A Dynamic Virtual Machine Placement and Migration Scheme for Data Centers

Thuan Duong-Ba*, Tuan Tran†, Thinh Nguyen*, Bella Bose*
*School of EECS, Oregon State University
Corvallis, OR, 97331
Email: {duongba, thinhq, bose}@eecs.oregonstate.edu
†College of Information and Computer Technology, Sullivan University
Louisville, KY 40205
Email: ttran@sullivan.edu

***Abstract*—We study the problem of virtual machine (VM) placement and migration in a data center. In the current approaches, VMs are assigned to physical servers using on-demand provisioning. Such an approach is simple but it often results in a poor performance due to resource fragmentation. Additionally, sub-optimal VM placement usually generates unneeded VM migration and unnecessary cross network traffic. The efficiency of a datacenter therefore significantly depends on how VMs are provisioned and where they are placed. A good placement scheme will not only improve the quality of service but also reduce the operation cost of the data center. In this paper, we study the problem of optimal VM placement and migration to minimize resource usage and power consumption in a data center. We formulate the optimization problem as a joint multiple objective function and solve it by leveraging the framework of convex optimization. Due to the intractable nature of the combinatorial optimization, we then propose Multi-level Join VM Placement and Migration (MJPM) algorithms based on the relaxed convex optimization framework to approximate the optimal solution. The theoretical analysis demonstrates the effectiveness of our proposed algorithms that substantially increases data center efficiency. In addition, our extensive simulation results on different practical topologies show significant performance improvement over the existing approaches.**

***Index Terms*—Cloud computing, Virtual machine placement, Energy conservation, Convex optimization, Min rank.**

I. Introduction

Data centers are the crucial part of any cloud computing provider. Effective operation of data centers will not only help cloud providers reduce cost but also deliver high quality services to customers. Much work has been done on designing a data center networking architecture, e.g., Fat-tree [1], Bcube [2], VL2 [3] and Dcell [4], etc. Generally, physical hosts[1] are placed in racks which are then clustered, i.e., pod in [1] or in Bcube [2]. Each host cluster may contain hundreds to thousands of physical hosts. Host clusters are then interconnected by high speed switches and routers which could be in multiple levels to create more sophisticated architectures for data centers. Typically, the communication links within a data center have different capacities. For example, local links at host cluster, e.g., pod in [1], have a bandwidth of 1 Gbps; links between pods and routers have higher capacity, e.g., 10 Gbps. Currently, VMs' data are stored in a storage network [5], [6] and replicated on multiple storage nodes for high availability and reliability. Fig. 1 shows a typical architecture of a data center using a Fat-tree topology.

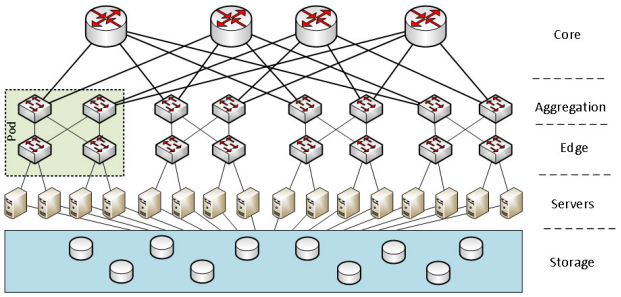
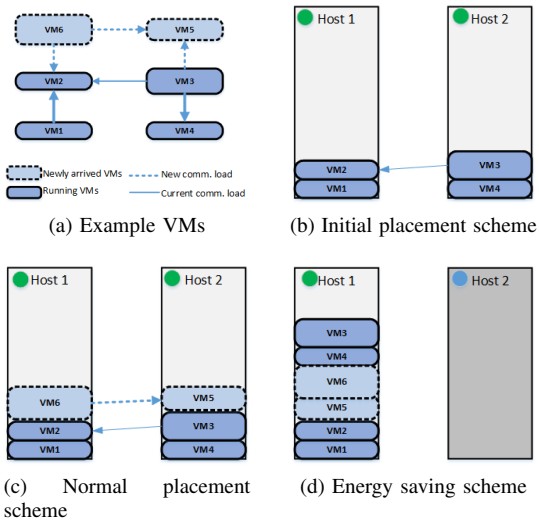


Fig. 1: A fat-tree topology based architecture of a data center

Due to large number of operational servers, energy conservation is one of the key factors that should be considered when designing a data center. In this paper, we focus on the energy consumed by PMs hosting VMs that process cloud services and by networking devices, e.g., data links, switches/routers, that transfer data among PMs. This type of energy consumption significantly depends on how VMs are placed or deployed onto PMs [7]. Current virtualization technology allows VM to be migrated from one PM to another without turning off virtual machines [8]. Such a VM migration capability provides flexibility in designing VM placement as a VM can be migrated from one VP to another without interrupting the running service. However, it will consume excessive energy if VMs are inappropriately deployed to the servers. Reducing number of running PMs and inter-host data traffic could significantly reduce energy consumption of data centers.

Consider an example of VM pattern in Fig. 2(a). In this illustration, VMs are depicted by nodes, and a directed edge and its thickness between two nodes represent the communication direction and load capacity, respectively. Running VMs are the ones which are already deployed onto PMs and serving users requests. On the other hand, newly arrived VMs are VMs awaiting for being deployed onto PMs. These new VMs

[1]In this paper, host, server and physical machine (VP) are used interchangeably.



(a) Example VMs (b) Initial placement scheme

(c) Normal placement scheme (d) Energy saving scheme

Fig. 2: Example VMs and different scheme

often come from users requests or cloud services that trigger provisioning more VMs. As an example, when there is an increase in traffic, service monitoring agent places requests to scale application fleets. On the other hand, when the service monitoring agent detects a defect instance that needs to be replaced, it sends a request to the cloud management service to replace the defect VM. Here, we define that current communication load is network traffic among the currently deployed VMs, whereas new communication load is defined by data traffic among newly deployed VMs or with currently VMs. Let's assume that the initial placement scheme (or the current placement scheme) of the 4 existing VMs is depicted in Fig. 2(b). When two new VMs arrive, a typical placement scheme which balances the load among PMs and/or minimizing network traffic would deploy each of the newly arrived VMs onto a host as shown in Fig. 2(c). In such a deployment, both PMs need to be switched to the running mode, and more importantly, there exists cross traffic between these two hosts. Alternatively, if only one PM can be used to accommodate all the VMs (Fig. 2(d)), the other PM can be switched to a standby mode to save energy. Furthermore, energy consumption could be further reduced by switching unused networking devices to standby mode as well.

In fact, migrating VMs from one PM to another is not free [9]. The cost of migration could be significant, and more importantly, it might cause interruptions of services. A good VM placement scheme decreases not only bottleneck traffic at backbone link but also responding time of applications and energy consumption. Most of the existing solutions focus on solving the VM placement problem and migration problem separately. In such an approach, the final solution might not be optimal as these two objectives are optimized separately. Additionally, cloud providers also offer on-demand services, where VMs are deployed or re-deployed frequently depending on user's need or monitoring service requests. For example, when a monitoring service detects a degrade in quality of service below a pre-specified threshold, it might trigger a request to replace defect VM instances, or scale up fleets, i.e., replacing VM instances with more powerful ones. Sometimes, the monitoring service might even issues request to scale down the fleet such as reducing VM instances or replacing VMs with less powerful instances to save cost during off-peak traffic periods. The system dynamics makes the problem of VM placement and migration much more complicated and considering the two objective functions separately in the existing approaches may result in poor performance.

In this article, we propose an adaptive VM management method to improve the efficiency and effectiveness of a data center in both resource usage and power consumption. Our proposed approach takes an arbitrary initial placement state of the data center and reallocates the load to an optimal state, where VM migration is minimized. Importantly, our proposed algorithm can cope with the dynamics of random patterns of VM requests in cloud data centers. In summary, our main contributions are:

- We propose a multi-objective formulation to the optimal VM management in data centers. We mathematically analyze and quantify the optimality of the multi-objective function.
- We theoretically show that finding the optimal solution is NP-hard. A heuristic algorithm based on a relaxed convex optimization is proposed to find a near optimal solution.
- We prove the convexity of the objective function and propose two Multi-level Joint VM placement and Migration (MJPM) algorithms based on its nuclear norm and augments. The nuclear norm based algorithm, MJPM-Nucnorm, obtains better results but it is slow and requires higher computational complexity. On the other hand, the augment based algorithm, MJPM-Attr, is scalable but does not guarantee of optimal solution. However, both algorithms produce better results compared to conventional algorithms. Importantly, the proposed MJPM algorithms solve both placement and migration problem together and adapt to the dynamics of data centers.
- The effectiveness of the proposed scheme is corroborated via both theoretical analysis and extensive simulations on practical network topologies.

