CACEV: a Cost and Carbon Emission-Efficient Virtual Machine Placement Method for Green Distributed Clouds

This paper was published in: *IEEE 13th International Conference on Services Computing*, pages 275-282, 2016.

Ehsan Ahvar[1], Shohreh Ahvar[1,3], Zoltán Ádám Mann[2], Noel Crespi[1], Joaquin Garcia-Alfaro[1] and Roch Glitho[3]
[1]Institut Mines-Telecom, Telecom SudParis, France.
Emails: {ehsan.ahvar,shohreh.ahvar,noel.crespi,joaquin.garcia_alfaro}@telecom-sudparis.eu
[2]University of Duisburg-Essen, Germany. Email: zoltan.mann@gmail.com
[3]Concordia University, Canada. Email: glitho@ciise.concordia.ca

***Abstract*—Distributed clouds have recently attracted many cloud providers and researchers as a topic of intensive interest. High energy costs and carbon emissions are two significant problems in distributed clouds. Due to the geographic distribution of data centers (DCs), there are a variety of resources, energy prices and carbon emission rates to consider in a distributed cloud, which makes the placement of virtual machines (VMs) for cost and carbon efficiency even more critical than in centralized clouds. Most previous work in this field investigated either optimizing cost without considering the amount of produced carbon or vice versa. This paper presents a cost and carbon emission-efficient VM placement method (CACEV) in distributed clouds. CACEV considers geographically varying energy prices and carbon emission rates as well as optimizing both network and server resources at the same time. By combining prediction-based A* algorithm with Fuzzy Sets technique, CACEV makes an intelligent decision to optimize cost and carbon emission for providers. Simulation results show the applicability and performance of CACEV.**

I. Introduction

Recent cloud infrastructures are increasingly geographically distributed. The distributed cloud has some benefits in comparison to the centralized cloud such as lower upfront investment, vulnerability to natural disasters and proximity to users. Due to the geographical distribution of data centers (DCs), distributed clouds offer a pool of resource options with different prices and carbon emission rates.

Carbon emission and cost are currently two critical concerns for cloud providers and network operators. Energy consumption (EC) has a direct effect on both cost and carbon emission. DCs worldwide consumed 270 TWh of energy in 2012, with a Compound Annual Growth Rate (CAGR) of 4.4% from 2007 to 2012 [1], [2]. According to predictions, they will account for about 8% of worldwide electricity consumption by 2020, and will generate about 2.6% of global carbon emission [3], [4]. Therefore, providers try to reduce their EC and carbon emission. For example, France Telecom-Orange has the ambition of decreasing its EC and carbon emission from 2006 to 2020 by 15 and 20% respectively [5].

As a result, energy saving methods are needed to help providers to reduce both cost and carbon emission. However, as energy prices and carbon emission rates vary by location (e.g., because of different energy sources), energy savings alone may not necessarily or effectively imply reduction of carbon emission and cost. In addition to EC which is important for both cost and carbon efficiency, energy price factor for cost efficiency and cleanness of energy sources for carbon efficiency have to be considered. Above all, there is no correlation between the cleanness (carbon footprint) of a locations energy sources and the energy price for that location [3]. Therefore, to optimize both cost and carbon emission, we need to solve a three-dimensional problem: (i) how to minimize EC (usually in form of resource utilization optimization) (ii) how to find resources with best energy price and (iii) how to choose the resources which are connected to energy sources with lowest carbon emission rate. Considering these three, sometimes conflicting, dimensions simultaneously makes the problem very complex.

The main objective of this paper is to devise a new VM placement algorithm considering the above three-dimensional problem to optimize both cost and carbon emission in a distributed cloud. To do so, we try to allocate newly requested VMs using currently active cloud devices (i.e., physical machines (PMs) and network elements) in order to avoid the EC associated with the activation of devices from sleep mode and also to have the lowest possible number of active devices.

To offer a realistic solution, the paper considers a detailed system model characterized by the following points:

1) heterogeneity of resources (DCs, PMs, switches) and VMs
2) heterogeneity of EC models of cloud elements (e.g., different PMs may have different EC models)
3) effects of workload on EC of devices (e.g., EC of a PM depends on its CPU load)
4) multiple paths between a pair of PMs
5) a more intelligent algorithm, in contrast to the greedy heuristics that most existing approaches use [17]
6) joint optimization of PM (server) and network resources
7) considering variety in price and location among the resources
8) ability to select resources from multiple DCs to serve a

request
9) taking into account inter-VM communication
10) IaaS Service Level Agreement (SLA) violation

To address these points, this paper proposes a cost and carbon emission-efficient VM placement method (CACEV) for distributed cloud environments. CACEV builds on ideas of integrating the prediction-based A* search algorithm [18] with Fuzzy Sets technique to obtain better results than typical greedy heuristics. It is a VM placement algorithm for joint optimization of servers and network, which also considers price, location and carbon emission rate of resources. It can select multiple DCs for the reasons of cost and carbon emission efficiency or capacity limitations. Network and data transfer awareness in selecting DCs/PMs and placing VMs makes CACEV capable to optimize network traffic as well.

CACEV is designed for allocating batch jobs, also supporting applications with inter-VM communication. To prevent SLA violations, CACEV adopts utilization thresholds (for more information, see [21]). The goal is to preserve free resources to prevent SLA violations due to consolidation in situations where resource requirements of VMs increase. When selecting PMs, CACEV makes sure not to violate the upper threshold.

The rest of the paper is organized as follows: Section II describes related work. Section III introduces our system model and Section IV defines the problem. The proposed algorithm is described in Section V. Section VI evaluates CACEV and Section VII concludes the paper.

II. Related work

Previous work addressed some of the 10 points listed in the Introduction to characterize the problem, but only in isolation. To our knowledge, our work is the first to address all aspects in combination.

Li et al. [10] and Dong et al. [11] considered server and network resources at the same time for their proposed VM placement algorithms. You et al. [12] designed a network-aware VM placement method to improve communication cost. But all these works are limited to a single-DC environment and are not appropriate for a distributed cloud with varying resource prices. Alicherry et al. [13] proposed algorithms for network-aware selection of DCs and allocation of VMs on PMs in a distributed cloud. But their primary objective is to minimize the maximum latency in communication between the VMs allocated for a user request, which is different from our objective. They also did not consider hardware and VM heterogeneity in their solution. Our previous study [14] focused on improving communication cost between DCs in a distributed clouds. But, unlike this paper, it did not consider server cost, nor VM placement inside DCs. In addition, our current work is different from all mentioned results since we consider not only costs but also carbon emission.

