VMP^2: Policy-Aware Virtual Machine Placement in Policy Driven Data Centers

Hugo Flores and Bin Tang
Department of Computer Science
California State University Dominguez Hills, Carson, CA 90747, USA
Email: hflores27@toromail.csudh.edu, btang@csudh.edu

***Abstract*—Virtual machine (VM) placement has been a very effective technique in the cloud data center to reduce its network traffic, bandwidth consumption, data access delay as well as energy consumption in the data center network. In this paper we identify and study a new VM placement problem in policy-driven data centers (PDDCs), wherein policies are established that require VM traffic to traverse a sequences of middleboxes (MBs) in order to achieve security and performance guarantee. Although there has been extensive research of VM placement, none of them takes into account aforesaid policies. We refer to the problem as VMP^2: virtual-machine placement in policy-driven data centers. Given a set of hardware-based MBs that have already been deployed in the PDDC, and a data center policy that communicating VM pairs must satisfy, VMP^2 studies how to place the VMs on the physical machines (PMs) in the PDDC in order to minimize the total energy cost of the VM pair communications. Under *ordered policy*, wherein all the VM pairs must traverse the MBs in a specific order, we design a time-efficient polynomial algorithm and prove its optimality. Under *unordered policy*, wherein different VM pairs can traverse the MBs in different orders, we show that VMP^2 is NP-hard thus there does not exist an efficient and optimal solution. We therefore design a 2-approximation algorithm to solve it. In both cases, we compare the algorithms with the state-of-the-art traffic-aware VM placement that does not consider the data center policies, and show via extensive simulations that our algorithms constantly outperform it under different network scenarios. To the best of our knowledge, our work is the first one that addresses the energy-efficient VM placement in policy-driven data centers.**

***Index Terms*—Policy-Driven Data Centers, Virtual Machine Placement, Middleboxes, Energy-Efficiency, Algorithms**

I. Introduction

Policy-Driven Data Centers. Cloud data centers, which consist of hundreds of thousands of physical machines (PMs) that support a wide range of cloud applications such as search engines, social media, and video streaming, have become the dominant computing infrastructure for the IT industry as well as an integral part of our Internet fabric [3], [26]. In recent years, middleboxes (MBs) [5] have been introduced into cloud data centers in order to improve the security and performance of the cloud applications [24], [14], [15]. MBs, also known as "network functions (NFs)", are intermediary network devices that perform functions on network traffic other than packet forwarding. Popular examples of MBs include firewalls, intrusion detection systems (IDSs) and intrusion prevention systems (IPSs), load balancers, and network address translators (NATs). In particular, *data center policies* [12], [23], [8], [27] are established in data centers that demand virtual machine (VM) traffic to traverse a sequence of MBs in order to provide security and performance guarantees to the cloud applications. Fig. 1 shows a simple example of the cloud data center policy, where VM traffic goes through a firewall, a load balancer, and a cache proxy in that order so it filters out malicious traffic and then diverts trusted VM traffic to avoid network congestion, and finally caches the content to share with other cloud users in the data center. Given the ever-increasing demands for security and performance from the diverse cloud user applications, data center policies have become an inseparable part of the Service Level Agreement (SLA) of data centers and an important measurement of their efficiencies. We refer to such cloud data centers as *policy-driven data centers (PDDCs)*.

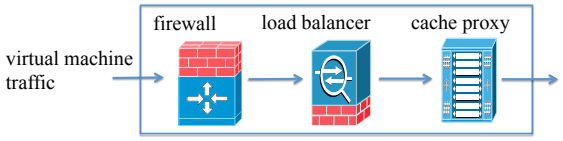


Fig. 1. A Data Center Policy in the PDDC.

Virtual Machine Placement in PDDCs. The current generation of MBs, being specialized systems that supports specific network functions, are mainly proprietary and purpose-built hardware that PDDC operators must deploy and install manually [25]. This not only takes up a significant part of the data center capital and operational expenditure but it is also error-prone. Therefore, once the MBs are physically deployed inside the PDDC, it is not ideal to move them around and to redeploy them. In contrast, with the ubiquity of virtualization technologies in cloud data centers for resource provisions and operating cost reduction, cloud user applications are now implemented as VMs that can be easily created and deployed inside PDDCs. Considering that energy consumption is a big concern in any cloud data centers [26], we study how to place the VMs inside PDDC to minimize their communication energy cost while satisfying data center policies given that hardware-based MBs are already installed inside the PDDC.