The organization of the paper is as follows. In Section II, we summarize the existing literature on the problem. Then we describe the system model, assumptions, and optimization objectives in Section III. In Section IV, we formulate the energy conservation as a multi-objective optimization problem and provide an optimal solution based on the graph partitioning method. We then describe a heuristic algorithm which exploits the convexity of the relaxed objective function to find a near optimal solution in Section V. Extensive simulation results and discussion are provided in Section VI. Finally, we conclude the paper in Section VII.

II. Related work

Many approaches have been proposed for the problem of VM placement in different settings. In [10], the VM placement problem is modeled as an instance of the multi-dimentional bin-packing problem defined by a set of bins $S = \{S_1, S_2, ...\}$

with size V and n number of items of sizes $v_1, v_2, .., v_n$, $v_j \leq V$. The goal is to find the minimum number of bins needed to pack all items. Bin-packing is a combinatorial NP-hard problem, and many heuristic algorithms have been proposed to find an approximation solution to the problem. One of the most popular algorithms solving bin-packing problems is the First Fit (FF) algorithm [11], a greedy method. Initially, a bin is selected and items are chosen in an arbitrary order. The chosen item will be packed into any currently used bin as long as it doesn't overload the bin. When the item size is larger than the available space of the current bin, a new bin is added to accommodate it. FF is time-effective but it usually requires more number of bins than the optimal solution. Another modified version of FF is First Fit Decreasing (FFD) algorithm [11]. In FFD, items are first sorted in decreasing order then the FF algorithm is applied to the ordered items. FFD ensures that larger size items will be processed first.

Utilizing emulation, the authors of [12] proposed a heuristic VM placement algorithm based on emulated VM migrations in order to optimize the total completion time. In this work, researchers focuses on the minimizing the migration and placement time rather than energy consumption as well as cross server communication. Differently, the authors in [13] proposed a green computing algorithm with load prediction to balance server loads via an inovative skewness concept. However, the authors do not specify how the skewness proptery would be impacted when VMs are migrated out from hot spots for overload prevention and from cold spots for green computing. A probabilistically dynamic VM placement scheme is proposed in [14] to dynamically map VM arrivals to PMs in order to minimize over all energy consumption of data centers. Applying the same predictive approach, the authors in [15] proposed a forecasting based multi-objective optimization genetic algorithm for maximizing server utilization.

In addition, in [16], authors proposed a VM placement solution focusing on load balancing among physical hosts. On the other hand, authors in [17], [18] try to consolidate VMs into minimum number of physical servers in order to maximize the utilization of the resources. In [19], the authors proposed the Max-Min Multidimensional Stochastic Bin Packing (M^3SBP) algorithm based on the FFD and Dominant Resource First (DRF) [20] algorithms. These algorithms focus only on minimizing the number of servers and ignore the inter-server communication effects. Contrarily, Traffic-aware Virtual Machine Placement Problem (TVMPP) formulated in [21] focuses on minimizing the network traffic by dividing physical hosts into slots with different communication costs. However, TVMPP ignores the effectiveness of reducing the number of physical hosts usage.

Multiple performance metrics are considered in [22] to design an incremental VM consolidation solution avoiding frequent migrations. Differently, in [23], the authors proposed a genetic algorithm for the VM placement problem that minimizes resource wastage, power consumption and heat dissipation. An evolutionary framework is proposed in [24] exploiting data sociality to optimize the trade-off between server efficiency and balancing data partitions. MinCS and MinES algorithms are proposed in [25] to minimize the energy of VM scheduling problem. A VM placement framework based on traffic patterns among services is proposed in [26]. The proposed algorithm relies on VMs' requirements and service types to optimize link utilization via minimizing inter-server communication load and latency.

Most of current solutions solve the VM placement problem from scratch which may not work well given an arbitrary current stage of the cloud data centers. This makes it difficult to apply those proposed algorithms to an operational cloud center where there are lots of VMs currently being served. Migrating VMs to a specific configuration which can be used with the existing solution may work but it will result in high migration cost. More importantly, such an approach may not work in high dynamic data centers where system configuration changes often. Furthermore, to the best of our knowledge, proposed frameworks are mostly executed at centralized controllers and not applicable to practical large-scale data centers. A Joint Virtual Machine Placement and Migration (JPM) framework has been proposed and described briefly in our previous work [27]. In this work, we extend JPM and propose a Multi-level Joint Virtual Machine Placement and Migration (MJPM) framework in order to support practically large scale data centers. In MJPM, we also take into account scheduled hardware maintenance plans of cloud providers. Table I summarizes the key technical capabilities of existing approaches against our proposed MJPM in this paper.

III. System Model, Assumptions, and Optimization Objectives

In this section, we formulate the VM placement and migration problem for energy conservation in data centers. Before mathematically formulating VM related problems, we first discuss assumptions and parameters used in our model.

1) System Model and Assumptions: We consider a centralized controlling system which manages VM requests and makes decisions in a data center. VM requests can be initiated by customers or cloud services while the VMs leaving events are sent from physical hosts. At the beginning of each operational cycle, we consider two types of VMs:

- *Running VM*: This type consists of VMs that are currently being served. The *migration decision* will be determined for this VM type.
- *New VM Request*: This type accounts for new VM requests and *placement decision* will be considered for these VMs.

Additionally, we consider an on-demand cloud service model where VMs are often turned off once they finish processing their designated service. Finally, we assume that the cloud data center has enough capacity to serve all requests including current and new VMs.

2) Optimization Objectives: In our framework, we consider the following key parameters of a data center as our optimization objectives.

- *Load Distribution*: Balancing load among physical servers is not crucial provided that VMs are placed on each host such that the total load is under a capacity threshold. Cloud providers might want to fill VMs to

TABLE I: Comparison of different approaches

Factors \ Approaches	[21]	[23]	[13]	[28]	[22]	[26]	[12]	[14]	[15]	JPM [27]	**MJPM**
Energy consumption						✓		✓	✓	✓	✓
VM Placement	✓	✓		✓		✓	✓	✓	✓	✓	✓
VM Migration			✓		✓	✓	✓	✓	✓	✓	✓
VM Consolidation			✓		✓			✓		✓	✓
Communication	✓			✓	✓	✓				✓	✓
Multiple resources		✓						✓	✓	✓	✓
PM reliability								✓			✓
Joint/multi-objective		✓			✓				✓	✓	✓
Optimization technique	Min-Cut	Genetic Alg.	Greedy	N/A	M-Convex	N/A	N/A	NA	Genetic Alg.	Convex optimization	Convex optimization
Distributed processing											✓

physical hosts such that they can maximize the number of idle physical hosts which can be turned off, or put into standby/sleep mode, to save energy.

- *Cross Traffic Cost*: Minimizing communication among servers in order to save energy consumed by network devices and links. More importantly, minimizing cross communication traffic will reduce traffic jam at the bottle neck nodes in the data center.
- *Migration Cost*: Minimizing migration requests to save cost or energy as well as avoid degradation in quality of services hosted by VMs.

3) Notation: In this subsection, we describe our notations and metrics used in the paper.

- **Resources**: Let N be the number of hosts in the data center. Without loss of generality, we assume that there are M_0 VMs running and M_+ newly requested VMs. Assume that there are K types of resources that each VM may consume including number of CPUs, amount of memory, network bandwidth, storage space, etc. A K-element vector is used to represent resources demanded by a VM. The overall resources demanded by all VMs of the data center is then represented by a matrix $R^{M\times K}$. In addition, we assume that each physical host has limited resource of each type. Let $C^{N\times K}$ be the resource capacity matrix, i.e., each row is a server's resource capacity/threshold. Our goal is to provision new VMs onto physical hosts and migrate some running VMs onto different hosts in order to optimize energy consumption of the data center.
- **Communication Model**: In our model, we consider two types of network communication used by each VM. First, the amount of data that services or applications hosted by VMs exchanges with clients over the Internet. Second, the data exchanged among VMs within the data center. We note that the former is independent of placement schemes that is modeled as a resource demanded by VMs. On the other hand, the later communication type is modeled by a directed graph which is represented as adjacent matrix $A^{M\times M}$. We denote $X^{M\times N}$ as a placement matrix whose rows represent VMs and columns represent PMs. For example, $X(i,j) \in \{0,1\}$ with $X(i,j) = 1$ means that VM i is decided to be placed on host j. Each VM must be placed on one and only one server, i.e., $\sum_j X(i,j) = 1$. If VM i is scheduled to be terminated, $\sum_j X(i,j) = 0$.

TABLE II: Notation

Notation	Description
N	Number of active hosts
M	Total number of VMs
K	Number of resources
$R^{M\times K}$	Resource demand matrix
$C^{N\times K}$	Resource capacity matrix
$A^{M\times M}$	Communication cost matrix
$X^{M\times N}$	Placement matrix
$X_0^{M\times N}$	Initial placement matrix
p_0	Power consumption per host at idle mode (zero load)
$p_1^{1\times K}$	Power consumption vector
$\boldsymbol{\lambda}^{N\times 1}$	Migration cost vector
$L^{N\times 1}$	Maintenance penalty vector
Δt	Optimization interval
C_t	Overall power consumption of inter-server communication
C_p	Overall power consumption of physical server
C_m	Overall migration cost
L_r	Maintenance penalty vector

Let's $X_0^{M\times N}$ be the initial placement matrix. All rows in X_0 corresponding to being served VMs have one '1' element representing current placement; all rows corresponding to newly arrival VMs are $\mathbf{0}^T$.