Khosravi et al. [15] addressed both energy and carbon emission during VM placement in a distributed cloud. However, they did not consider the variability of energy prices. Also, inter-VM communication was not considered, although it is an important factor for reducing traffic and energy of network resources. Zhou et al. jointly considered SLA, electricity cost, and emission reduction for distributed DCs [4] and, recently, Gu et al. proposed a method to minimize carbon emission of cloud DCs while satisfying constraints on response time, electricity budget and maximum number of running PMs in an environment with homogeneous PMs [16]. Different from our work which is for batch jobs with constraints on inter-VM communication, these papers target service jobs with constraints on response time.

III. System model

We consider a hierarchical distributed cloud architecture [6] including a cloud controller and site (i.e., DC) and PM controllers. The DCs are given by a complete graph $G1 = (Dc, E1, wDc, wE1)$ where Dc is a set of DCs, wDc denotes their current capacity, $E1$ consists of connected edges (representing network paths), and $wE1$ denotes the edge weights (e.g., number of routers on the network path). Each DC has its own Power Usage Effectiveness (PUE) value and is associated with one or more energy sources with different carbon footprint rates and energy prices. PUE is defined as the ratio of total power consumed by a DC to the power consumed by IT devices [15]. We assume the cloud provider owns or leases a high-capacity backbone network to carry the traffic between DCs. Inside DCs, our model (and our proposed VM placement algorithm) supports both structured (e.g., Fat-Tree [7]) and arbitrary [8] topologies.

Each DC$_j$ ($1 \le j \le |Dc|$) is represented by a weighted graph $G2_j = (N_j, E2_j, wN_j, wE2_j)$ where N_j is the set of n_j PMs and wN_j shows their current capacity. Similar to [9], for every pair of PMs x and y in DC$_j$, a set of pre-calculated paths from PM x to PM y is considered. $E2_j$ consists of the set of calculated paths between pairs of PMs in DC$_j$. Resource parameters of each PM_i are given as a vector $wE2_i$, including CPU, memory, disk, and I/O bandwidth. Each basic resource unit is represented by one slot [10]. We consider sleep and active modes for both PMs [26] and switches [24], [25].

The model supports different types of VMs. Each VM type k ($1 \le k \le U$) is specified by a vector Vc^k of requested resources, including CPU, memory, disk, and I/O bandwidth in unit of slots.

To handle time-varying request rates and energy prices, time is split into equal time windows (T). A set A including r requests is received in each time slot T [19]. We assume that within a time slot T the energy price is fixed, and the IaaS provider rents VMs based on unit of the time slots (even if a VM is finished in the middle of T, the rental fee should be paid for the whole time window).

Each request R_i ($1 \le i \le r$) demands m_i VMs where $M = \sum_1^r R_i$. A request usually requires multiple VMs which need to communicate to each other. A traffic matrix $TR_{M.M}$ contains VM relations and/or the amount of traffic exchanged among the M VMs. $TR_{M.M}$ can be created either from traffic information attached to the requests coming from PaaS or normal users, or created based on estimation techniques. Operating cost and carbon emission are related to the amount of EC by server and network resources. EC of a PM is considered as a function of its CPU, since the CPU is the major component of power consumption in a PM [2], [26], [23]. Routers and switches are the main contributors to network EC [20].

IV. Problem formulation

An IaaS cloud controller receives a set A of requests (i.e., tasks or applications) in a time slot T. The cloud controller has to select appropriate DCs and distributes VM requests to the selected DCs. The distributed requests in each selected DC are then allocated on appropriate PMs. Moreover, appropriate paths are selected between related PMs. All symbols in this section and their definitions can be found in Table I.

The objective is to select a subset of DCs and, then, in each selected DC_j choose a subset of the PMs to accommodate the set A of requests (including M VMs) in a way which leads to optimal overall cost (OverallCost) and carbon emission (CarbonEmission). OverallCost depends on the EC of the selected resources and on its price. Carbon emission can be improved by reducing EC of resources and selecting resources with access to energy sources with low carbon emission (i.e., low carbon emission rate). The Cost and Carbon emission Optimization Problem (CCOP) in distributed clouds is formalized as follows:

$$\begin{cases} \text{minimize } OverallCost + CarbonEmission\text{, where} \\ OverallCost = DcCost + ComCost\text{, and} \\ CarbonEmission = DcEmission + ComEmission \end{cases} \tag{1}$$

with the constraint that the selected DCs must have enough total capacity to accommodate the M VMs:

$$\sum_{j \in Dc} \sum_{i=1}^{n_j} (wN_i \cdot SelDC_j) \geq \sum_{j=1}^{M} Vc_j. \tag{2}$$

A. *Overall cost formulation (OverallCost)*

DcCost in Eq.(1) is the cost of incremental energy of selected DCs (both servers and intra-DC networks) to accommodate the requests. $TSerEn_j$, $TNetEn_j$ and $EnPr_j$ are the incremental server energy, network energy, and the energy price in DC_j.

$$DcCost = \sum_{j=1}^{d} PUE_j \cdot (TSerEn_j + TNetEn_j) \cdot EnPr_j \cdot SelDC_j \tag{3}$$

$$TSerEn_j = \sum_{i=1}^{n_j} SerEn_i \cdot Selt_i \tag{4}$$

$$SerEn_i = \sum_{k=1}^{M} \mathcal{E}^i_{inc,VM_k} \cdot VM_{k,i} \tag{5}$$

Subject to the following constraints:

$$VM_{k,i} \leq Selt_i, \quad \forall i,k \quad 1 \leq i \leq n_j, \quad 1 \leq k \leq M, \tag{6}$$

$$\sum_{i=1}^{n_j} VM_{k,i} = 1, \quad \forall k \quad 1 \leq k \leq M, \tag{7}$$

$$\sum_{k=1}^{M} VM_{k,i} \cdot Vc_k \leq wN_i, \quad \forall_i \quad 1 \leq i \leq n_j. \tag{8}$$

Eq. (6) ensures that a VM can be assigned only to one of the selected PMs. Eq. (7) guarantees that each VM is assigned to exactly one PM and (8) guarantees that the total load of the VMs assigned to a PM does not exceed its capacity. IE of running VM_k on PM_i is computed as follows:

$$\mathcal{E}^i_{inc,VM_k} = (EN^i_{wakeUp} + EN^i_{idle}) \cdot Ser^i_{Slp} + \mathcal{E}^i_{VM_k}. \tag{9}$$

