus it should be dispatched to another provider. Results also show that rejection rate converges around 94% for extreme loads. Likewise the throughput result discussed above, roughly the right half of the chart corresponds to unrealistically heavy workload which is useful to evaluate robustness of the algorithms. Under realistic workload, rejection rate of the TBM algorithm is under 30%.

5.3.4. Cost

As explained in Section 3.1.4, total cost of a VM cluster is directly proportional to its execution time under the same unit price. Because of that, change

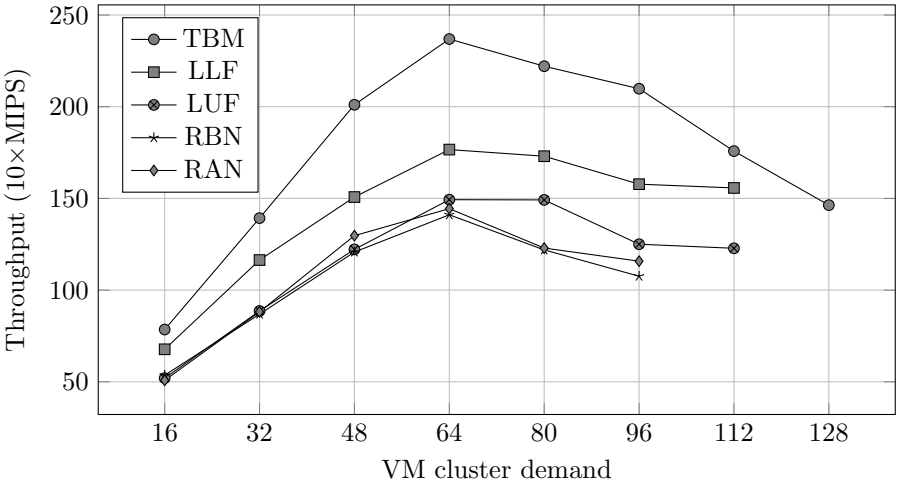


Figure 11: Evaluation results in terms of throughput with varying demand.

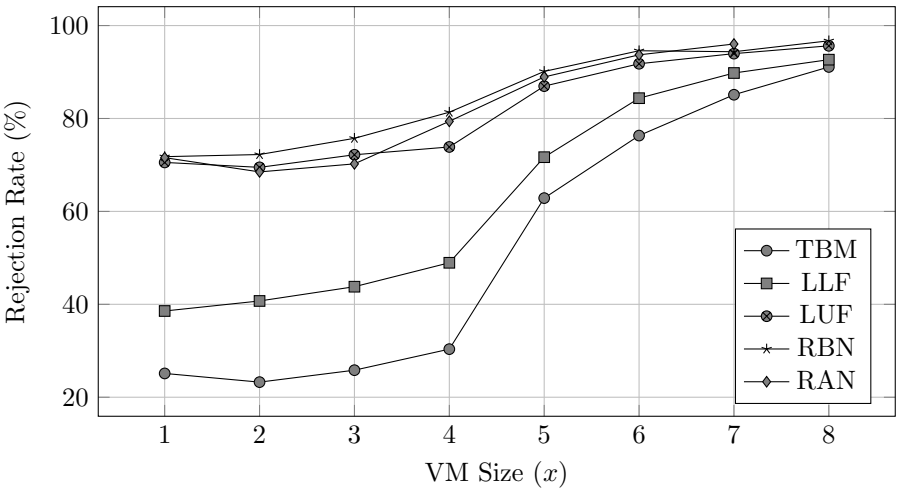


Figure 12: Evaluation results in terms of rejection rate with varying VM size.