- **Energy Measure**: In our model, energy consumed by physical hosts is divided into two main types: idle mode energy and running mode energy. Idle mode energy is needed to keep host up and running and ready for hosting VMs. Running mode energy is energy needed to process data. Each resource consumes different levels of power consumption. Let p_0 denote the power consumption of host at idle mode (zero load) and $p_1^{1\times K}$ power consumption vector whose elements are power consumption of 1 unit of corresponding resources. The power consumption by VMs can be written as $P_1^{M\times 1} = Rp_1$. Migrating VMs from one physical host to another also costs power consumption which depends on types of migrated resources. Normally, in-memory data and persistent data on storage devices are resources that need to be transferred from current host to new host.
- **Migration Cost**: We denote $\boldsymbol{\lambda}^{N\times 1}$ as the migration cost vector whose elements are the amount of energy needed to migrate a unit of corresponding resources from one PM to another. Intuitively, the element corresponding to

CPU is 0 since we don't need to migrate CPU; only in-memory data and persistent data stored on hard drives are migrated. Further, we denote $L^{1\times N}$ the maintenance penalty vector whose elements are the cost if a VM is served by a specific PM. The values of the elements in this vector are statically assigned by operators and they are proportional to the hardware maintenance cost. A value of 0 means that the physical server is healthy and can normally host VMs. On the other hand, a positive penalty (cost) value is assigned to a server being planned for maintenance and currently VMs hosted on the server. This is because the hosted VMs will need to be migrated away.

- **Operational Time**: Finally, Δt is defined as the effective period of optimization process. Our system will execute the optimization algorithm every Δt time.

Table II summarizes the notation used in this paper.

IV. Multi-Objective Optimization to VM Deployment

A. Multi-Objective Optimization Function

The overall energy consumption of a data center is formulated by a multi-objective function consisting of three terms: C_P - the energy consumption per time unit by all physical hosts, C_t - the inter-server communication load per time unit, C_m - the cost of VM migration, and L_r - the maintenance penalty. Mathematically, we write the multi-objective optimization as:

$$\text{Minimize:} \quad \alpha C_t \Delta t + \beta C_P \Delta t + \gamma C_m + \theta L_r \tag{1}$$

$$\textit{subject to:} \quad X(v,s) \in \{0,1\},\ \forall 1 \le v \le M, 1 \le s \le N, \tag{2}$$

$$X^T \mathbf{1} = \mathbf{1}, \tag{3}$$

$$X^T R, \preceq C, \tag{4}$$

where α, β, γ, and θ denote the weight factors indicating how much each term contribute to the objective function. Constrain (2) represents if a VM v is placed on server s; constrain (3) ensures that a VM is placed on only one server; constrain (4) guarantees the total resources requested by VMs won't exceed the server capacities. The function optimizes the energy over a period of time Δt. Note that both $C_t \Delta t$ and $C_P \Delta t$ are functions of time while migration cost C_m and L_r are a onetime cost. Next, we discuss the components of objective function.

1) *Communication Power Consumption*: This parameter accounts for power used for inter-server communication. It is proportional to the amount of data transmitted across the network (i.e., between servers.) Thus, minimizing communication load will reduce the network power consumption. The total communication load per time unit can be written as

$$C_t = \sum_s X(u,s)(1 - X(v,s))A[u,v].$$

Let D be the diagonal matrix representing the sum of communication cost/rate, i.e,:

$$D[u,v] = \begin{cases} \sum_i A[u,i] & if\ u \neq v \\ 0 & otherwise. \end{cases}$$

The communication load can be compactly rewritten as:

$$C_t = Tr(X^T(D - A)X),$$

where $Tr(.)$ denotes the trace of matrix - the sum of all main diagonal elements of the matrix. In fact, $D - A$ is the Laplacian matrix which is positive semi-definite.

2) *Total Power Consumption*: The amount of energy consumed per time unit at server s is expressed as

$$C_P^s = \begin{cases} p_0 + \sum_v X(v,s)P(v) & if\ \sum_v X(v,s) > 0 \\ 0, & otherwise. \end{cases}$$

Hence the total power consumption at all servers is computed by

$$\begin{aligned} C_P &= \sum_s C_P^s \\ &= \sum_s (p_0 + X(v,s)R^T p_1)) I_{\{\sum_v X(v,s)>0\}} \\ &= ||R^T p_1||_1 + \sum_s p_0 I_{\{\sum_v X(v,s)>0\}} \qquad (5) \\ &= ||R^T p_1||_1 + p_0 Rank(X). \qquad (6) \end{aligned}$$

The first term of (5) is a constant representing the total power consumption to serve all VM requests. The second term of (5) is the power consumption of running hosts. The indicator function $I_{\{\sum_v X(v,s)>0\}}$ indicates that if there is at least a VM placed on a specific physical machine. Thus, minimizing Pw is equivalent to minimizing the number of physical machines needed to server all VM requests. Mathematically, minimizing Pw is equivalent to minimizing the rank of the placement matrix X as in (6).

3) *Migration Cost*: The migration cost can be determined by the total energy consumed to migrate VMs. The energy consumed by VM migration is mainly caused by data transferred from one physical host to another [9]. We have

$$C_m = (R\boldsymbol{\lambda})^T (X_0 \circ (1 - X))\mathbf{1}_N,$$

where "$\circ$" operator denotes the element-wise product (Hadamard product) of two matrices. Each element in $X_0 \circ (1-X)$ represents if corresponding VM was initially placed on one server and finally re-placed on another server. Hence $\mathbf{1}_M^T (X_0 \circ (1 - X))\mathbf{1}_N$ is the total number of VM migration requests. We note that the migration cost depends only on the changes in primary servers of VMs that are already placed on physical hosts. Newly assigned VMs do not contribute to the migration cost.

4) *Maintenance Penalty*: Commodity hardware is not always reliable; therefore, cloud computing providers regularly have schedules for hardware maintenance. Outages and/or quality degradation will occur if hosted services are being served by VMs associated with those maintained hardware. In order to guarantee service agreement level, VMs should be proactively migrated away from those PMs before their maintenance occurrences. We introduce a penalty vector $L^{1\times N}$ that models the penalty of a VM not being migrated from a PM before its maintenance. Specifically, we denote $L(s)$ as the penalty that a VM

is running on server s might bear if not being migrated to another healthy PM. $L(s) = 0$ implies that server s is normally operational. Operator will choose to set $L(s)$ values in order to reflect upcoming scheduled maintenance. A large value of $L(s)$, e.g., ∞, indicates that VMs on server s will experience severe penalty if their host PM does not change and the optimization algorithm will force to migrate all VMs hosted by server s to a new server immediately. We have that

$$L_r = XL.$$

Thus, minimizing our objective function in (1) can be rewritten as:

$$\text{Minimize:} \quad \alpha C_t \Delta t + \beta^r Rank(X) \Delta t + \gamma C_m + \theta L_r, \quad (7)$$

where β^r is the equivalent coefficient when we replace energy conssumption C_P by $Rank(X)$. Minimizing $Rank(X)$ can be approximated via minimizing the nuclear norm of X [29]:

$$||X||_* = \sum_{i=1}^{r} \sigma_i(X),$$

where $\{\sigma_i(X)\}$ are singular values of matrix X. However, determining nuclear norm is known to be computational expensive as it needs to compute the eigenvalues of a large-size matrix $X'X$. We have the following results.

Theorem 1. *The placement and migration problem is NP-complete.*

Proof. The proof is done via reducing the placement and migration problem to the bin packing problem which is NP-complete [30]. Suppose that there are M items with volumes $v_1, v_2, .., v_m$ and N identical bins $S_1, S_2, .., S_N$ with capacity V (N is large such that there are enough bins to pack all items). Let $X \in \mathbb{R}^{M\times 1}$ be the vector representing item placements, e.g., $X(i) = k, i \in [1..M], k \in [1..N]$ means item i is placed into bin k. The bin-packing problem is to find $N_0 \leq N$, the minimum number of bins, such that

$$\begin{cases} 0 < \sum_i v_i I_{\{X(i)=k\}} \leq V, & \forall k \in [1, .., N_0] \\ \sum_i v_i I_{\{X(i)=k\}} = 0, & \forall k \in [N_0 + 1, .., N], \end{cases}$$

where $I_{\{X(i)=k\}}$ is the indicator function, i.e.,

$$I_{\{X(i)=k\}} = \begin{cases} 1 & \text{if } X(i) = k \\ 0 & \text{o.w.} \end{cases}$$

Consider the following VM placement and migration problem. Suppose that the data center has N identical physical servers. Let $c_0 \in \mathbb{R}^{K\times 1}$ be the resource capacity vector, and $r_0 \in \mathbb{R}^{K\times 1}$ be the atomic resource demand vector such that $\frac{r_0(1)}{c_0(1)} = \frac{r_0(2)}{c_0(2)} = .. = \frac{r_0(K)}{c_0(K)}$. There are M VMs with resource demand vectors which are multiple times of the atomic resource demand vector. In other words, the resource demand matrix $R \in \mathbb{R}^{M\times R}$ satisfies $R(v,:) = p \times r_0^T$ with $p > 0$ being a real-value multiplier. Assume also that there are no communication load among VMs.