If PM_i is in sleep mode and receives the first VM, it needs to spend energy EN^i_{wakeUp} to go from sleep to active mode. If active but idle, PM_i consumes constant energy of EN^i_{idle}; VM$_k$ adds $\mathcal{E}^i_{VM_k}$ to it. As the first VM lets the PM wake from sleep mode, the resulting EC is $EN^i_{wakeUp} + EN^i_{idle} + \mathcal{E}^i_{VM_k}$. But for VMs added to an already active PM, the increase in energy is only $\mathcal{E}^i_{VM_k}$. To compute $\mathcal{E}^i_{VM_k}$, we use formulas from [26] and [22]:

$$EN^i = EN^i_{idle} + \sum_{k=1}^{M} (\mathcal{E}^i_{VM_k} \cdot VM_{k,i}) \tag{10}$$

$$EN^i = EN^i_{idle} + (EN^i_{max} - EN^i_{idle}) \cdot \omega^i \tag{11}$$

$$\omega^i = \frac{\sum_{k=1}^{M} (CPU_{VM_k} \cdot VM_{k,i})}{CPU^i_{max}} \tag{12}$$

Using (10) and (11) for one VM_k, $\mathcal{E}^i_{VM_k}$ is computed:

$$EN^i_{idle} + \sum_{k=1}^{M} (\mathcal{E}^i_{VM_k} \cdot VM_{k,i}) = EN^i_{idle} + (EN^i_{max} - EN^i_{idle}) \cdot \omega^i \tag{13}$$

$$\mathcal{E}^i_{VM_k} = (EN^i_{max} - EN^i_{idle}) \cdot \frac{CPU_{VM_k}}{CPU^i_{max}}. \tag{14}$$

To calculate TNetEn$_j$, we first compute NetEn$_{ij}$:

$$NetEn_{ij} = \sum_{pkt=1}^{\delta_{ij}} NetEn^{pkt}_{ij}. \tag{15}$$

Here, δ_{ij} is computed based on the characteristics of the allocated VMs on PM_i and PM_j and also TR information:

$$\delta_{ij} = \sum_{q=1}^{M} \sum_{w=1}^{M} VM_{qi} \cdot VM_{wj} \cdot tr_{q,w} \tag{16}$$

$$NetEn^{pkt}_{ij} = \sum_{\phi=1}^{PA_{ij}} \alpha^{pkt}_{ij\phi} \cdot \sum_{B \in \lambda_{ij\phi}} \mathcal{E}^B_{inc,pkt} \tag{17}$$

$$\sum_{\phi=1}^{PA_{ij}} \alpha^{pkt}_{ij\phi} = 1; \quad \forall_{pkt \in \delta_{ij}} \quad i,j \in X \tag{18}$$

Today's DC networks are typically provisioned with redundant links and excessive bandwidth to accommodate peak traffic loads and tolerate link failures, and run well below capacity most of the time [27]. Therefore, in (17), we assume at least one path with enough link capacity between each PM pair.

The incremental energy consumption of a network element B is computed analogously to servers (see Eq. (9)):

$$\mathcal{E}^B_{inc,pkt} = (\mathcal{E}^B_{wakeUp} + \mathcal{E}^B_{idle}) \cdot N^B_{Slp} + \mathcal{E}^B_{pkt} \tag{19}$$

TABLE I
OVERVIEW OF THE USED NOTATION

General parameters and notation			
$d = \mid Dc \mid$	number of DCs	N_j	set of PMs in DC_j
n_j	number of PMs in DC_j	ComCost	inter-DC network costs for running the set A of requests
OverallCost	total cost of running the set A of requests	DcEn	IE for running the set A of requests within DCs
DcCost	costs of running the set A of requests within DCs	IE	Incremental energy
CarbonEmission	carbon emission amount for running A	EC	Energy consumption
Request parameters			
M	number of requested VMs in A	Vc_j	vector of requested resources of VM_j
VM_i	requested VMs, $1 \leq i \leq M$		
Quantities relating to server costs			
wN_i	capacity vector of PM_i	CE$_j$	carbon emission rate (in g/kW) for DC$_j$
$SerEn_i$	IE caused by running VMs of set A on PM_i	$TSerEn_j$	IE caused by servicing the set A of requests in DC$_j$
$Selt_i$	=1 if at least one VM of set A is on PM_i, otherwise 0	VM_{ki}	=1 if VM_k is allocated on PM_i
$\mathcal{E}^i_{VMk}$	EC of running VM_k on PM_i (without considering PM_i mode)	$\mathcal{E}^i_{inc,VM_k}$	IE on PM_i caused by running VM_K
EN^i	EC of PM_i for processing VMs of set A	EN^i_{wakeUp}	energy needed for PM_i to go from sleep to active mode
EN^i_{idle}	EC of PM_i if idle (i.e., active, with zero load)	EN^i_{max}	maximum EC of PM_i (i.e., with full load)
ω^i	percentage of the CPU usage of PM_i	CPU^i_{max}	processing capacity of PM_i (FLOPs/sec)
CPU_{VM_k}	CPU load of VM_k (FLOPs/sec)	Ser^i_{Slp}	=1 if PM_i is in sleep mode, 0 if in active mode
r^k	carbon emission rate of energy source k in DC$_j$	$SelDC_j$	=1 if DC_j is selected, otherwise 0
$EnPr_j$	price of energy for DC_j		
Quantities relating to network costs			
N^B_{Slp}	=1 if network element B is in sleep mode, otherwise 0	ComEn$_a b$	IE of data transfer between DC$_a$ and DC$_b$ for set A
δ_{ij}	number of exchanged packets between PM_i and PM_j for set A	$\mathcal{E}^B_{pkt}$	EC of element B to serve a packet (without considering element modes)
$\mathcal{E}^B_{inc,pkt}$	IE of an element B caused by servicing the packet pkt	$\lambda_{ij\phi}$	number of intermediate elements between PM_i and PM_j on ϕ_{th} path
$NetEn_{ij}$	network EC for data transfer between PM_i and PM_j	TNetEn$_j$	total network EC for set A on selected PMs of DC$_j$
$\alpha^{pkt}_{ij\phi}$	=1 if packet pkt is assigned to the ϕ_{th} path between PM_i and PM_j	$NetEn^{pkt}_{ij}$	IE of network elements between PM_i and PM_j for transferring pkt
R_{ij}	number of exchanged packets between PMs i and j for the set A	$TR(tr)_{M.M}$	traffic matrix for set A (M VMs)
$tr_{q,w}$	amount of traffic (packet numbers) between VM_q and VM_w	$\mathcal{E}^B_p$	needed per-packet processing energy
$\mathcal{E}^B_{S\&F}$	per-byte store and forward energy	L	packet length in bytes
PA_{ij}	number of paths between PM_i and PM_j	λ'_{ab}	number of intermediate elements between DC_a and DC_b
δ'_{ab}	number of exchanged packets between DC_a and DC_b for set A	ComEn$_a b^{pkt}$	IE of network to transfer pkt between DC$_a$ and DC$_b$
$ComPr_{ab}$	energy price of communication between DC$_a$ and DC$_b$	ComCE$_{ab}$	carbon emission of communication between DC$_a$ and DC$_b$

In (19), $\mathcal{E}^B_{pkt}$ is computed as follows [20]:

$$\mathcal{E}^B_{pkt} = \mathcal{E}^B_p + \mathcal{E}^B_{S\&F} L, \tag{20}$$

where $\mathcal{E}^B_p$ and $\mathcal{E}^B_{S\&F}$ are constants for a given switch or router configuration. Finally, the total IE of the network for running A on selected PMs of DC$_j$ is given by:

$$TNetEn_j = \sum_{i \in N} \sum_{\substack{j \in N \\ j \neq i}} NetEn_{ij}. \tag{21}$$

Only the selected PMs will be considered automatically in (21), because in (15), when there is no traffic between PM_i and PM_j (δ_{ij}=0), the related NetEn$_{ij}$ $= 0$ as well.