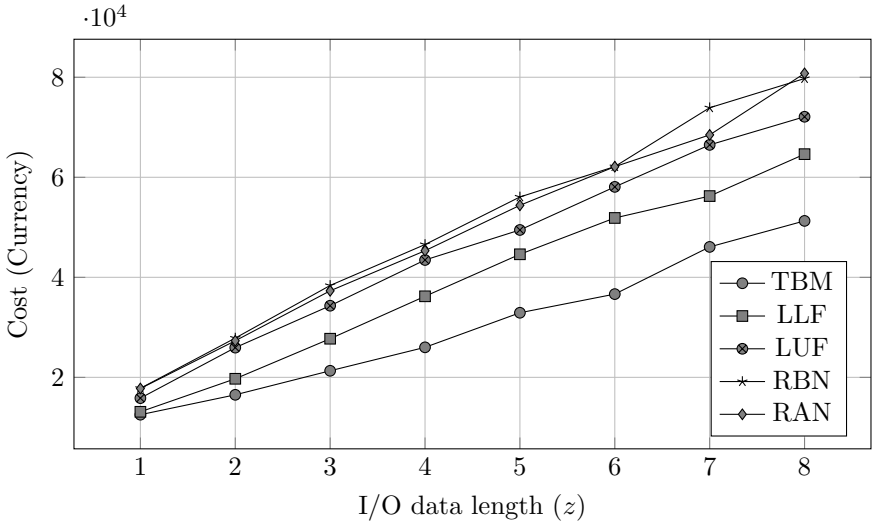


Figure 13: Evaluation results in terms of cost with varying I/O data length.

of costs in Figure 13 are quite similar to execution times in Figure 10.

Although, LUF always maps VMs to the clouds with least unit costs, that does not result in lower overall cost than TBM or LLF. Since LUF does not take latency and bandwidth into consideration, its lengthy execution time becomes the determinant for cost. As in the case of cloud-to-user latency performance of LLF, cost performance of LUF indicates that TBM is superior to heuristics that optimize single criterion due to its broad perspective of the whole aspects of the problem.

5.3.5. Algorithm Runtime Performance

Figure 14 demonstrates the runtime performance of the TBM algorithm. Four VM clusters with different topologies of five VMs are requested from each randomly generated federation. Federated cloud topologies are generated using the Watts-Strogatz model [56] with increasing number of vertices. This experiment is carried out on a workstation with Intel Xeon E5 CPU and 16 GB memory.

Results indicate that the algorithm can find a mapping and submit VMs to the corresponding clouds within a second for federations consisting of up to 70

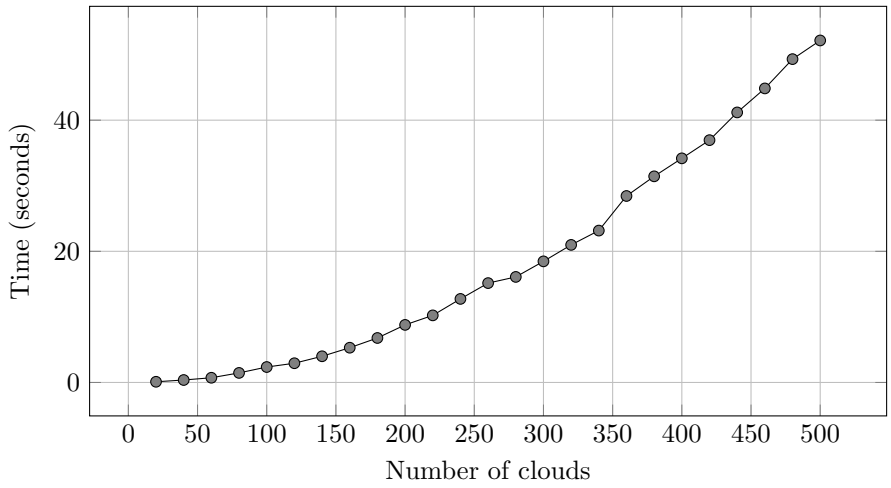


Figure 14: Runtime performance of the TBM algorithm with varying federation size

clouds. Even in extreme case of 500 clouds forming a federation, the algorithm can answer requests under a minute.

6. Conclusion

In this paper, we tackle the problem of efficiently embedding VM clusters onto the federated cloud infrastructure. The motivation for distributed VM placement includes easier failure recovery, better geographical coverage, vendor lock-in avoidance and reduced infrastructure cost.