VM placement research has attracted lots of attention in recent years as it is a very effective technique to reduce network

traffic, bandwidth consumption, user application delay, as well as energy consumption in cloud data centers [21], [2], [17], [6], [11], [19]. For example, Meng et al. [21] designed one of the first *traffic-aware* VM placement algorithm, wherein VMs with large communications are assigned to the same PMs or PMs in close proximity. Alicherry and Lakshman [2] designed optimal and approximation algorithms that place VMs to minimize data access latencies while satisfying system constraints. However, none of above VM placement research considered data center policies (please refer to [19] for a complete survey of VM placement in data centers), thus falling short of achieving performance and security guarantees brought about by various of MBs deployed inside PDDCs.

While most research on hardware-based MBs focused on their Software-Defined Networking (SDN) support [23], [8], [27], [9], it has not been studied how VM placement coordinates with MBs to achieve a secure and performance-optimal PDDC. We are aware of the actively studied Network Function Virtualization (NFV) [10], [7], [22] and service function chaining [27], [18], [20], where the MBs are implemented as software or VMs running on commodity hardwares. However, given the ubiquitous existence of hardware-based MBs such as firewalls and load balancers in the market, we anticipate that they will persist for a relatively long time therefore our research is meaningful and timely.

Our Contributions. In particular, we study that given a set of MBs that have already been deployed in the PDDC and the data center policy that each communicating VM pair needs to satisfy, how to place the VMs on the PMs in order to minimize the total energy cost of the VM pairs while satisfying the resource constraint of the PMs. We refer to the problem as vrtual-machine placement in policy-driven data centers (VMP^2). Under *ordered policy*, wherein all the VM pairs must traverse the MBs in a specific order, we design a polynomial and optimal algorithm to solve VMP^2. Under *unordered policy*, wherein different VM pairs can traverse the MBs in different orders, we show that VMP^2 is NP-hard thus there does not exist efficient and polynomial VM placement algorithms. We put forward a 2-approximation algorithm that achieves total energy cost for all the VM pairs at most twice of the optimal energy cost. In both cases, we compare the algorithms with the state-of-the-art traffic-aware VM placement algorithm [21], as its problem setup is most comparable to ours. We show via extensive simulations that our algorithms constantly outperform it under different PDDC parameters in both ordered and unordered data center policies. To the extent of our knowledge, this work is the first one that tackles the VM placement to minimize energy consumption for VM communications in policy-driven data centers.

II. Problem Formulation of VMP^2

System Model. We model a PDDC as an undirected general graph $G(V, E)$. $V = V_p \cup V_s$ is the set of PMs V_p and the set of switches V_s. E is the set of edges; each edge connects either one switch to another switch or a switch to a PM. We adopt fat tree topology [1], a popular data center topology for PDDCs, but our designed algorithms are applicable to any data center topologies. Fig. 2 shows a PDDC of 16 PMs with $k = 4$ where k is the number of ports each switch has.

A set of m hardware-based MBs, denoted as $M = \{mb_1, mb_2, ..., mb_m\}$, are already deployed inside the PDDC, with mb_j being installed at switch $sw(j) \in V_s$. We adopt the *bump-off-the-wire* design [12], which takes the dedicated MB hardware out from the physical data path. It uses a policy-aware switching layer that leverages the data center network's conduciveness for indirection and explicitly redirects traffic to off-path MBs. Fig. 2 shows that three MBs MB_1, MB_2 and MB_3 are attached to the switches using this design.

There are l VM pairs $P = \{(v_1, v_1^{'}), (v_2, v_2^{'}), ..., (v_l, v_l^{'})\}$, wherein v_i communicates with $v_i^{'}$ constantly, $1 \leq i \leq l$. v_i and $v_i^{'}$ are referred to as the *source VM* and the *destination VM* of this VM pair respectively. The *communication frequency* of $(v_i, v_i^{'})$ is denoted as λ_i, indicating number of communication taking place between v_i and $v_i^{'}$ in unit time. WLOG, we assume $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_l$ (otherwise it can be sorted to be so). All the VMs need to be created and placed into the PMs for execution. We refer to the PMs where v_i and $v_i^{'}$ will be placed as the *source PM* and *destination PM* of $(v_i, v_i^{'})$.

Let $V_m = \{v_1, v_1^{'}, v_2, v_2^{'}, ..., v_l, v_l^{'}\}$. Each VM $v \in V_m$ needs one unit amount of resource to execute. Here the resource refers to an aggregated characterization of all the hardware resources needed to create and execute VMs (i.e., CPU, memory, storage, and bandwidth). The *resource capacity* of PM i is denoted as $m(i)$, which is also an aggregated characterization of its hardware resources. That is, PM i has $m(i)$ *resource slots*, each can be used to create and execute one VM. Table I shows all the notations in this paper.