Inter-DCs communication cost (ComCost) in Eq.(1) includes the incremental energy of the inter-DC network to transfer data for running the set A of requests. $ComEn_{ab}$ is the incremental energy between DC_a and DC_b; $ComPr_{ab}$ is the energy unit price for communication between DC_a and DC_b.

$$ComCost = \sum_{a,b \in Dc} ComEn_{ab} \cdot ComPr_{ab} \cdot SelDC_a \cdot SelDC_b \tag{22}$$

$$ComEn_{ab} = \sum_{pkt=1}^{\delta'_{ab}} ComEn^{pkt}_{ab} = \sum_{pkt=1}^{\delta'_{ab}} \sum_{B \in \lambda'_{ab}} \mathcal{E}^B_{inc,pkt} \tag{23}$$

$$\delta'_{ab} = \sum_{i \in Dc_a} \sum_{j \in Dc_b} \sum_{q=1}^{M} \sum_{w=1}^{M} VM_{qi} \cdot VM_{wj} \cdot tr_{q,w} \tag{24}$$

B. Computing carbon emission

DC carbon emission (DcEmission) in Eq.(1) is the carbon emission amount caused by incremental energy within the selected DCs (servers and intra-DC networks) to run the requests:

$$DcEmission = \sum_{j=1}^{d} PUE_j \cdot (TSerEn_j + TNetEn_j) \cdot CE_j \cdot SelDC_j, \tag{25}$$

where CE_j is the average carbon emission rate (in g/kW) of the energy sources of DC_j. It is computed as follows [4]:

$$CE_j = \frac{\sum_{k=1}^{\ell} CE^k_j \cdot r^k}{\sum_{k=1}^{\ell} CE^k_j}, \tag{26}$$

where CE^k_j and r^k denote the electricity generated by energy source k and its carbon emission rate, respectively.

Inter-DC carbon emission (ComEmission) in Eq.(1) is the amount of incremental carbon emission resulting from data transfer between the selected DCs:

$$ComEmission = \sum_{a,b \in Dc} ComEn_{ab} \cdot ComCE_{ab} \cdot SelDC_a \cdot SelDC_b, \tag{27}$$

where $ComCE_{ab}$ is the average carbon emission rate for communication between DC_a and DC_b.

V. Cost and Carbon Emission-Efficient Virtual Machine Placement (CACEV)

CACEV is a two-stage VM placement algorithm (see Algorithm 1). Stage 1 selects DCs and distributes VMs on them simultaneously (joint DC selection-VM distribution, lines 2-3). Stage 2 chooses PMs in each selected DC and allocates VMs on them simultaneously (joint PM selection-VM placement, lines 4-6). For both stages, CACEV first creates candidate subgraphs and then selects the best subgraph in terms of overall cost and carbon emission. CACEV is structured as follows.

Module 1: VM Mapper (VMM). The VMM module (lines 30-38) receives a candidate v (PM or DC) with its current capacity and a set X of VMs with their traffic information (TR$_{M.M}$) as input. Starting from each $VM_i \in X$, VMM creates one subset. It first starts from VM_1. If the capacity of v is greater than the load of VM_1, VMM adds an element VM$_{new}$ which has the highest traffic with VM_1. If v still has free capacity, a third element (a VM which has highest traffic with the two already selected VMs) is added to the subset. This procedure continues until the capacity of v is exhausted or all elements of X are allocated. After creating a subset starting from VM_1, the VMM module creates the second subset starting from VM_2, etc. Finally, there are $|X|$ subsets of VMs so that any would fit into v. VMM selects the subset with highest inter-VM traffic and maps it to v.

Module 2: Candidate Subgraph Creator (CSC). This module (lines 8-20) receives a weighted graph $G = (V, E, wV, wE)$ and a list of VMs as input and returns $|V|$ subgraphs. The aim of CSC is to determine for each $v_i \in V$ an induced subgraph $G'(v_i)$ with sufficient total capacity and optimized overall cost-carbon emission. $G'(v_i)$ is grown from $\{v_i\}$ as starting point by iteratively adding one vertex a time. The already selected vertices are stored in an array SbG; initially, SbG[i][0]=v_i. In each step, CSC checks whether the selected vertices have sufficient total capacity. If yes, $G'(v_i)$ is finished. Otherwise, the PFBS module is called to select one more vertex for inclusion in SbG, and the cycle continues, until the total capacity of the selected vertices is sufficient. Then, $G'(v_i)$ is the subgraph induced by SbG. This way, a subgraph is created for each vertex v as starting point, yielding altogether $|V|$ candidate subgraphs.

Selecting each vertex v_i as starting point is important because the subgraph formed starting from v_i will often be biased towards vertices in the proximity of v_i; taking the best one of the candidate subsets helps to find a globally optimal subset. In principle, it would also be possible to consider all subgraphs of G with sufficient total capacity. However, the number of all such subgraphs can be exponential, making this approach intractable in practice. In contrast, our method is a faster, polynomial-time heuristic.

Module 3: Prediction and Fuzzy Sets-Based Selector (PFBS). Whenever CSC needs to add a new vertex to the candidate subgraph G' being generated, it calls the PFBS module (lines 21-29). Let SubGr denote the list of already selected vertices in subgraph G' and $V \setminus SubGr$ the vertices still available in G for selection. PFBS receives a graph G, SubGr and a set of VMs as input and returns the most cost/carbon effective vertex $v_i \in V \setminus SubGr$ to be included in the subgraph.