Our approach places emphasize of minimizing latency and optimizing bandwidth allocation. TBM algorithm employs a subgraph search to locate clouds and network connections with a topology isomorphic to the VM cluster. If this search fails, then the algorithm falls back to heuristics in such a way that both latency and utilized bandwidth are decreased.

Additionally, we present RalloCloud simulation framework for modeling and simulating distributed resource allocation. It extends CloudSim with cloud federation and VM cluster related properties. RalloCloud also includes detailed network and cost modeling as well as several performance criteria in order to evaluate resource allocation algorithms in a realistic way.

Using RalloCloud, we evaluated TBM algorithm with several performance criteria against multiple greedy heuristics that optimize solely latency or cost. Results indicate that TBM outperforms all baseline heuristics in latency, job run time, throughput, cost and acceptance rate. They also emphasize that multiple performance criteria should be considered jointly for efficient VM cluster embedding.

We are currently focusing our efforts on managing resources in finer granular cloud environments such as fog computing or mobile cloud computing with cloudlets. Since the target topology is much broader and more dynamic, we strive to develop a distributed and context-aware version of our cluster embedding algorithm.

Acknowledgement

This research was partially supported by İTU-BAP (Grant No: 38450), NETAŞ PhD Project Incentive Award, and TÜBİTAK 2211 Graduate Scholarship.

References

[1] O. Sefraoui, M. Aissaoui, M. Eleuldj, Openstack: toward an open-source solution for cloud computing, International Journal of Computer Applications 55 (3) (2012) 38–42.

[2] D. Milojičić, I. M. Llorente, R. S. Montero, Opennebula: A cloud management tool, IEEE Internet Computing 15 (2) (2011) 11–14.

[3] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, R. Buyya, Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software: Practice and Experience 41 (1) (2011) 23–50.

[4] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, et al., The reservoir

model and architecture for open federated cloud computing, IBM Journal of Research and Development 53 (4) (2009) 4–1.

[5] R. Buyya, R. Ranjan, R. N. Calheiros, Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services, in: Algorithms and architectures for parallel processing, Springer, 2010, pp. 13–31.

[6] N. Grozev, R. Buyya, Inter-Cloud architectures and application brokering: taxonomy and survey, Software: Practice and Experience 44 (3) (2014) 369–390.

[7] C. Pittaras, C. Papagianni, A. Leivadeas, P. Grosso, J. van der Ham, S. Papavassiliou, Resource discovery and allocation for federated virtualized infrastructures, Future Generation Computer Systems 42 (2015) 55–63.

[8] A. Aral, T. Ovatman, Subgraph matching for resource allocation in the federated cloud environment, in: Proceedings of the 8th IEEE International Conference on Cloud Computing (CLOUD), IEEE, 2015, pp. 1033–1036.

[9] C. Papagianni, A. Leivadeas, S. Papavassiliou, V. Maglaris, C. Cervello-Pastor, A. Monje, On the optimal allocation of virtual resources in cloud computing networks, IEEE Transactions on Computers 62 (6) (2013) 1060–1071.

[10] A. Fischer, J. F. Botero, M. Till Beck, H. De Meer, X. Hesselbach, Virtual network embedding: A survey, IEEE Communications Surveys & Tutorials 15 (4) (2013) 1888–1906.

[11] Z. Zhang, S. Su, Y. Lin, X. Cheng, K. Shuang, P. Xu, Adaptive multi-objective artificial immune system based virtual network embedding, Journal of Network and Computer Applications 53 (2015) 140–155.

[12] J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters, Communications of the ACM 51 (1) (2008) 107–113.

[13] M. García-Valls, T. Cucinotta, C. Lu, Challenges in real-time virtualization and predictable cloud computing, Journal of Systems Architecture 60 (9) (2014) 726–740.

[14] G. Smaragdakis, N. Laoutaris, K. Oikonomou, I. Stavrakakis, A. Bestavros, Distributed server migration for scalable internet service deployment, IEEE/ACM Transactions on Networking 22 (3) (2014) 917–930.

[15] H. Xu, B. Li, Joint request mapping and response routing for geo-distributed cloud services, in: Proceedings of the 32nd Annual IEEE International Conference on Computer Communications (INFOCOM), IEEE, 2013, pp. 854–862.