TABLE I
NOTATION SUMMARY

Notation	Description
V_p	The set of physical machines (PMs) in PDDC
V_s	The set of switches in PDDC
P	The set of l VM communication pairs, $(v_i, v_i^{'})$
$p(v)$	The PM where the VM v is placed with VM placement p
M	The set of m MBs, mb_j
$sw(j)$	The switch where mb_j is installed
$c(i, j)$	The energy cost between and PM or switch i and j
c_i^p	The energy cost for $(v_i, v_i^{'})$ in ordered policy
π^i	The order at which $(v_i, v_i^{'})$ visits MBs in unordered policy
c_i^{p,π^i}	The energy cost for $(v_i, v_i^{'})$ in unordered policy
C_p	The total energy cost for all with VM placement p

PDDC Policies. For security and performance reasons, each VM pair must traverse the m MBs mb_1, mb_2, ..., mb_m in the same or different orders. In some policies, as the function of one MB can only be performed after another, it requires the cloud application VM traffic to go through the sequence of the MBs strictly in a specific order. In our previous example (Fig. 1), the VM traffic must go through the firewall first, and then the load balancer and finally the cache proxy for security and performance reasons. We refer to such policy as *ordered*

policy and denote it as $(mb_1, mb_2, ..., mb_m)$, signifying that the VM traffic must traverse mb_1, mb_2, ..., mb_m in that order.

However, as the functions of the consisted MBs are independent from each other, some policies do not require the VM traffic to follow a specific order of the MBs as long as each MB in the policy is visited thus each MB's function is performed on the traffic. We refer to such policy as *unordered policy* and denote it as $\{mb_1, mb_2, ..., mb_m\}$. For example, there is little difference to arrange a passive monitor before or after a deep packet inspector (DPI) from security point of view [16].

In either policies, we refer to the switch where the first visited MB is located as the *ingress switch* and the switch where the last visited MB is located as the *egress switch*.

Energy Model. Following [21], we measure the power consumption of any VM pair communication inside PDDC by counting the number of switches it goes through. That is, when VM traffic traverses inside the PDDC, it consumes the same amount of energy on different switches including edge, aggregate, and core switches. We are aware of some other energy models wherein the core switches handle more traffic therefore consume more energy power than aggregate switches, which consume more energy power than edge switches [4]. However, as such observations do not affect the hardness of the problem and the design and analysis of our algorithms, we do not consider them in this paper. Let $c(i, j)$ denote the minimum energy consumption between any PM (or switch) i and j.

Problem Formulation of VMP^2. We define a *VM placement function* as $p : V_m \to V_p$, signifying that VM $v \in V_m$ will be placed on PM $p(v) \in V_p$. Denote the total energy consumption of all the l VM pairs with VM placement p as C^p.

Ordered policy. We first consider ordered policy, wherein each VM pair must traverse mb_1, mb_2, ..., mb_m in that order. In this case, the ingress switch is $sw(1)$ and the egress switch is $sw(m)$. Given any VM placement function p, the energy consumption for VM pair $(v_i, v_i^{'})$ under ordered policy is then

$$c_i^p = \lambda_i \cdot c\big(p(v_i), sw(1)\big) + \lambda_i \cdot \sum_{j=1}^{m-1} c\big(sw(j), sw(j+1)\big) + \lambda_i \cdot c\big(sw(m), p(v_i^{'})\big). \quad (1)$$

In Equation 1, the first term on the right-hand side is the energy consumption of the VM pair $(v_i, v_i^{'})$ when it traverses from its source PM to the ingress switch, the second term is its energy consumption traversing the next $m-1$ MBs, and the third term is its energy consumption from the egress switch to the destination PM. We have

$$\begin{aligned} C^p &= \sum_{i=1}^{l} c_i^p \\ &= \sum_{i=1}^{l} \lambda_i \cdot \Big(c\big(p(v_i), sw(1)\big) + c\big(sw(m), p(v_i^{'})\big) \Big) \\ &+ \sum_{i=1}^{l} \lambda_i \cdot \sum_{j=1}^{m-1} c\big(sw(j), sw(j+1)\big). \end{aligned} \quad (2)$$

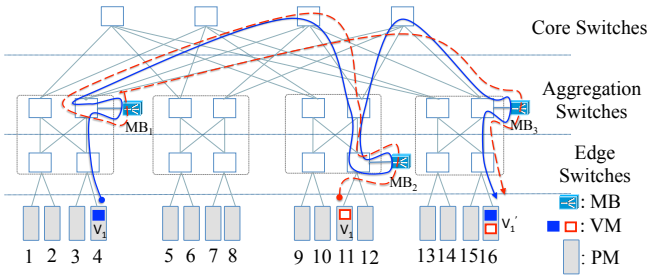


Fig. 2. A PDDC with 16 PMs, 3 MBs (MB_1, MB_2, and MB_3), and a VM pair $(v_1, v_1^{'})$. For ordered-policy (MB_1, MB_2, MB_3), the minimum energy VM placement for $(v_1, v_1^{'})$ is shown in solid blue; for unordered-policy $\{MB_1, MB_2, MB_3\}$, its minimum energy VM placement is shown in dashed red. • and ▶ indicate source VM and destination VM in each case.