The PFBS mechanism is based on a combination of the A^* algorithm [18] and Fuzzy Sets technique. Selecting the best node only in cost or carbon emission is almost a simple work. But making a *certain decision* to select the best node considering these two, sometimes conflicting, metrics simultaneously is more complex and even sometimes not possible. To this end, we use Fuzzy Sets technique. Fuzzy Sets technique [28] is an effective method for modeling uncertainty and for processing vague or subjective information in mathematical models. It has been utilized to a great variety of real problems. The Fuzzy Sets theory considers membership values which are indicated by a value on the range [0, 1]. Where 0 representing

Algorithm 1: CACEV Algorithm

```
Input : $G1(V1, E1, wV1, wE1)$: a weighted graph of DCs; TotalVM[M]:M
        requested VMs; TR[M][M]:VM traffic matrix;
        G3={$G2^i(V2^i, E2^i, wV2^i, wE2^i)$ : $G2^i$ internal graph of
        $DC_i$,$1 \le i \le |V1|$ }
Output: Selecting appropriate DCs and PMs and Placing requested VMs on them
1  Function mainCACEV($G1, G2, TotalVM$)
2  (sGrDCs,VMsOnDCs)= CSC (G1,TotalVM);
3  SelsGrDC = FBSS(sGrDCs,VMsOnDCs);/*returns the best subgraph of DCs*/
4  foreach $DC \in SelsGrDC$ do
5      (sGrPMs,VMsOnPMs)=CSC($G2^{DC}$,VMsOnDCs[SelsGrDC][DC])/*returns
       subgraphs of PMs of $DC_k$*/
6      SelsGrPM= FBSS(sGrPMs,VMsOnPMs);
7  return(SelsGrDC,SelsGrPM,VMsOnPMs[SelsGrPM])/*selected DCs,PMs,VMs on
   them*/
8  Function CSC(G(V,E,wV,wE),VM[ ])/*returns G subgraphs, their allocated VMs*/
9  Let SbG[$|V|$ ][$|V|$ ]= $\{\}$, SubGrVM[$|V|$ ][$|V|$ ][$|X|$ ]= $\{\}$
10 copy members of array VM[ ] into a set X;
11 for $1 \le i \le |V|$ do
12     Let SbG[$i$][j]=$v_i$, j=0, $X' = \{\}$
13     SubGrVM[$i$][j]← VMM(w$v_i$, $X \setminus X'$, TR);
14     TC=wv$_i$; $TR = \sum_{k=1}^{|X|} size(VM[k])$
15     /*TC:Total Capacity, TR:Total Requirement*/
16     while $TC < TR$ do
17         j ← j+1;
18         $(v_{new}, SubGrVM[v_i])$ = PFBS($G, SubGr[v_i], X \setminus X', TR$);
           /*selects one more vertex and its VMs*/
19         Let SbG[i][j]=$v_{new}$, TC=TC+w$v_{new}$
20 return (SbG,SubGrVM) /*returns candidate subgraphs*/
21 Function PFBS (G(V,E,wV,wE),SbG[ ],X,TR)
22 $c_{min} = \infty$;
23 foreach $v_i \in V \setminus SubGr$ do
24     if $v_i^{cpu} < SLA_{Threshold}$ then
25         SubGrVM[$v_i$][j]← VMM(w$v_i$, X, TR);
26         compute $c(v_i)$ using Equations (28)-(38);
27         if $c(v_i) < c_{min}$ then
28             $c_{min} = c(v_i)$, selected = $v_i$;
29 return (selected,SubGrVM[$v_i$])
30 Function VMM($v$, X, TR)
31 foreach $VM_i \in X$ do
32     $Y = \{\}$
33     Add $VM_i$ to set Y
34     Allocate $VM_i$ on $v$
35     while ($v$ not fulled) or ($Y \neq X$) do
36         Find a $VM_{new} \in X \setminus Y$ with total highest traffic with $Y$ elements
37         Allocate $VM_{new}$ on $v$, add $VM_{new}$ on Y, add Y to sSet
38 return (a subset of sSet with highest inter-VMs traffic)
39 Function FBSS(SbG[ ][ ],SbV[ ][ ])/*returns the best subgraph and its VMs*/
40 $MF_{min} = \infty$, i=0
41 while $SbG[i] \neq null$ do
42     compute $cost(SbG[i])$ using Equation (39);
43     compute $carbon(SbG[i])$ using Equation (40);
44     compute $MF(SbG[i])$ using Equation (41);
45     if $MF(SbG[i]) < MF_{min}$ then
46         MF$_{min}$=MF(SbG[i])
47         selected=SbG[i]
48     i=i+1
49 return selected
```

absolute Falseness and 1 shows absolute Truth. PFBS considers a (fuzzy) set which includes all possible (DC or PM) candidates. The membership function of this set maps each candidate to a membership value (MV) in the range [0, 1]. Inspired by the A^* algorithm, the membership function (for each possible candidate $v_i \in V \setminus SubGr$) combines the costs and carbon emission incurred by selecting the candidate v_i and an estimate of the overall costs and carbon emission that will be incurred in the future if v_i is selected now. The estimation aspect of A^* algorithm helps to consider capacity of each candidate in addition to cost and carbon emission. For each possible candidate $v_i \in V \setminus SubGr$, the A^* function is

$$c(v_i) = g(v_i) + h(v_i). \tag{28}$$

The algorithm selects the candidate with the smallest $c(v_i)$ value or highest membership value of $MV(v_i)$ (i.e., $MV(v_i) = 1 - c(v_i)$). Here, $g(v_i)$ is the incremental overall cost and carbon emission incurred by selecting v_i and includes the incremental network and server costs and carbon emission. $g(v_i)$ will be certainly incurred if v_i is selected.

$$g(v_i) = K1 \cdot K2, \tag{29}$$

where $K1$ and $K2$ are normalized values of cost and carbon emission (in range of [0,1]) respectively.

$$K1 = \frac{SerEn_{v_i} \cdot EnPr_{v_i} + NetEn_{v_i} \cdot NetPr_{v_i}}{SerEn_{max} \cdot EnPr_{max} + NetEn_{max} \cdot NetPr_{max}}, \tag{30}$$

$$K2 = \frac{SerEn_{v_i} \cdot SerCE_{v_i} + NetEn_{v_i} \cdot NetCE_{v_i}}{SerEn_{max} \cdot SerCE_{max} + NetEn_{max} \cdot NetCE_{max}}, \tag{31}$$

where SerEn$_{v_i}$ is the IE of the selected v_i caused by running allocated VMs of the set A on it. EnPr$_{v_i}$ is the current price of energy in location of v_i. NetPr$_{v_i}$ is the energy price for network elements. For intra-DC network, it can be the same as EnPr$_{v_i}$. NetEn$_{v_i}$ is the IE of network elements caused by adding the candidate v_i to the subgraph:

$$NetEn_{v_i} = \sum_{v_j \in SbG} NetEn_{v_i,v_j}, \tag{32}$$

where NetEn$_{v_i,v_j}$ is the IE of transferring data from candidate v_i to already selected vertices. For DC selection, NetEn$_{v_i,v_j}$ is computed based on (23) and for PM selection (15) is used. PFBS calls for each candidate v_i the VMM module to detect allocated VMs on the candidate v_i and, then, computes $\delta_{i,j}$ for PMs based on (16) and for DCs from (24). Recall that (15) also selects the best path between two PMs.