The next step is to normalize the problem by dividing VM sizes by corresponding resource capacities of servers. In the normalized problem, all servers have capacities of 1 for all resources and VMs' resource demand vectors have all elements with the same value. In addition, the VM placement and migration problem at the first iteration is reduced to the placement problem since the migration cost will be 0. Now we map $\frac{r_0(1)}{c_0(1)}$ to v_i, the VM placement and migration problem can be restated as follows.

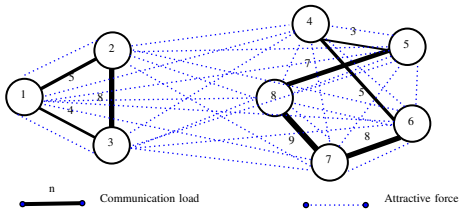


Fig. 3: Communication load and attractiveness of VMs

Given N identical server of size $V = 1$ and M VMs of sizes $v_1, v_2, ..., v_M$. Let $X \in \mathbb{R}^{M\times 1}$ be the vector representing VM placements, e.g., $X(i) = k, i \in [1..M], k \in [1..N]$ means VM i is placed into server k. Find $N_0 \leq N$, the minimum number of servers, such that

$$\begin{cases} 0 < \sum_i v_i I_{\{X(i)=k\}} \leq V, & \forall k \in [1, .., N_0] \\ \sum_i v_i I_{\{X(i)=k\}} = 0, & \forall k \in [N_0 + 1, .., N]. \end{cases}$$

Apparently, the solution to the normalized placement and migration problem exists if and only if there exists a solution to the normalized bin-packing problem. □

B. Graph Partitioning Based Rank Minimization

Assume that there exists a (weakly) attractive force[2] between any pairwise VMs. The attractive force would try to pull VMs together on a single machine. In a partitioning scheme, an attractive force is satisfied if both VMs are placed on the same partition (host). A good partitioning scheme will satisfy as many attractive forces as possible, i.e., minimizing the number of unsatisfied attractive forces. The attractive force graph can be considered as a fully connected graph where edges are pairwise attractive forces. Minimizing the number of unsatisfied attractive forces is to minimize the number of edges cut of the attractive force graph. Fig. 3 shows an example of two groups of VMs serving two cloud services. An optimal placement solution in terms of inter-server communication load minimization might place VMs on different servers. If a single physical host can accommodate all VMs, the optimization scheme that minimizes both inter-server communication cost and unsatisfied attractive force would place all VMs on the same physical server which will save the power consumption of using two servers. Given a fully connected graph $G = (V, E)$ of M vertices and $M(M-1)/2$ edges. Assume that G is divided into $K \leq M$ partitions; and

[2]Attractive force can be interpreted as augmented communication load between any pairwise VMs.

M_i ($i \in [1..K], 1 \leq M_i \leq M-K$ and $\sum_i M_i = M$) be the number of vertices belonging to partition i. Let E_i denote the number of internal edges within partition i. We have that

$$E_i = \frac{1}{2}M_i(M_i - 1).$$

We next show that minimizing $\#edges(cut)$ is equivalent to minimizing $rank(X)$. Let $\mathcal{M}(K)$ be the set of K non-zero positive integers $M_1, .., M_K$ that satifies $\sum_{i=1}^{K} M_i = M$. The partitioning scheme associated with $\mathcal{M}(K)$ has the total number edges in all cuts:

$$\epsilon(G, \mathcal{M}(K)) = \frac{M(M-1)}{2} - \frac{1}{2}\sum_{i=1}^{K} M_i(M_i - 1).$$

Claim 1. *Given a partitioning scheme $\mathcal{M}(K)$. If there is a partition i having more than 1 vertices, i.e., $M_i > 1$, and $M_i^{'}$ vertices are selected to create a new partition ($1 \leq M_i^{'} < M_i$). Let $\mathcal{M}(K+1|K)$ be the new partitioning scheme which has $K+1$ partitions, i.e., $\mathcal{M}(K+1|K) = \{M_1, ..., M_i - M_i^{'}, ..., M_K, M_{K+1} = M_i^{'}\}$. We have the following property:*

$$\epsilon(G, \mathcal{M}(K+1|K)) = \epsilon(G, \mathcal{M}(K)) + M_i^{'}(M_i - M_i^{'}).$$

Proof.

$$\begin{aligned}
\epsilon(G, \mathcal{M}(K+1|K)) &= \frac{M(M-1)}{2} \\
&\quad -\frac{1}{2}\sum_{j=1, j\neq i}^{K} M_j(M_j - 1) \\
&\quad -\frac{(M_i - M_i^{'})(M_i - M_i^{'} - 1)}{2} \\
&\quad -\frac{M_i^{'}(M_i^{'} - 1)}{2} \\
&= \epsilon(G, \mathcal{M}(K)) + M_i^{'}(M_i - M_i^{'})
\end{aligned}$$

□

In other words, further partitioning a given partitioning scheme always results in higher number of edges in cuts. In reverse, we have the following Claim:

Claim 2. *Given a partitioning scheme $\mathcal{M}(K)$; denote $\mathcal{M}(K-1|K)$ as the new partitioning scheme obtained by merging partition i and j in $\mathcal{M}(K)$ into a single partition, the number of edges in cut of new partitioning scheme will reduce:*

$$\epsilon(G, \mathcal{M}(K-1|K)) = \epsilon(G, \mathcal{M}(K)) - M_i M_j.$$

Proof.

$$\begin{aligned}
\epsilon(G, \mathcal{M}(K-1|K)) &= \frac{M(M-1)}{2} \\
&\quad -\frac{1}{2}\sum_{k=1, k\notin\{i,j\}}^{K} M_k(M_k - 1) \\
&\quad -\frac{(M_i + M_j)(M_i + M_j - 1)}{2} \\
&= \frac{M(M-1)}{2} \\
&\quad -\frac{1}{2}\sum_{k=1, k\notin\{i,j\}}^{K} M_k(M_k - 1) \\
&\quad -\frac{M_i(M_i - 1)}{2} \\
&\quad -\frac{M_j(M_j - 1)}{2} - M_i M_j \\
&= \epsilon(G, \mathcal{M}(K)) - M_i M_j.
\end{aligned}$$

□

In other words, merging partitions always reduce the number of edges in cuts.

Let $\mathscr{M}(K)$ be the set of all possible $\mathcal{M}(K)$. We denote

$$\mathcal{M}^*(K) = \arg\min_{\mathscr{M}(K)} \epsilon(G, \mathcal{M}(K)).$$

We now obtain our main result.

Theorem 2. *Given two integers K_1 and K_2 between 1 and M, i.e., $1 \leq K_1 < K_2 \leq M$:*

$$\epsilon(G, \mathcal{M}^*(K_1)) < \epsilon(G, \mathcal{M}^*(K_2)).$$

Proof. Let $\mathcal{M}_m^*(K_1)$ be the partitioning scheme resulted from merging partitions of $\mathcal{M}^*(K_2)$. From Claim 2 we have:

$$\epsilon(G, \mathcal{M}_m^*(K_1)) < \epsilon(G, \mathcal{M}^*(K_2)).$$

On the other hand,

$$\epsilon(G, \mathcal{M}^*(K_1)) < \epsilon(G, \mathcal{M}_m^*(K_1)).$$

The result then just follows.

□

Let X be the matrix representing a partitioning scheme, D_M be the diagonal matrix whose diagonal elements are $M-1$, and A_M be the adjacent matrix representing the fully connected graph G. The total number of edges of G is $\frac{1}{2}Tr(X^T D_M X)$; the total internal edges in all partitions of the partition scheme associated with X is: $\frac{1}{2}Tr(X^T A_M X)$. Therefore the total number of edges in all cuts is:

$$\frac{1}{2}Tr(X^T D_M X) - \frac{1}{2}Tr(X^T A_M X) = \frac{1}{2}Tr(X^T (D_M - A_M) X).$$

In other words, minimizing number of edges cut can be done via minimizing $Tr(X^T(D_M - A_M)X)$. Note that $D_M - A_M$ is the Laplacian matrix of the fully connected graph G and positive semi-definite.