Given that all the switches that MBs are installed are fixed, the second term on the right-hand side of Equation 2 is a constant. Therefore we only need to find VM placement to minimize the first term. The objective of VMP^2 is to find a VM placement p to minimize C^p while satisfying the resource capacity of each PM, $|\{v \in V_m | p(v) = i\}| \leq m_i, \forall i \in V_p$.

EXAMPLE 1: Fig. 2 shows that for a VM pair $(v_1, v_1^{'})$, under ordered policy (MB_1, MB_2, MB_3), one of its optimal VM placement is in solid blue: v_1 is placed at PM 4 and $v_1^{'}$ placed at PM 16; the VM traffic visits MB_1, MB_2, and MB_3 in that order by traversing 9 switches (therefore cost of 9). □

Unordered policy. In unordered policy, each VM pair can traverse mb_1, mb_2, ..., mb_m in different orders. Therefore to solve VMP^2, it finds not only a VM placement function but also for *each* VM pair, a sequence of MBs to visit. We define a *permutation function* $\pi^i : [1, 2, ..., m] \to [1, 2, ..., m]$, signifying that for $(v_i, v_i^{'})$, the j^{th} MB to visit is $mb_{\pi^i(j)}$.

Denote the power consumption of $(v_i, v_i^{'})$ given p and π^i as c_i^{p,π^i}. Then

$$\begin{aligned} c_i^{p,\pi^i} &= \lambda_i \cdot c\Big(p(v_i), sw\big(\pi^i(1)\big)\Big) + \\ &\lambda_i \cdot \sum_{j=1}^{m-1} c\Big(sw\big(\pi^i(j)\big), sw\big(\pi^i(j+1)\big)\Big) + \\ &\lambda_i \cdot c\Big(sw\big(\pi^i(m)\big), p(v_i^{'})\Big). \end{aligned} \quad (3)$$

The objective of VMP^2 in unordered policy case is to minimize $C^p = \sum_{i=1}^{l} c_i^{p,\pi^i}$ while satisfying the resource capacity of each PM, that is, $|\{v \in V_m | p(v) = i\}| \leq m_i, \forall i \in V_p$.

EXAMPLE 2: Fig. 2 shows for $(v_1, v_1^{'})$ under unordered policy $\{MB_1, MB_2, MB_3\}$, one of the optimal VM placement in dashed red: v_1 is placed at PM 11 and $v_1^{'}$ placed at PM 16; the VM traffic visits MB_2, MB_1, and MB_3 in that order by traversing 7 switches (therefore cost of 7). □

III. Algorithms for VMP^2

A. Ordered Policy.

For ordered policy, we design an efficient polynomial algorithm (Algorithm 1) and prove its optimality. Recall that PM i has $m(i)$ resource slots, each can be used to create and execute one VM. To place the l VM pairs $\{(v_1, v_1^{'}), (v_2, v_2^{'}), ..., (v_l, v_l^{'})\}$, Algorithm 1 identifies two disjoint sets of l resource slots each that are closest to the ingress and egress switches, respectively (line 1-24), and create and execute the source and destination VMs on these two sets of resource slots (line 25-33).

We refer to these two sets of resource slots as *ingress resource set* (IRS) and *egress resource set* (ERS). We assign each of the resource slots in the PDDC a unique ID. Each element in these two sets include the ID of the resource slot as well as the cost of its belonged PM to the corresponding ingress or egress switch. We use IRS$[i].id$, IRS$[i].dist$, ERS$[i].id$, ERS$[i].dist$ to denote ID and cost of the i^{th} resource slot in IRS and ERS respectively. We can then represent the placement of the l VM pairs $\{(v_1, v_1^{'}), (v_2, v_2^{'}), ..., (v_l, v_l^{'})\}$ as $p = \{(\text{IRS}[i].id, \text{ERS}[i].id)\}$, $1 \le i \le l$, where IRS$[i].id$ stores v_i and ERS$[i].id$ stores $v_i^{'}$.

Algorithm 1: Optimal Algorithm for Ordered Policy.
Input: A PDDC with ordered policy $(mb_1, mb_2, ..., mb_m)$, VM pairs $P = \{(v_1, v_1^{'}), (v_2, v_2^{'}), ..., (v_l, v_l^{'})\}$.
Output: A placement p and the total energy cost C^p for P.
Notations:
IRS$[i].id$, IRS$[i].dist$, ERS$[i].id$, ERS$[i].dist$: ID and cost of the i^{th} resource slot in IRS and ERS.