The function $h(v_i)$ is an estimate of the incremental overall cost and carbon emission caused by the further vertices that we have to select later on to accommodate all the M VMs.

$$h(v_i) = K1' \cdot K2', \tag{33}$$

where $K1'$ and $K2'$ are normalized values of the estimated cost and carbon emission (in range of [0,1]) respectively.

$$\frac{NS \cdot SerEn_{avg} \cdot EnPr_{avg} + NE \cdot NetEn_{avg} \cdot NetPr_{avg}}{NS \cdot SerEn_{max} \cdot EnPr_{max} + NE \cdot NetEn_{max} \cdot NetPr_{max}}. \tag{34}$$

$$\frac{NS \cdot SerEn_{avg} \cdot SerCE_{avg} + NE \cdot NetEn_{avg} \cdot NetCE_{avg}}{NS \cdot SerEn_{max} \cdot SerCE_{max} + NE \cdot NetEn_{max} \cdot NetCE_{max}}, \tag{35}$$

where NS and NE are the estimated number of vertices and edges (network paths) that will be added later to the subgraph (in the course of allocating the remaining VMs), SerEn$_{avg}$ is the estimated average and SerEn$_{max}$ the maximum possible IE for a new vertex, NetEn$_{avg}$ is the estimated average and NetEn$_{max}$ the maximum possible IE of the network for the further edges. EnPrice$_{avg}$ and NetPrice$_{avg}$ are the average, EnPrice$_{max}$ and NetPrice$_{max}$ the maximum price of energy for vertices and edges, respectively. To estimate NE, recall that G is a complete graph, so that each new node added to a subgraph with z vertices will add z new edges. After adding v_i to the subgraph with z vertices, it will consist of $z+1$ vertices, so adding further vertices will lead to $z+1, z+2, \ldots$ new edges. Hence, if y further vertices will have to be selected after v_i, we have

$$NE = \sum_{k=z+1}^{(z+1)+(y-1)} k = z \cdot y + \frac{y \cdot (y+1)}{2}. \tag{36}$$

$$y = \frac{M - ((\sum_{w \in Al} F(w)) + F(v_i))}{AvgS}, \tag{37}$$

where M is the total number of VMs needed for the set A of requests, $\sum_{w \in Al} F(w)$ is the number of allocated VMs till now for the user request, $F(v_i)$ is the number of VMs that can be allocated if v_i is chosen next, and $AvgS$ is the average capacity of all vertices.

It remains to estimate NetEn$_{avg}$, the average network EC for the edges that will be added to the subgraph in subsequent steps. One possibility is to use the average network EC among all PMs. This would be a good estimate if we sampled edges randomly. However, our algorithm is biased towards edges of lower energy, so that the overall average may be an overestimate. We can get a more accurate estimation by calculating the average EC of the edges that the algorithm has selected so far, i.e., the edges within a set SubGr that includes the subgraph G' as well as the candidate v_i (say Al'). However, when selecting the second vertex, G' has only one vertex and no edge, so in this case, we use the average network EC between the first vertex and all other vertices.

$$NetEn_{avg} = \begin{cases} \frac{\sum_{v_i \in Al'} \sum_{v_j \in Al', v_j \neq v_i} NetEn(v_i v_j)}{z(z+1)/2} & \text{if } z > 1 \\ \sum_{w \in P} NetEn(sw)/N - 1 & \text{if } z = 1, Al' = \{s\} \end{cases} \tag{38}$$

Putting all the pieces together, we get a fairly good estimate of the overall cost-carbon emission to select candidate v. Based on these estimates, the algorithm can select the best choice.

Module 4: Fuzzy Sets-based Best Subgraph Selector (FBSS). This module receives as input a set of subgraphs (SbG) along with a list of allocated VMs on their vertices (SbV). Subgraph i of this set is denoted by SbG[i]. SbV[i] consists of the allocated VMs of SbG[i]. FBSS computes the overall cost (Eq.(39)) and carbon emission (Eq.(40)) of all subgraphs (lines 42-43) and then selects the most appropriate one in terms of overall cost and carbon emission using Fuzzy Sets technique (lines 44-47).

While selecting the best subgraph based on only cost or carbon emission is easy, finding the best subgraph in both aspects simultaneously is a real challenge. Fuzzy Sets, here, help us to find the best subgraph in both aspects of cost and carbon emission.

$$\sum_{v \in SbG[i]} \sum_{j \in SbV[i]} \mathcal{E}^{v}_{inc,j} \cdot EnPr(v) + \sum_{v,v' \in SbG[i]} Net_{v,v'} \cdot NetPr(v) \tag{39}$$

$$\sum_{v \in SbG[i]} \sum_{j \in SbV[i]} \mathcal{E}^{v}_{inc,j} \cdot CE(v) + \sum_{v,v' \in SbG[i]} Net_{v,v'} \cdot NetCE(v), \tag{40}$$

where $\mathcal{E}_{inc,i}$ is computed based on (9)-(14), $Net_{v,v'}$ for DC subgraphs is computed from (23) and for PM subgraphs from (15)-(18). As we have the list of allocated VMs for each vertex, the number of exchanged packets between two nodes is easily computed for PMs from (16) and for DCs from (24).

After computing OverallCost$_i$ and CarbonEmission$_i$ for a subgraph SbG[i], function $MF(SbG[i])$ is considered

$$\frac{OverallCost_i}{MaxOverallCost} \cdot \frac{CarbonEmission_i}{MaxCarbonEmission}. \tag{41}$$

Finally, the subgraph with lowest $MF(SbG[i])$ or highest membership value $MV(SbG[i])$ is selected ($MV(SbG[i]) = 1 - MF(SbG[i])$).

VI. Performance Evaluation

For our experiments, we used a modified version of the CloudSim simulator [29]. We considered a distributed cloud including 10 DCs. The capacity of the whole distributed cloud was chosen randomly, between 1,000 and 2,000 resource slots, in each run. This total capacity was divided among the DCs (each DC has capacity between 100 and 200 slots). Each PM has available capacity between 10-15 slots. Three different sets of requests with 100, 200, and 300 VMs were considered. The traffic matrix of the VMs was generated randomly.