V. Relaxed Convex Optimization Based Heuristic Algorithm

A. Relaxed Convex Optimization

Since the minimization problem for the placement and migration problem is NP-complete, we propose a heuristic algorithm based on relaxed convex optimization of (7). Relaxing X to accept real values in $[0, 1]$, the energy saving optimization problem for placement and migration VMs in cloud centers becomes:

Minimize:

$$F(X) = \alpha C_t(X) + f^R(X) + \gamma C_m(X) + \theta L_r(X) \quad (8)$$

$$\text{subject to: } 0 \leq X(u,v) \leq 1, 1 \leq u, v \leq M,$$
$$X^T\mathbf{1} = \mathbf{1},$$
$$X^T R \preceq C,$$

where $f^R(X)$ can be chosen as $f^R(X) = \beta_1^r ||X||_*$ or $f^R(X) = \beta_2^r Tr(X^T(D_M - A_M)X)$. Since $D - A$ and $D_M - A_M$ are the Laplacian matrices of communication graph and the attractive force graph respectively, they are positive semi-definite [31]. Hence all four terms in $F(X)$ are convex. In addition, all constraints are also convex therefore relaxed VM placement and migration problem is a convex optimization problem which can be solved efficiently. In fact, in our simulation we use CVX package [32] to solve the relaxed convex optimization problem. Solving relaxed optimization problems return a solution which has real values. We can make final decision based on the real solution to obtain an optimal VM placement and migration scheme.

We next describe how to apply solution of the relaxed problem in determining final decision of placing and migrating to cloud centers.

B. Joint VM Placement and Migration (JPM) Algorithm

We next describe the JPM algorithm for minimizing the energy consumption of a data center. Algorithm 1 sketches the main steps in JPM. At the beginning of every optimization period Δt, assume that there are M_0 VMs currently being served, and there is a queue of M_1 new VMs waiting to be served. Without loss of generality, assume that the total capacity of all physical hosts of the cloud center can accommodate all VMs requests including VMs being served and newly arrived VMs, i.e., $R^T\mathbf{1} \leq C^T\mathbf{1}$.

Let X_0 be the initial placement scheme: first M_0 rows corresponds to currently being served VMs; last M_+ rows corresponds to newly arrival VMs. Let $\mathcal{V}$ be the set of VMs that have been made decision; the corresponding rows in matrix X will be kept fixed during successive optimization iterations. On the other hand, let $\mathcal{V}^{'}$ be the set of undecided VMs which are corresponding to optimization variables in successive iterations. Additionally, let $S^{'}$ be the set of available servers which are under load and willing to accept new possible VM requests. Denote $\overline{S'}$ the set of saturated servers who are underload but will be overloaded if another VM is placed on. At every optimization iteration, we fix the VM placement/migration decisions that have been made in previous iterations. We also exclude the servers that are saturated. The remaining elements are optimization variables. Let Ω be the set of placement matrix X whose elements at corresponding rows in $\mathcal{V}$ or columns in $\overline{S'}$ are fixed. The optimization problem now will have $|\Omega| = |\mathcal{V}^{'}| \times |S^{'}|$ variables. In each optimization iteration, X^*, the optimal solution to the relaxed optimization problem (8), is used in order to make decision.

Algorithm 2 summarizes how decisions are made based on X^*. Let W be the decision window which is the maximum number of VMs that will be decided to be placed on a possible physical server. In each iteration, a row v_0 in X^* which has the highest value is selected and the server s_o corresponding to the highest value of v_0 is chosen. However, if chosen server s_o will be overloaded, the decision will be discarded and s_o will be removed from available server set of next optimization iterations. The decision is discarded by not adding v_0 to

Algorithm 1 JPM Algorithm

```
M_0 ← # VMs currently being served.
M_+ ← # new VMs waiting for placement.
M ← M_0 + M_+
X_0 ← current placement scheme.
L ← maintenance plan vector.
Initialize α, β, γ, W, Δt
V ← ∅ //Set of VMs having been made decisions
V' ← [1..M] //Set of remaining rows
S' ← [1..N] //Set of available servers
X(i, j) ← 0 ∀i ∈ V, j ∈ S'
while |V'| > 0 do
    Ω ← {X|X(i, j) = constant if i ∈ V
            OR (i ∈ V' AND j ∈ S'-bar)}
    X* ← arg min_{X∈Ω} F(X)
    (X, V'_d, S') ← Make Decision (X*, W, X, V', S')
    V ← V ∪ V'_d
    V' ← V' \ V'_d
end while
Return X.
```

Algorithm 2 VM Deployment Algorithm

```
Input X*, W, X, V', S'
V'_d ← ∅
while |V'_d| < W ∧ |V'| do
    (v_0, s_0) ← arg max_{(v,s)} X*(v, s)
    X(v_0, s_0) ← 1
    X(v_0, s) ← 0 ∀s ≠ s_0
    if There is overloaded server s_o then
        S' ← S' \ s_o
    else
        V'_d ← V'_d ∪ v_0
    end if
    X*(v_0, s) ← 0 ∀s //Remove row v_0 from X*
end while
Return X, V'_d, S'.
```

TABLE III: Amazon EC2 Instances.

Instance Type	CPU (#core)	Memory (GB)	HDD (GB)
m3.medium	1	3.75	4
m3.large	2	7.5	32
m3.xlarge	4	15	40
m3.2xlarge	8	30	80

the set of decision made VMs and the corresponding VM will be reconsidered in successive optimization iterations. The decision process will iterate up to W times. Once the optimization process finishes, the solution will be applied to place new VM requests and/or migrate current VMs onto selected servers. The redundant servers, i.e., have no VMs placed on, can be turned off for energy saving.

Complexity Analysis: JPM is comprised of M/W iterations. At the iteration i^{th}, JPM solves the relaxed optimization problem of $N(M - iW)$ variables. Hence the complexity of JPM is $O(\frac{M}{W}f(MN))$ where $f(MN)$ is the complexity of the algorithm used for solving the relaxed optimization problem of variable $X \in [0,1]^{M\times N}$.

C. Multi-level Joint VM Placement and Migration (MJPM) Algorithm

A data center typically has thousands of servers and may scale to even million servers in the future [4] [3]. A centralized algorithm is definitely not viable as it can not be managed by any single machine. It is also known that the VM interconnection graphs are very sparse with VMs are strongly connected with each other within groups [33]. Particularly, VMs belonging to the same service/application usually communicate more often with each other than VMs of different service types. In other words, VMs belonging to different services/applications normally have weak or zero communication.

With that observation, we propose a multi-level join VM placement and migration algorithm (MJPM). In this approach, the VM graphs will firstly be partitioned into smaller communities by community detection algorithm. Some of the algorithms used for identifying community structures of a network have been proposed [34]. We illustrate the operation of MJPM in Fig. 4. In this example, PMs are statically grouped into cells based on physical relationship. Due to the very high sparsity of VM graphs, the first phase to identify community structures could be done in linear time [35] [36]. Next, subset sum algorithms [37] are then employed to group communities into clusters whose sizes are approximately equal. Identified VM community clusters will be mapped to PM cells based on current state of the data center. Finally, JPM will be applied to solve the placement and migration problem for each VM community in parallel.

VI. Performance Evaluation and Discussion

In this section, we show the performance of MJPM compared to other methods. Specifically, we compare MJPM using nuclear norm and (weakly) attractive force with a random method which place VMs onto server randomly, the FFD-like algorithm and round-robin method. We modify FFD a little bit by ordering the servers first and try placing VMs onto acceptable hosts in decreasing order of available resources. By doing this we would like to maximize the servers' resource utilization in the hope to reduce the power consumption. We also apply Random method and FFD-like for newly arrival VM requests rather than running from scratch.

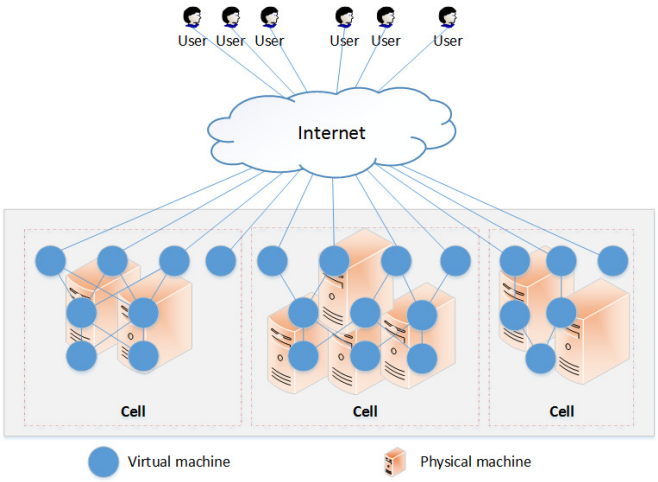


Fig. 4: An abstraction of multi-level joint VM placement and migration algorithms