[16] M. R. Garey, D. S. Johnson, Computers and intractability, Vol. 29, wh freeman New York, 2002.

[17] C. Solnon, Alldifferent-based filtering for subgraph isomorphism, Artificial Intelligence 174 (12) (2010) 850–864.

[18] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Generation Computer Systems 25 (6) (2009) 599–616.

[19] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al., A view of cloud computing, Communications of the ACM 53 (4) (2010) 50–58.

[20] S. Sotiriadis, N. Bessis, P. Kuonen, N. Antonopoulos, The inter-cloud meta-scheduling (icms) framework, in: Proceedings of the IEEE 27th International Conference on Advanced Information Networking and Applications (AINA), IEEE, 2013, pp. 64–73.

[21] D. C. Erdil, Autonomic cloud resource sharing for intercloud federations, Future Generation Computer Systems 29 (7) (2013) 1700–1708.

[22] G. A. Lewis, Role of standards in cloud-computing interoperability, in: Proceedings of the 46th Hawaii International Conference on System Sciences (HICSS), IEEE, 2013, pp. 1652–1661.

[23] J. L. Lucas-Simarro, R. Moreno-Vozmediano, R. S. Montero, I. M. Llorente, Scheduling strategies for optimal service deployment across multiple clouds, Future Generation Computer Systems 29 (6) (2013) 1431–1441.

[24] N. Ferry, A. Rossini, F. Chauvel, B. Morin, A. Solberg, Towards model-driven provisioning, deployment, monitoring, and adaptation of multi-cloud systems, in: Proceedings of the IEEE Sixth International Conference on Cloud Computing (CLOUD), IEEE, 2013, pp. 887–894.

[25] A. Barker, B. Varghese, L. Thai, Cloud services brokerage: A survey and research roadmap, in: Proceedings of the IEEE 8th International Conference on Cloud Computing, IEEE, 2015, pp. 1029–1032.

[26] J. Tordsson, R. S. Montero, R. Moreno-Vozmediano, I. M. Llorente, Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers, Future Generation Computer Systems 28 (2) (2012) 358–367.

[27] T. Cucinotta, D. Lugones, D. Cherubini, K. Oberle, Brokering slas for end-to-end qos in cloud computing., in: Proceedings of the 4th International Conference on Cloud Computing and Services Science (CLOSER), 2014, pp. 610–615.

[28] G. Aceto, A. Botta, W. De Donato, A. Pescapè, Cloud monitoring: A survey, Computer Networks 57 (9) (2013) 2093–2115.

[29] A. Iosup, R. Prodan, D. Epema, Iaas cloud benchmarking: approaches, challenges, and experience, in: Cloud Computing for Data-Intensive Applications, Springer, 2014, pp. 83–104.

[30] B. Varghese, O. Akgun, I. Miguel, L. Thai, A. Barker, Cloud benchmarking for performance, in: Cloud Computing Technology and Science (Cloud-

Com), 2014 IEEE 6th International Conference on, IEEE, 2014, pp. 535–540.

[31] K. Alhazmi, M. Abu Sharkh, D. Ban, A. Shami, A map of the clouds: virtual network mapping in cloud computing data centers, in: Proceedings of the 27th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), IEEE, 2014, pp. 1–6.

[32] M. Alicherry, T. Lakshman, Network aware resource allocation in distributed clouds, in: Proceedings of the 31st Annual IEEE International Conference on Computer Communications (INFOCOM), IEEE, 2012, pp. 963–971.

[33] J. Altmann, M. M. Kashef, Cost model based service placement in federated hybrid clouds, Future Generation Computer Systems 41 (2014) 79–90.

[34] I. De Falco, U. Scafuri, E. Tarantino, Two new fast heuristics for mapping parallel applications on cloud computing, Future Generation Computer Systems 37 (2014) 1–13.

[35] K. Konstanteli, T. Cucinotta, K. Psychas, T. A. Varvarigou, Elastic admission control for federated cloud services, IEEE Transactions on Cloud Computing 2 (3) (2014) 348–361.