Based on information from the US Energy Information Administration [31, Table.5.6.A], we consider energy price is in range [4,20] Dollar Cents/kWh and for each DC was randomly selected between 4 and 20. For inter-DC networks the energy price was considered as average (12 Cent/kWh). The PUE value was considered in range [1.56,2.1] based on [15]. We considered six energy sources with different carbon emission rates from [4] (Nuclear:15, Coal:968, Gas:440, Oil:890, Hydro:13.5 and Wind:22.5 g/kWh), and assumed five different combinations with average 100,200,300,400 and 500 g/kWh. We selected one of them randomly for each DC.

The path length for each PM pair inside a DC was randomly chosen from 1 to 8 hops (switches) and for DC pairs from 10 to 20 routers. We used real energy models for routers and switches from [20] and for servers from [30].

We evaluate the performance of CACEV by comparing it against CACEV-Cost, CACEV-Carbon, Random and Greedy resource allocation algorithms. CACEV-Cost is a version of CACEV which only considers cost optimization and CACEV-Carbon only considers carbon footprint optimization. Comparing CACEV to these two special versions can show how CACEV manages to find a trade-off between the two optimization goals. The Random algorithm starts by selecting a vertex (DC or PM) randomly and placing as many VMs as possible in the selected vertex. If not all VMs could be allocated in the selected vertex, then a further vertex is selected, again randomly, to place the remaining VMs. This process is repeated until the requested number of VMs is placed. The Greedy algorithm selects a vertex with maximum free capacity and allocates as many VMs from the request as possible in the selected vertex. If further VMs are necessary, then the Greedy algorithm selects from the remaining vertices again the one with maximum free capacity. This process continues until all VMs are placed [13].

It is worth highlighting that, even though the parameters (e.g. capacity) of the DCs and PMs are set randomly, they remain fixed across the runs of all tested algorithms, to ensure comparability of the results. Because of simulation limitations, each run simulated one hour. However, the result values may seem small on the given scale (1 hour), but it is only to show efficiency of the proposed algorithm. The values can be much larger in real world with a longer time scale. For each test, we report the results as average of 10 runs.

Fig. 1 shows the simulation results (each run simulated one hour, so that the cost and carbon emission values are for one hour). In particular, Fig. 1 (a) and (e) show how CACEV could make a joint cost-carbon emission optimization successfully. CACEV outperforms the Random and Greedy algorithms in both dimensions. It was predictable that CACEV-Cost can get the best cost efficiency. However, the carbon emission of CACEV-Cost is sometimes even worse than that of Random or Greedy. Similarly, CACEV-Carbon is the best in carbon efficiency but performs poorly in cost efficiency. In general, CACEV improves 45-115% in carbon emissions on CACEV-Cost while incurring 20-60% higher costs. In comparison to CACEV-Carbon, CACEV improved total cost by 40-60% while increasing carbon emission only by 10-30%.

As seen in Fig. 1 (i), Greedy always has the least number of selected DCs. Together with Fig. 1(a)-(d), this shows that only reducing the number of selected DCs (and PMs) does not lead to total cost reduction. In Fig. 1 (j), because of the limited simulation time scale, there is no significant difference in EC between the methods. Since there are still big differences in costs and carbon emissions, this shows the importance of taking into account the different energy sources (i.e., with variety of prices and emission rates) of the DCs in a distributed cloud.

VII. Conclusion

This paper addressed the problem of allocating VMs in distributed clouds with the aim of optimizing overall cost and carbon emission together. We have claimed that combining multiple metrics, i.e., joint optimization of network and server resources along with resource prices and carbon emission rate, can significantly optimize overall cost and carbon emission together. We have also claimed a prediction-based A* algorithm can give more sophisticated results than typical greedy heuristics for DC/PM selection, because it also predicts the overall cost and carbon emission that will be incurred by future DC/PM

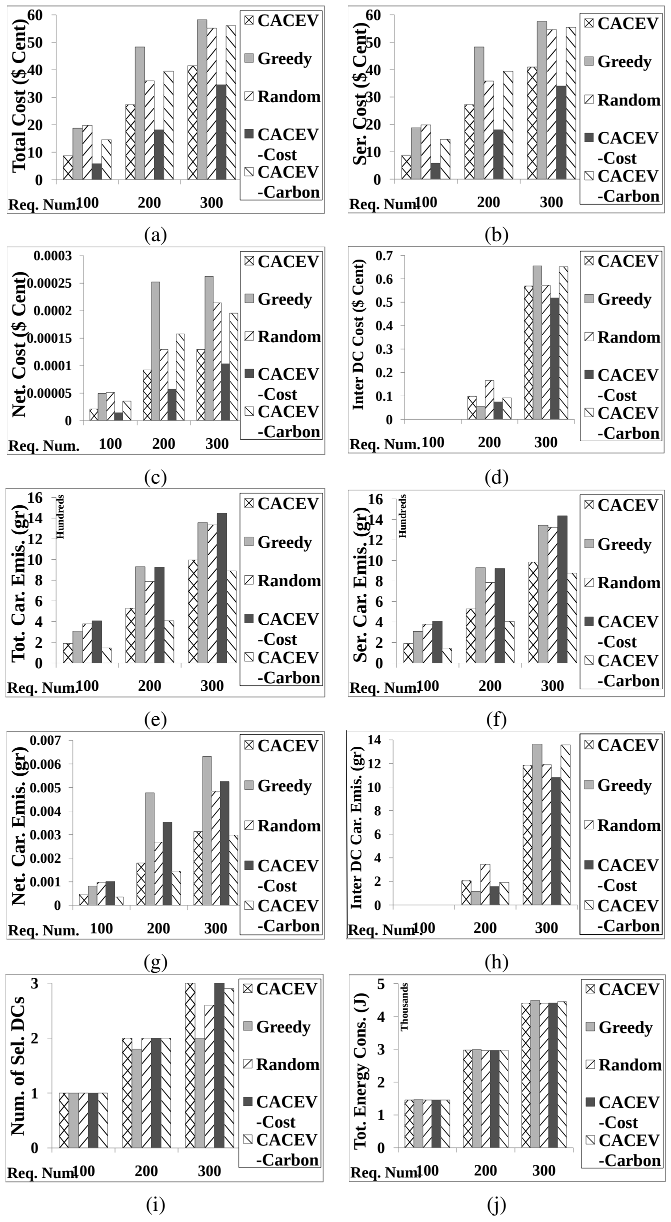

Fig. 1. Simulation results for set of requests with 100, 200 and 300 VMs: (a) total cost, (b) server cost, (c) intra-DC network cost, (d) inter-DC network cost, (e) total carbon footprint, (f) server carbon footprint, (g) intra-DC carbon footprint, (h) inter-DC carbon footprint, (i) number of selected DCs and (j) total energy consumption (EC).

selection and based on the prediction makes more intelligent VM placement decisions. To this end, motivating from the A* algorithm, we have proposed a cost and carbon efficient VM placement method (CACEV). We have also proposed the idea of using Fuzzy Sets to make an appropriate decision in this environment with multiple, sometimes conflicting, metrics. Simulation results prove that CACEV can considerably optimize overall cost and carbon emission in comparison to other algorithms. The results also show that only minimizing the number of used DCs/PMs is not enough to optimize the overall cost and carbon emission.