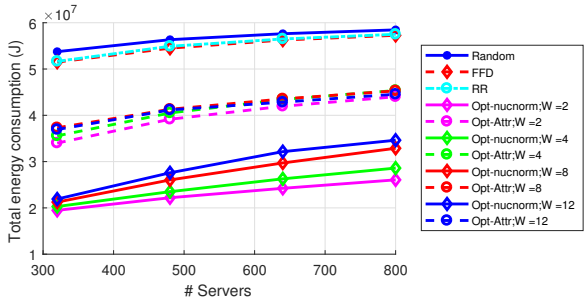
A. Simulation Setup

We simulate a cloud data center with VMs requests being chosen with typical configuration as in Amazon EC2 [38]. Table III lists some parameters of VM instances used in our simulation. We generate $M = 5120$ VMs of random sizes and used instance parameters as VM's resource request vector. Initially, a portion of VMs, e.g., 70%, are placed onto physical hosts randomly as the initial placement scheme X_0 (the current snapshot of the data center). The remaining VMs are considered as newly arrival VM requests. The capacities of physical hosts are equal and resource capacities are chosen such that the total capacity of cloud center is at least double the demanded capacity of all VMs. That means, all VMs will be placed successfully on a physical host. Consequently, physical server capacity is inversely proportional to the number of physical servers in our simulated data centers. We assume that the data center is consisted of 40 server cells. Without loss of generality, we assume that the simulated data center has no maintaince schedule during our simulations, i.e., $L = 0^T$. Finally, we generate a random adjacent matrix A of size $M \times M$ representing the communication load among VMs based on the Barabasis algorithm [39]. Communication weight is uniformly selected between 0Mbps and 75Mbps. All simulations are run in Matlab on a PC[3], and CVX package [32] is used in solving the relaxed convex optimization.

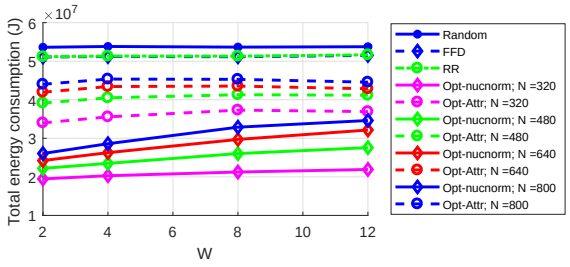
B. Performance Evaluation

We evaluate the performance of algorithms in terms of following metrics: Total communication load; Migration cost;

[3]Intel Core 2 Quad Q8200 processor 2.33 GHz, 4 MB L2 cache; 8GB RAM).



(a)



(b)

Fig. 5: Total energy consumption versus (a) number of servers and (b) window size W's

Power consumption; Time elapsed; and Rank of the final decision matrix. In our simulation we set power consumption of server at idle mode as 480 W[4]. Power consumed by VMs is set to 10 W, 1 W, and 0.5 W per virtual core, 1 GB memory and 1 GB of HDD respectively. Energy needed to transmit 1 GB of data between two physical hosts is set to 512 J. VM migration energy consumption is chosen at 0 J [5], 250 J, and 250 J per CPU core, 1GB of RAM and 1 GB of HDD, respectively [9].[6]

C. Simulation Results and Discussion

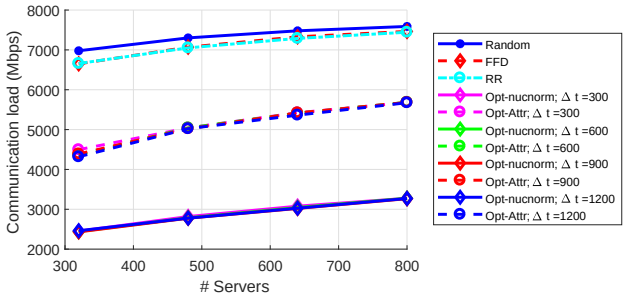
Fig. 5 shows total energy consumption in different views. In general, MJPM (using either nuclear norm or attractive force method) outperforms conventional algorithm in term of saving energy consumed by data centers. MJPM using nuclear norm achieves higher performance compared to MJPM using attractive force. However, by adjusting coefficient in (8), MJPM using attractive force can approach to the performance of MJPM using nuclear norm.

Figures 6 and 7 show the power consumption of data center versus different optimization period Δt and window size W, respectively. In our simulations, we use different cloud center configurations by varying the number of physical hosts. As shown in Fig. 6(a), the network traffic significantly decreases by using the proposed MJPM compared with the existing approaches. The migration costs are depicted in Fig. 6(b). As we can see, the existing methods which just solve VM placement problem only do not consume energy on migrating

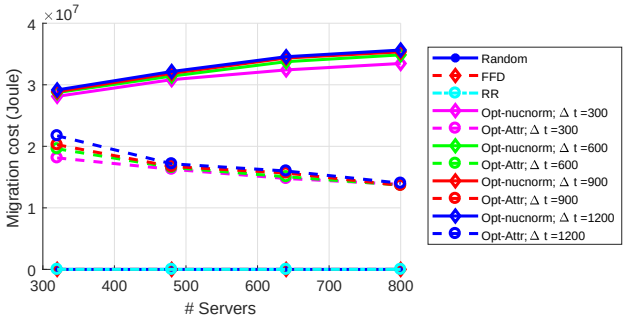
[4]Without loss of generality, we choose this value based on the evarage power consumption per server class as listed in [40].

[5]CPU state is not migrated.

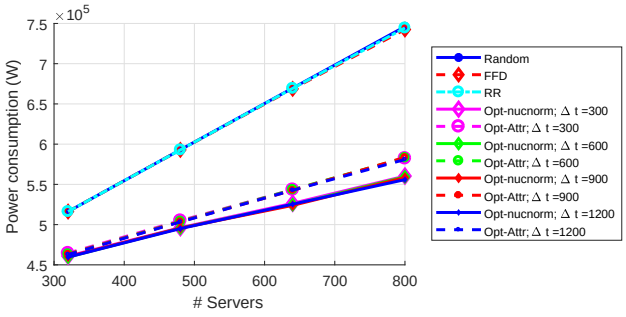
[6]We assume that applications utilizes RAM and HDD maximally.



(a)



(b)

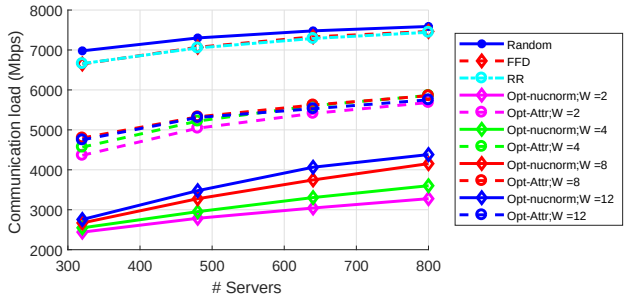


(c)

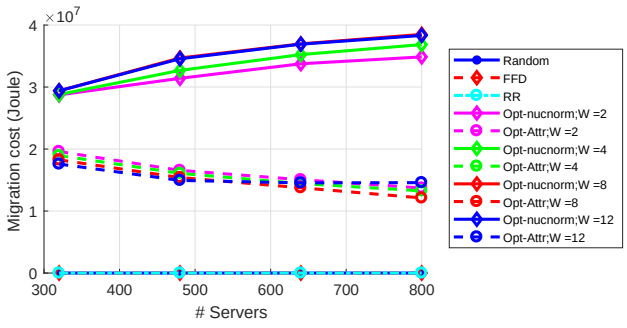
Fig. 6: Performance comparison with different Δt vs. number of servers: (a) communication load, (b) migration cost, and (c) power consumption

VMs. This is because once a VM is placed onto a host, it will stay there until the VM is terminated. On the other hand, the proposed MJPM jointly solves both placement and migration problems, and as expected, it consumes energy on migrating VMs from one physical server to another. The amount of energy used for migration depends on initial assignment scheme as well as data center configurations. Fig. 6(c) shows the total power consumption of all physical servers in data centers. As expected, the proposed MJPM results in the lowest total power consumption despite of additional energy for migration. In our simulations, the overall capacity of the data center is chosen to stay the same; thus, capacity of each server decreases when we increase the number of servers. As a result, more physical servers are needed to serve the same number of VMs. As expected, the total power consumption increases linearly with the number of servers for all approaches as shown in Fig. 6(c). The same observations of power consumption versus W can also be seen in Fig. 7 where our proposed scheme significantly

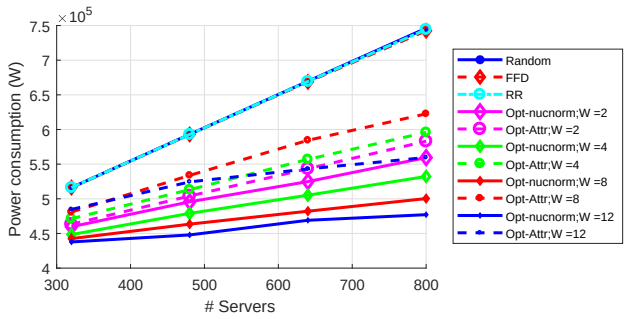
reduces the total power consumption.