0. $i = 1$, $j = 1$, $k = 1$, $C^p = 0$, IRS = ϕ (empty set), ERS = ϕ, $p = \phi$;
1. Assign all resource slots in the PDDC unique IDs;
2. Sort them in ascending order of their costs to ingress switch $sw(1)$ (and egress switch $sw(m)$), store first $2l$ resource slots and costs in array A (and B);
3. **while** $(k \le l)$
4. **while** $(A[i].id \in$ IRS$)$ $i++$;
5. **while** $(B[j].id \in$ ERS$)$ $j++$;
6. **if** $(A[i].id == B[j].id$)
7. **while** $(A[i+1].id \in$ IRS$)$ $i++$;
8. **while** $(B[j+1].id \in$ ERS$)$ $j++$;
9. **if** $(A[i+1].dist \le B[j+1].dist)$
10. IRS$[k].id = A[i+1].id$, IRS$[k].dist = A[i+1].dist$;
11. ERS$[k].id = B[j].id$, ERS$[k].dist = B[j].dist$;
12. $i = i + 2$; $j++$;
13. **else**
14. IRS$[k].id = A[i].id$, IRS$[k].dist = A[i].dist$;
15. ERS$[k].id = B[j+1].id$, ERS$[k].dist = B[j+1].dist$;
16. $i = i++$; $j = j + 2$;
17. **end if;**
18. **else**
19. IRS$[k].id = A[i].id$, IRS$[k].dist = A[i].dist$;
20. ERS$[k].id = B[j].id$, ERS$[k].dist = B[j].dist$;
21. $i++$; $j++$;
22. **end if;**
23. $k++$;
24. **end while;**
25. Sort all VM pairs $\{(v_i, v_i^{'})\}$, $1 \le i \le l$, in descending order of their communication frequencies λ_i. WLOG, assume $\lambda_1 \ge \lambda_2 \ge ... \ge \lambda_l$;
26. $a = \sum_{j=1}^{m-1} c\big(sw(j), sw(j+1)\big)$;
27. **for** $(1 \le i \le l)$
28. Place v_i at resource slot IRS$[i].id$;
29. Place $v_i^{'}$ at resource slot ERS$[i].id$;
30. $p = p \cup \{(\text{IRS}[i].id, \text{ERS}[i].id)\}$;
31. $c_i^p = \lambda_i * (\text{IRS}[i].dist + a + \text{ERS}[i].dist)$; // Eqn. 1
32. $C^p = C^p + c_i^p$;
33. **end for;**
34. **RETURN** p and C^p.

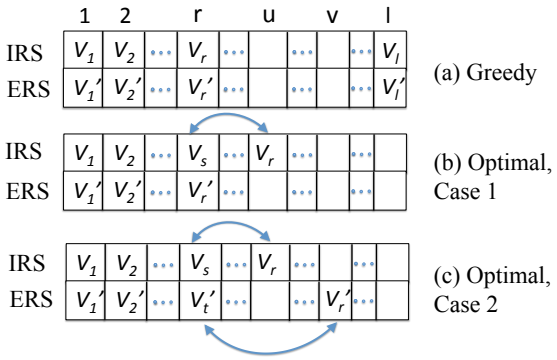


Fig. 3. Optimal Proof for Algorithm 1.

Time Complexity of Algorithm 1. Sorting all the resource slots takes $O(|V_p| \cdot \bar{m} \cdot \lg(|V_p| \cdot \bar{m}))$, where $|V_p|$ is the number of PMs and $\bar{m}$ is the average resource capacity of PMs. Finding the sets of IRS and ERS takes l rounds, each could take $O(l)$. Sorting all the VM pairs takes $O(l \cdot \lg l)$ and calculating C^p takes $O(l)$. Therefore, the total complexity of Algorithm 1 is $O(|V_p| \cdot \bar{m} \cdot \lg(|V_p| \cdot \bar{m}) + l^2)$.

Theorem 1: Algorithm 1 is optimal and finds the VM placement minimizing total energy cost for all the VM pairs.
Proof: In Algorithm 1, the pair of resource slots allocated to $(v_i, v_i^{'})$ is IRS$[i].id$ and ERS$[i].id$, wherein IRS$[i].id$ stores v_i and ERS$[i].id$ stores $v_i{'}$, as shown in Fig. 3 (a). Now, by way of contradiction, assume that Algorithm 1 is not optimal and there exists an optimal algorithm called Optimal. There must exist an instance of the VMP^2 problem such that the VM placements resulted from Greedy and Optimal are different. Let's assume that r, $1 \le r \le l$, is the smallest index at which the pair of resource slots store different pair of VMs in Algorithm 1 and Optimal. There are two cases.