References

[1] M. Dayarathna, Y. Wen and R. Fan, “Data Center Energy Consumption Modeling: A Survey”, IEEE Communications Surveys & Tutorials, 18(1), 2016.

[2] D. Hatzopoulos, I. Koutsopoulos, G. Koutitas, W. van Heddeghem, “Dynamic Virtual Machine Allocation in Cloud Server Facility Systems with Renewable Energy Sources”, IEEE ICC Conference, Budapest, Hungary, 2013.

[3] P. Xiang Gao, A. R. Curtis, B. Wong, S. Keshav, “Its Not Easy Being Green”, ACM SIGCOMM, Finland, 2012.

[4] Z. Zhou, F. Liu, Y. Xu, R. Zou, H. Xu, J. C. S. Lui and H. Jin, “Carbon-aware Load Balancing for Geo-distributed Cloud Services”, IEEE 21st International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems, San Francisco, CA, 2013.

[5] S. Gosselin, F. Saliou, F. Bourgart, E. Le Rouzic, S. Le Masson, A. Gati, “Energy Consumption of ICT Infrastructures: an Operator’s Viewpoint”, 38th ECOC Conference, Amsterdam, 2012.

[6] M.H. Kabir, G.C. Shoja, S. Ganti, “VM Placement Algorithms for Hierarchical Cloud Infrastructure”, 6th IEEE CloudCom, Singapore, 2014.

[7] M. Al-Fares, A. Loukissas, A. Vahdat, “A scalable, commodity data center network architecture”, ACM SIGCOMM. USA, 2008.

[8] A. Singla, C. Hong, L. Popa, P. Brighten Godfrey, “Jellyfish: Networking Data Centers Randomly”, 9th USENIX conference (NSDI), USA, 2012.

[9] M. Rahnamay-Naeini, S. Sen Baidya, E. Siavashi, and N. Ghani, “A Traffic and Resource-aware Energy-Saving Mechanism in Software Defined Networks”, IEEE ICNC-SIREN, USA, 2016.

[10] X. Li, J. Wu, S. Tang, S. Lu, “Let’s stay together: Towards traffic aware virtual machine placement in data centers”, IEEE INFOCOM. 1842–1850 Toronto, CA, 2014.

[11] J. Dong, X. Jin, H. Wang, Y. Li, P. Zhang, S. Cheng, “Energy-Saving Virtual Machine Placement in Cloud Data Centers”, IEEE/ACM CCGrid. 618–624 Delf, 2013.

[12] K. You, B. Tang, and F. Ding, “Near-optimal virtual machine placement with product traffic pattern in data centers”, IEEE ICC, 3705-3709, 2013.

[13] M. Alicherry, and T.V. Lakshman, “Network aware resource allocation in distributed clouds”, IEEE INFOCOM, 963-971, 2012.

[14] E. Ahvar, S. Ahvar, N. Crespi, J. Garcia-Alfaro, Z.A. Mann, “NACER: a Network-Aware Cost-Efficient Resource allocation method for processing-intensive tasks in distributed clouds”, IEEE NCA, Cambridge, USA, 2015.

[15] A. Khosravi, S. Kumar Garg, and R. Buyya, “Energy and Carbon-Efficient Placement of Virtual Machines in Distributed Cloud Data Centers”, Euro-Par, 2013.

[16] C. Gu, C. Liu, J. Zhang, H. Huang and X. Jia, “Green scheduling for cloud data centers using renewable resources”, IEEE Infocom workshop, Hong Kong, 2015.

[17] Z.A. Mann, “Allocation of virtual machines in cloud data centers – a survey of problem models and optimization algorithms”, ACM Computing Surveys, 48(1), 2015.

[18] S. J. Russell and P. Norvig, “Artificial Intelligence: A Modern Approach”, Prentice Hall, 2010.

[19] Z. Xu, W. Liang, “Minimizing the Operational Cost of Data Centers via Geographical Electricity Price Diversity”, IEEE Conference on Cloud Computing. 99–106 Santa Clara, 2013.

[20] A. Vishwanath, K. Hinton, R.W.A. Ayre, R.S. Tucker, “Modeling Energy Consumption in high-capacity routers and switches”, IEEE Journal on selected areas in communication. 32(8) 1524–1532, 2014.

[21] A. Beloglazov and R. Buyya, “Energy Efficient Allocation of Virtual Machines in Cloud Data Centers”, 10th IEEE/ACM Conference on Cluster, Cloud and Grid Computing, 2010.

[22] G. Warkozek, E. Drayer, V. Debusschere, S. Bacha, “A new approach to model energy consumption of servers in Data Centers”, IEEE Conference on Industrial Technology (ICIT), 211–216 Athens, 2012.

[23] I.S. Moreno, J. Xu, “Customer-Aware Resource Overallocation to Improve Energy-Efficiency in Real-Time Cloud Computing Data Centers”, IEEE Conference on Service-Oriented Computing and Applications. 1–8 Irvine, USA, 2011.

[24] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, N. McKeown, “ElasticTree: Saving Energy in Data Center Networks”, 7th USENIX conference on Networked systems design and implementation, USA, 2010.

[25] N. Vasi, P. Bhurat, D. Novakovic, M. Canini, S. Shekhar, and D. Kosti, “Identifying and Using Energy-Critical Paths”, 7th ACM Conference on Emerging Networking Experiments and Technologies, USA, 2011.

[26] C. Mobius, W. Dargie, A Schill, “Power Consumption Estimation Models for Processors, Virtual Machines, and Servers” IEEE Transactions on Parallel and Distributed Systems. 25(6), 2014.

[27] W. Fang, L. Xiangmin, S. Li, L. Chiaraviglio, N. Xiong, “VMPlanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers”, Computer Networks, 57(1), 179–196, 2013.

[28] L. Zadeh, “Fuzzy sets”, Inform. Control, Vol.8, pp.338-353, 1965.

[29] R.N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. 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) 23-50, 2011.

[30] X. Zhang, J. Lu, X. Qin, “BFEPM:Best Fit Energy Prediction Modeling Based on CPU Utilization”, IEEE Conference on Networking, Architecture and Storage. 41–49, 2013.

[31] US Energy Information Administration. www.eia.gov/electricity/monthly/epm_table_grapher.cfm?t=epmt_5_6_a