(a)



(b)



(c)

Fig. 7: Performance comparison of different schemes with different W vs. number of servers: (a) communication load, (b) migration cost, and (c) power consumption

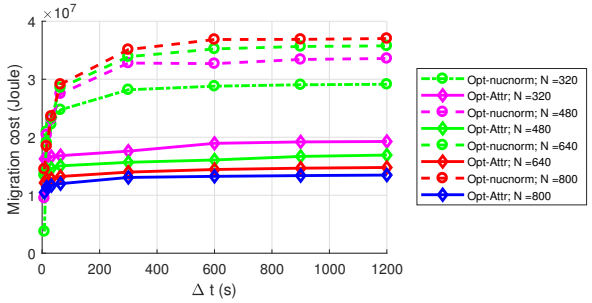


Fig. 8: Migration cost of different schemes vs. Δt

Fig. 8 shows the relationship between migration cost of MJPM with the optimization period Δt. As we can see, when value of Δt is small, the migration cost is small as well. This is because the energy saved on inter-server communication is much smaller compared to the energy needed to migrate VMs. Thus, in the case of small Δt, e.g., less than 200 (s), MJPMs only reassigns a few small-size VMs. On the contrary, when Δt increases, MJPM considers larger number of VMs with bigger size. This is because the energy saved on communication will now surpass the energy needed for migration. We also observe that MJPM using nuclear norm migrates more VMs resulting in fewer number of running PMs. As a result, MJPM requires the least energy consumption compared with other approaches as shown in Fig. 5. Interestingly, we also observe that the migration cost will stable when value of Δt is larger a threshold, e.g., 600 (s). Our intuition is that when the value of Δt is sufficiently large, MJPM algorithms can achieve the optimal solution by identifying and migrating all VMs to appropriate PMs. As a result, the system achieves a minimum inter-server communication. Thus, when Δt increases larger than this threshold, the migration cost will be stable as no more VMs need to be migrated.

Fig. 9 shows the comparison of the ranks of assignment matrices X's (solutions) of different methods. In our calculation, $rank(X)$ is the total number of physical servers needed by assignment schemes produced by corresponding algorithms. Conventional algorithm always uses all available servers. On the other hand, MJPMs tries to minimize the total number of physical servers needed to assign all VMs. It also migrates VMs from one physical host to another in order to create more idle hosts.

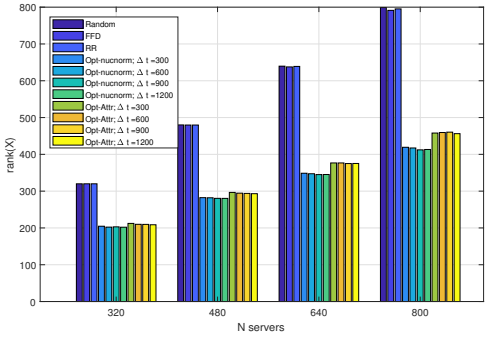


Fig. 9: Matrix rank of different Δt vs. number of servers

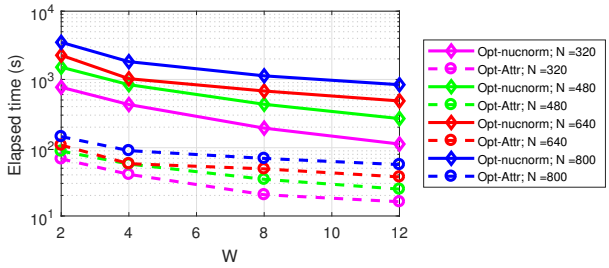


Fig. 10: Elapsed time comparison using different window size W

Finally, Fig. 10 shows the comparison in total time elapsed by the simulation program for each method. Apparently, FFD, RR, and random take constant time to making decision on VM

TABLE IV: Large data center simulation setup.

#VMs	#Servers	#Cells
3840	360	30
5120	480	40
6400	600	50
7680	720	60
10240	960	80
20480	1920	160
40960	3840	320

assignment so they are excluded from the graph. In general, when window size W is increased, the total elapsed time is decreasing. This can be explained by the decreasing in the number of optimization iterations when W increases , i.e., more VMs are made decision per iteration.

Moreover, MJPM using nuclear norm which produce the best solutions takes longer time to solve optimization problem. Meanwhile, MJPM using attractive force take much less time. This can be explained by the computation complexity of each objective function. Evaluating nuclear norm involves in computing the matrix product $X^T X$ in addition to computing the eigenvalues of the $N \times N$ matrix. However computing edges cut of a graph via $Tr(X^T(D_M - A_M)X)$ is much more efficient. Although MJPM based on nuclear norm optimization produce better results, its high complexity prevent it to be applicable in practical data center. On the other hand, MJPM based on dummy attractive force achieve good performance with lower complexity and could be applied for practically large data centers. In the next section, we will show the scalability of Att-force based MJPM in large data centers.

D. Large Size Data Centers

In order to further show the scalability of the Attr-force based MJPM algorithm [7], we simulate data centers varying in size. Specifically, the total number of VMs, servers and cells are chosen as listed in Table IV. Again, we assume that there is no maintenance plan during simulation period. Fig. 11 shows simulations results for large-size data centers with different VM graph sizes. Although, having migration costs, MJPM still outperforms conventional algorithms in terms of overall power consumption. This can be explained by the placements and migrations decided to consolidate VMs onto less PMs as well as co-locate VMs with strong communication needs on the same or closer PMs.

VII. Conclusion and Future work

In this paper, we proposed two Multi-level Joint VM placement and migration algorithms (MJPMs) for optimizing energy consumption in cloud data centers. We formulated and solved a multi-objective function for optimal solution. Leveraging both theoretical analysis and extensive simulations on different topologies, we show that the proposed MJPM algorithm is effective and efficient in improving data center energy consumption. Although we focused on VM placement and migration problem as the target problem, MJPMs can also be applied in many similar resource allocation problems such as assigning/scheduling tasks or application instances to servers.

For future work, we will explore MJPMs further such as developing an algorithm to determine optimal W's based on data center configurations; develop a more efficient algorithm for the first phase of MJPMs, e.g., detecting communities and flexibly clustering into cells. In addition, we will compare MJPMs with other state-of-the-art approaches. Finally, a parameter selection scheme will be studied and proposed in order to best apply MJPMs in different cloud providers with different data center configurations.

[7] Nuclear norm based MJPM is prohibitively expensive and not applicable for large VM graphs.

References

[1] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, Aug. 2008. [Online]. Available: http://doi.acm.org/10.1145/1402946.1402967

[2] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: A high performance, server-centric network architecture for modular data centers," in *SIGCOMM*, 2009.

[3] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: A scalable and flexible data center network," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 51–62, Aug. 2009. [Online]. Available: http://doi.acm.org/10.1145/1594977.1592576

[4] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: A scalable and fault-tolerant network structure for data centers," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 75–86, Aug. 2008. [Online]. Available: http://doi.acm.org/10.1145/1402946.1402968

[5] R. Katz, "Network-attached storage systems," in *Scalable High Performance Computing Conference, 1992. SHPCC-92, Proceedings.*, April 1992, pp. 68–75.

[6] A. T. Jon Tate, Rajani Kanth, "Introduction to storage area networks (SANs)," *IBM Redbooks*.

[7] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao, "Energy-aware server provisioning and load dispatching for connection-intensive internet services," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, ser. NSDI'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 337–350. [Online]. Available: http://dl.acm.org/citation.cfm?id=1387589.1387613

[8] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2*, ser. NSDI'05. Berkeley, CA, USA: USENIX Association, 2005, pp. 273–286. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251203.1251223

[9] H. Liu, C.-Z. Xu, H. Jin, J. Gong, and X. Liao, "Performance and energy modeling for live migration of virtual machines," in *Proceedings of the 20th International Symposium on High Performance Distributed Computing*, ser. HPDC '11. New York, NY, USA: ACM, 2011, pp. 171–182. [Online]. Available: http://doi.acm.org/10.1145/1996130.1996154

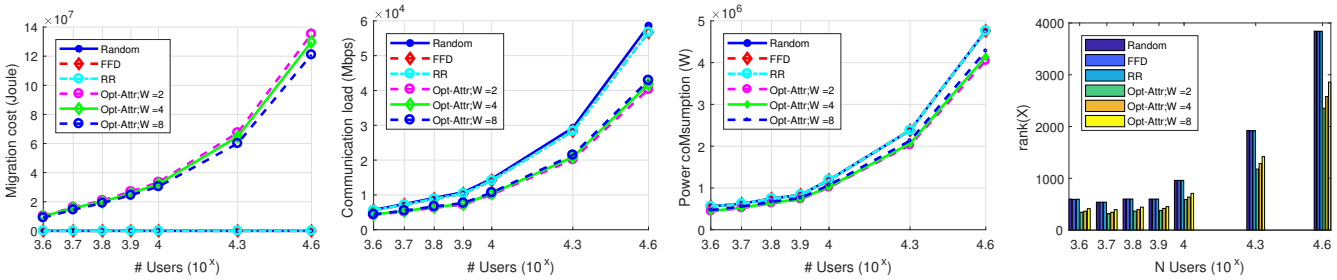
[10] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and gray-box strategies for virtual machine migration," in *Proceedings of the 4th USENIX Conference on Networked Systems Design & Implementation*, ser. NSDI'07. Berkeley, CA, USA: USENIX Association, 2007, pp. 17–17. [Online]. Available: http://dl.acm.org/citation.cfm?id=1973430.1973447

[11] V. V. Vazirani, *Approximation algorithms*. springer, 2001.