Case 1: only one of the resource slots, IRS$[r].id$ or ERS$[r].id$, stores different VMs. For example, both algorithms stores $v_r^{'}$ at ERS$[i].id$ while Algorithm 1 stores v_r and Optimal stores v_s at IRS$[i].id$, as shown in Fig. 3 (b). Since r is the smallest index wherein algorithms differ, it must be $s > r$ and

$\lambda_r \geq \lambda_s$. In Optimal, v_r must be stored in a later resource slot, say IRS$[u].id$, with $u > r$ and IRS$[u].dist \geq$ IRS$[r].dist$.

Now by swapping v_s and v_r in Optimal, that is, by moving v_r from IRS$[u].id$ to IRS$[r].id$ and v_s from IRS$[r].id$ to IRS$[u].id$, the amount of energy cost reduced for $(v_r, v_r^{'})$ is $\lambda_r \cdot$ (IRS$[u].dist$-IRS$[r].dist$), and the amount of energy cost increased for $(v_s, v_s^{'})$ is $\lambda_s \cdot$ (IRS$[u].dist$-IRS$[r].dist$). Therefore, it reduces $(\lambda_r - \lambda_s) \cdot$ (IRS$[u].dist$-IRS$[r].dist$) ≥ 0 amount of energy cost for these two VM pairs while other VM pairs are not affected. This contradicts that Optimal achieves the minimum energy cost for all the VM pairs. The other case wherein Algorithm 1 and Optimal store v_r at IRS$[r].id$ while store different VMs at ERS$[r].id$ can be dealt with similarly.

Case 2: both resource slots IRS$[r].id$ and ERS$[r].id$ store different VMs for both algorithms. As shown in Fig. 3 (c), in Optimal, it stores v_s at IRS$[r].id$ and $v_t^{'}$ at ERS$[r].id$, with $s, t > r$, and v_r is stored at IRS$[u].id$ and $v_r^{'}$ at ERS$[v].id$, with $u, v > r$. Since $s, t > r$, $\lambda_r > \lambda_s, \lambda_t$. Since $u, v > r$, IRS$[u].dist \geq$ IRS$[r].dist$ and ERS$[v].dist >$ ERS$[r].dist$.

Now by swapping v_s with $v_r^{'}$ and v_t with $v_r^{'}$, the amount of energy cost reduced for $(v_r, v_r^{'})$ minus the amount of energy cost increased for $(v_s, v_s^{'})$ and $(v_t, v_t^{'})$ is

$$\begin{aligned} & \lambda_r \cdot (\text{IRS}[u].dist - \text{IRS}[r].dist) \\ & + \lambda_r \cdot (\text{ERS}[v].dist - \text{ERS}[r].dist) \\ & - \lambda_s \cdot (\text{IRS}[u].dist - \text{IRS}[r].dist) \\ & - \lambda_t \cdot (\text{ERS}[v].dist - \text{ERS}[r].dist) \qquad (4) \\ = & (\lambda_r - \lambda_s) \cdot (\text{IRS}[u].dist - \text{IRS}[r].dist) \\ & + (\lambda_r - \lambda_t) \cdot (\text{ERS}[v].dist - \text{ERS}[r].dist) \\ \geq & 0. \end{aligned}$$

This contradicts again that Optimal is optimal, which gives that Algorithm 1 is optimal. ∎

B. Unordered Policy.

For unordered-policy, we show that even for one pair of VMs, VMP^2 is NP-hard. We then propose an algorithm that achieves energy cost at most twice of the optimal.

Theorem 2: Even when there is only one pair of VMs to be placed in the PDDC (i.e., $l = 1$), VMP^2 is NP-hard.

Proof: We show that VMP^2 with $l = 1$ is equivalent to *traveling salesman path problem* (TSPP) [13], which is NP-hard. Given a complete undirected graph $K = (V_K, E_K)$, with edge costs $c : E_K \rightarrow \mathbb{R}^+$ and that these edge costs satisfy *triangle inequality* $c(u, v) \leq c(u, v) + c(v, w)$ for all $u, v, w \in V_K$, and a pair of vertices $s, t \in V_K$, TSPP is to find a cheapest path that starts at s, visits each vertex exactly once, and ends at t. When $s = t$, TSPP becomes well-known *traveling salesman problem* (TSP), which is to find a cheapest Hamiltonian cycle that starts at s, visits each vertex exactly once, and returns to s.