[36] A. Leivadeas, C. Papagianni, S. Papavassiliou, Efficient resource mapping framework over networked clouds via iterated local search-based request partitioning, IEEE Transactions on Parallel and Distributed Systems 24 (6) (2013) 1077–1086.

[37] Y. Xin, I. Baldine, A. Mandal, C. Heermann, J. Chase, A. Yumerefendi, Embedding virtual topologies in networked clouds, in: Proceedings of the 6th International Conference on Future Internet Technologies, ACM, 2011, pp. 26–29.

[38] M. Chowdhury, M. R. Rahman, R. Boutaba, Vineyard: Virtual network embedding algorithms with coordinated node and link mapping, IEEE/ACM Transactions on Networking (TON) 20 (1) (2012) 206–219.

[39] J. Lischka, H. Karl, A virtual network mapping algorithm based on subgraph isomorphism detection, in: Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures, ACM, 2009, pp. 81–88.

[40] X. Wei, H. Li, K. Yang, L. Zou, Topology-aware partial virtual cluster mapping algorithm on shared distributed infrastructures, IEEE Transactions on Parallel and Distributed Systems 25 (10) (2014) 2721–2730.

[41] I. Houidi, W. Louati, W. B. Ameur, D. Zeghlache, Virtual network provisioning across multiple substrate networks, Computer Networks 55 (4) (2011) 1011–1023.

[42] K. Konstanteli, T. Cucinotta, T. Varvarigou, Optimum allocation of distributed service workflows with probabilistic real-time guarantees, Service Oriented Computing and Applications 4 (4) (2010) 229–243.

[43] F.-E. Zaheer, J. Xiao, R. Boutaba, Multi-provider service negotiation and contracting in network virtualization, in: IEEE/IFIP Network Operations and Management Symposium (NOMS), IEEE, 2010, pp. 471–478.

[44] O. Biran, A. Corradi, M. Fanelli, L. Foschini, A. Nus, D. Raz, E. Silvera, A stable network-aware vm placement for cloud systems, in: Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), IEEE, 2012, pp. 498–506.

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

[46] K. LaCurts, S. Deng, A. Goyal, H. Balakrishnan, Choreo: Network-aware task placement for cloud applications, in: Proceedings of the Internet Measurement Conference (IMC), ACM, 2013, pp. 191–204.

[47] X. Li, H. Wang, B. Ding, X. Li, D. Feng, Resource allocation with multi-factor node ranking in data center networks, Future Generation Computer Systems 32 (2014) 1–12.

[48] G. Sun, H. Yu, V. Anand, L. Li, A cost efficient framework and algorithm for embedding dynamic virtual network requests, Future Generation Computer Systems 29 (5) (2013) 1265–1277.

[49] P. Mell, T. Grance, The NIST definition of cloud computing, Tech. rep., Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg (2011).

[50] M. Alicherry, T. Lakshman, Optimizing data access latencies in cloud systems by intelligent virtual machine placement, in: Proceedings of the 32nd Annual IEEE International Conference on Computer Communications (INFOCOM), IEEE, 2013, pp. 647–655.

[51] I. A. Moschakis, H. D. Karatza, Multi-criteria scheduling of bag-of-tasks applications on heterogeneous interlinked clouds with simulated annealing, Journal of Systems and Software 101 (2015) 1–14.

[52] L. Thai, A. Barker, B. Varghese, O. Akgun, I. Miguel, Optimal deployment of geographically distributed workflow engines on the cloud, in: Proceedings of the IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom), IEEE, 2014, pp. 811–816.

[53] A. Aral, RalloCloud, `https://github.com/atary/RalloCloud` (2015).

[54] A. Greenberg, J. Hamilton, D. A. Maltz, P. Patel, The cost of a cloud: research problems in data center networks, ACM SIGCOMM computer communication review 39 (1) (2008) 68–73.

[55] M. Campanella, F. Farina, The FEDERICA infrastructure and experience, Computer Networks 61 (2014) 176–183.

[56] D. J. Watts, S. H. Strogatz, Collective dynamics of small-world networks, Nature 393 (6684) (1998) 440–442.