[12] K. Li, H. Zheng, and J. Wu, "Migration-based virtual machine placement in cloud systems," in *2013 IEEE 2nd International Conference on Cloud Networking (CloudNet)*, Nov 2013, pp. 83–90.

[13] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1107–1117, June 2013.

[14] X. Zheng and Y. Cai, "Dynamic virtual machine placement for cloud computing environments," in *2014 43rd International Conference on Parallel Processing Workshops*, Sept 2014, pp. 121–128.



(a) (b) (c) (d)

Fig. 11: Simulation results with large number of VMs

[15] F. H. Tseng, X. Wang, L. D. Chou, H. C. Chao, and V. C. M. Leung, "Dynamic resource prediction and allocation for cloud data center using the multiobjective genetic algorithm," *IEEE Systems Journal*, vol. PP, no. 99, pp. 1–12, 2017.

[16] A. Singh, M. Korupolu, and D. Mohapatra, "Server-storage virtualization: Integration and load balancing in data centers," in *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, ser. SC '08. Piscataway, NJ, USA: IEEE Press, 2008, pp. 53:1–53:12. [Online]. Available: http://dl.acm.org/citation.cfm?id=1413370.1413424

[17] G. Khanna, K. Beaty, G. Kar, and A. Kochut, "Application performance management in virtualized server environments," in *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, April 2006, pp. 373–381. [Online]. Available: http://dx.doi.org/10.1109/NOMS.2006.1687567

[18] B. Speitkamp and M. Bichler, "A mathematical programming approach for server consolidation problems in virtualized data centers," *Services Computing, IEEE Transactions on*, vol. 3, no. 4, pp. 266–278, Oct 2010.

[19] H. Jin, D. Pan, J. Xu, and N. Pissinou, "Efficient vm placement with multiple deterministic and stochastic resources in data centers," in *Global Communications Conference (GLOBECOM), 2012 IEEE*, Dec 2012, pp. 2505–2510.

[20] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: fair allocation of multiple resource types," vol. 11, 2011, pp. 24–24.

[21] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *INFOCOM, 2010 Proceedings IEEE*, 2010, pp. 1–9. [Online]. Available: http://dx.doi.org/10.1109/INFCOM.2010.5461930

[22] Z. Huang and D. H. K. Tsang, "M-convex vm consolidation: Towards a better vm workload consolidation," *IEEE Transactions on Cloud Computing*, vol. 4, no. 4, pp. 415–428, Oct 2016.

[23] J. Xu and J. A. B. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, Dec 2010, pp. 179–188. [Online]. Available: http://dx.doi.org/10.1109/GreenCom-CPSCom.2010.137

[24] D. A. Tran and T. Zhang, "S-put: An ea-based framework for socially aware data partitioning," *Computer Networks*, vol. 75, Part B, pp. 504 – 518, 2014, special Issue on Online Social NetworksThe Connectedness, Pervasiveness and Ubiquity of Online Social Networks. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128614003296

[25] X. Dai, M. Wang, and B. Bensaou, "Energy-efficient virtual machines scheduling in multi-tenant data centers," *Cloud Computing, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.

[26] F. H. Tseng, Y. M. Jheng, L. D. Chou, H. C. Chao, and V. C. M. Leung, "Link-aware virtual machine placement for cloud services based on service-oriented architecture," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2017.

[27] T. Duong-Ba, T. Nguyen, B. Bose, and T. Tran, "Joint virtual machine placement and migration scheme for datacenters," in *Global Communications Conference (GLOBECOM), 2014 IEEE*, Dec 2014, pp. 2320–2325.

[28] S. Filiposka, A. Mishev, and C. Juiz, "Community-based vm placement framework," *J. Supercomput.*, vol. 71, no. 12, pp. 4504–4528, Dec. 2015. [Online]. Available: https://doi.org/10.1007/s11227-015-1546-1

[29] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM review*, vol. 52, no. 3, pp. 471–501, 2010.

[30] "Bin-packing," in *Combinatorial Optimization*, ser. Algorithms and Combinatorics 21. Springer Berlin Heidelberg, 2006, vol. 21, pp. 426–441. [Online]. Available: http://dx.doi.org/10.1007/3-540-29297-7-18

[31] J. Kelner, "Online lecture notes for course - 18.409 algorithmist's toolkit," Jul 2012, accessed on 05.11.2017. [Online]. Available: http://ocw.mit.edu/courses/mathematics/18-409-topics-in-theoretical-computer-science-an-algorithmists-toolkit-fall-2009/lecture-notes

[32] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," Mar. 2014, accessed on 10.10.2017. [Online]. Available: http://cvxr.com/cvx

[33] J. Tordsson, R. S. Montero, R. Moreno-Vozmediano, and I. M. Llorente, "Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers," *Future Generation Computer Systems*, vol. 28, no. 2, pp. 358 – 367, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X11001373

[34] M. Plantié and M. Crampes, *Survey on Social Community Detection*. London: Springer London, 2013, pp. 65–85. [Online]. Available: https://doi.org/10.1007/978-1-4471-4555-4-4

[35] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Phys. Rev. E*, vol. 70, p. 066111, Dec 2004. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.70.066111

[36] A. Hollocou, J. Maudet, T. Bonald, and M. Lelarge, "A linear streaming algorithm for community detection in very large networks," *CoRR*, vol. abs/1703.02955, 2017. [Online]. Available: http://arxiv.org/abs/1703.02955

[37] S. Martello and P. Toth, *4 Subset-sum problem*. New York, NY, USA: John Wiley & Sons, Inc., 1990.

[38] Amazon, "Amazon elastic compute cloud," Feb 2014, accessed on 10.10.2017. [Online]. Available: http://awsdocs.s3.amazonaws.com/EC2/latest/ec2-ug.pdf

[39] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," *SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 4, pp. 251–262, Aug. 1999. [Online]. Available: http://doi.acm.org/10.1145/316194.316229

[40] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Y. Zomaya, "A taxonomy and survey of energy-efficient data centers and cloud computing systems," *CoRR*, vol. abs/1007.0066, 2010. [Online]. Available: http://arxiv.org/abs/1007.0066

Thuan Duong-Ba received his B.S and M.S degrees in Electronics and Telecommunications from Hanoi University of Science and Technology (HUST), Vietnam, in 2002 and 2005 respectively. From 2006 to 2009 he joined HUST as a lecturer. He obtained a PhD degree in Electrical and Computer Engineering at Oregon State University in 2014. He is currently a Software Engineer at Amazon Web Service. His research interests include computer networks, network coding, random optimization, and distributed systems.

Tuan Tran received the B.S. degree from Hanoi University of Science and Technology, Viet Nam and the Ph.D. degree from Oregon State University, USA in 2000 and 2010, respectively, all in computer engineering. Currently, he is an Assistant Professor with the College of Computer and Information Technology, Sullivan University, Louisville, USA. Prior to that, he was a Postdoctoral Scholar with Arizona State University and University of Louisville from 2010 to 2012. He was a recipient of the best paper runner-up award at the IEEE International Conference on Computer Communication Networks (ICCCN) in 2010 and the Jack Neubauer Memorial Award for the Best Systems Paper published in the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY in 2012. His research interests include machine learning for computer networks, stochastic system modeling, and network coding. .

Thinh Nguyen received the B.S. degree the University of Washington, Seattle, WA, USA, in 1995 and the Ph.D. degree from the University of California, Berkeley, CA, USA, in 2003. He is currently a Professor with the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR, USA. He is interested in theories and applications of all things stochastic. His current and past research projects span a number of application areas ranging from signal processing and communication, to distributed systems and quantum computing. Dr. Nguyen has served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY and the IEEE TRANSACTIONS ON MULTIMEDIA.

Bella Bose received the B.E. degree in electrical engineering from Madras University, Madras, India in 1973, the M.E. degree in electrical engineering from Indian Institute of Science, Bangalore, in 1975, and the M.S. and Ph.D. degrees in computer science and engineering from Southern Methodist University, Dallas, TX, in 1979 and 1980, respectively. Since 1980, he has been with Oregon State University, Corvallis, Oregon, where he is a Professor and the Associate Director for the School of EECS. His current research interests include error control codes, fault-tolerant computing, parallel processing, and computer networks. Bose is a Fellow of both ACM and IEEE.