To show that VMP^2 with $l = 1$ is equivalent to TSPP, we transform any instance of the PDDC graph $G(V = V_p \cup V_s, E)$ to a complete graph $K = (V_K, E_K)$. Here, $V_K = \{pm_1, pm_2, sw(1), sw(2), ...sw(m)\}$ includes an arbitrary pair of PMs pm_1 and pm_2 and m switches with MBs. The weight of edge $(u, v) \in E_K$ is $c(u, v)$, the minimum energy consumption between u and v in original PDDC graph $G(V, E)$. VMP^2 with $l = 1$ in G is then TSPP in K with $s = pm_1$ and $t = pm_2$, by running on all $|V_p| \cdot (|V_p| + 1)/2$ PM pairs (pm_1, pm_2). Among these pairs, $|V_p|$ pairs have $pm_1 = pm_2$, at which it becomes TSP. ∎

Algorithm for Unordered Policy. With above preparation, we propose Algorithm 2 below.

Algorithm 2: Approximation Algorithm for Unordered Policy.

Step 0. For arbitrary pair of PMs $pm_1, pm_2 \in V_p$, construct above defined complete graph K.

Step 1. Find a minimum spanning tree MST of K.

Step 2. Find a walk W from pm_1 to pm_2 on MST by visiting all vertices using each edge *at most twice*, and calculate the cost of this walk. If $pm_1 = pm_2$, it is indeed to find a Hamiltonian cycle from pm_1 to pm_1 on MST by visiting all vertices using each edge *exactly twice*.

Step 3. Repeat Step 0 to 2 for all $|V_p| \cdot (|V_p| + 1)/2$ pairs of PMs (pm_1, pm_2), find all such walks/cycles and their costs.

Step 4. Sort all the walks and cycles obtained in Step 3 in the ascending order of their costs, and order the corresponding PM pairs accordingly. Note each PM pair could have two different PMs or one PM paired with itself.

Step 5. Sort all VM pairs $\{(v_i, v_i^{'})\}$, $1 \leq i \leq l$ in descending order of their communication frequencies λ_i. WLOG, assume $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_l$;

Step 6. Place each VM pair in the sorted PM pairs while satisfying their resource capacities.

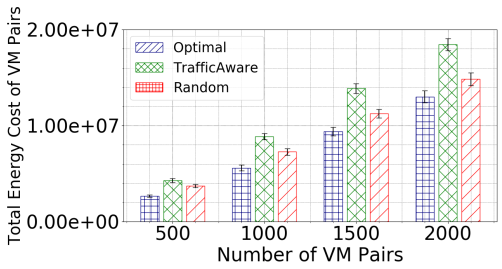
Time Complexity of Algorithm 2. Step 0 takes $O(m^3)$ as finding the all-pair shortest-paths between all the m switches with MBs takes $O(m^3)$. Step 1 takes $O(m^2 \text{lg} m)$ for a complete graph of $m + 2$ or $m + 1$ nodes. Step 2 takes $O(m^2)$. Therefore the running time of Algorithm 2 is $(|V_p| \cdot (|V_p| + 1)/2 \cdot m^3) = O(|V_p|^2 \cdot m^3)$.

Theorem 3: Algorithm 2 is a 2-approximation algorithm. That is, Algorithm 2 achieves total energy cost of all the VM pairs that is at most two times of the optimal cost.

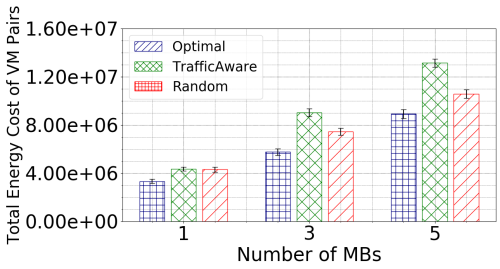
Proof: Let W^* denote the optimal walk from pm_1 to pm_2 on a given K. The cost of the MST computed in Step 1 is a lower bound on the cost of the optimal walk, $c(\text{MST}) \leq c(W^*)$. Since the walk W found in Step 2 visits all vertices using each edge of the MST at most twice, $c(W) \leq 2 \cdot c(\text{MST})$. Therefore we have $c(W) \leq 2 \cdot c(W^*)$. ∎

Traffic-Aware VM Placement [21]. Meng et al. [21] proposed a traffic-aware VM placement algorithm by optimizing the placement of VMs on host machines. They observed that traffic patterns among VMs can be coupled with the communication distance between them to minimize the energy consumption of the VM communications. That is, VMs with large amount of traffic should be assigned to the same PMs or PMs in close proximity. We refer to this algorithm as **TrafficAware**. In ordered-policy, TrafficAware considers all the VM pairs in their descending order of communication frequencies, and places each VM pair to the PM that is closest to the ingress switch until all the VM pairs are placed.

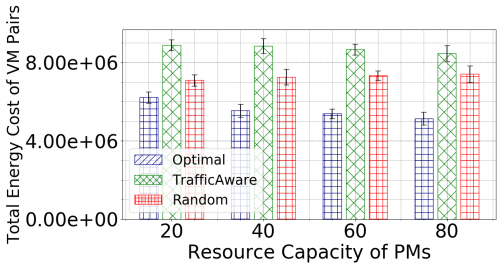
In unordered-policy, it works as Algorithm 2 while only considering the case of $pm_1 = pm_2$, as TrafficAware always places each VM pair in the same PM if possible. In Section IV, we compare our algorithms with TrafficAware in both ordered- and unordered-policy, and show that our algorithms constantly outperform TrafficAware.



(a) Varying l, number of VM pairs. $m = 3$, $rc = 40$.



(b) Varying m, number of MBs. $l = 1000$, $rc = 40$.



(c) Varying rc, resource capacity of PMs. $l = 1000$, $m = 3$.

Fig. 4. Performance comparison in ordered-policy.

IV. Performance Evaluation

Simulation Setup. We investigate the performances of our policy-aware VM placement algorithms. We refer to Algorithm 1 in ordered-policy as **Optimal** and the Algorithm 2 in unordered-policy as **Approximation**. We consider a PDDC of $k = 8$ fat-tree topology with 128 PMs. There are $m = 1, 3, 5$ MBs that are randomly placed on different switches in the PDDC. We vary the number of VM pairs l from 500, 1000, 1500, to 2000. The communication frequency of each VM pair is a random number in [1, 1000]. In all the simulation plots, each data point is an average of 20 runs, with error bars indicating 95% confidence interval.

We design two more algorithms for further comparison. For ordered-policy, we design an algorithm that randomly places all the VMs in the PMs and refer to it as **Random**; for unordered-policy, we design a greedy algorithm called **Greedy** that works as follows. For each VM pair $(v_i, v_i^{'})$, the Greedy places v_i at a PM that is closest (in terms of energy cost) to one of the MBs among all the PMs with available resources, then from this MB it visits an unvisited MB that is closest, so on and so forth until all the MBs are visited; and finally places $v_i^{'}$ at a PM that is closest to the last visited MB.

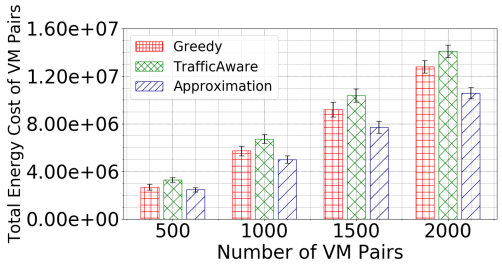
Ordered Policy. First we vary the number of VM pairs l from 500, 1000, 1500, to 2000 while setting the number of MBs m as 3 and resource capacity of each PM rc as 40. Fig. 4 (a) shows that for each algorithm, the total energy cost of all the VM pairs increases with the increase of l. This is obvious as more VMs communication consumes more energy. Besides, in all the cases, Optimal performs better than Random, which performs better than TrafficAware. Optimal is the only one among the three that is policy-aware, therefore performs the best. The reason that even Random performs better than TrafficAware is as follows. As TrafficAware assigns VM pairs into the same PM or PMs in proximity, the VM traffic visits all the MBs and then must return to where it starts. In contrast, as the MBs are randomly placed as well as the VMs in Random algorithm, VM traffic does not always return therefore saving energies.

Fig. 4 (b) varies m from 1 to 3 to 5, while keeping l as 1000 and rc as 40. We observe that with increase of number of MBs, the energy cost of each algorithm increases as each VM pair traverses more MBs. Again we observe that Optimal performs better than Random, which performs better than TrafficAware. Finally, Fig. 4 (c) varies rc from 20, 40, 60, to 80 while keeping l as 1000 and m as 3. We observe that Optimal performs the best. We also observe that increasing rc reduces the total energy costs for Optimal and TrafficAware. As both algorithms attempt to place VMs to the PMs close to the MBs, increasing resource capacities of PMs can thus place more VMs closer to the MBs, therefore saving energies for these two algorithms. Note this is not the case for the Random.

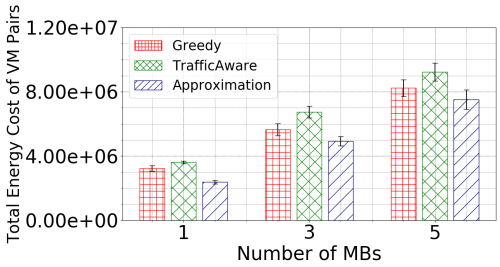
Unordered Policy. Fig. 5 shows the performance comparison under unordered-policy. It shows that both Approximation and Greedy outperform TrafficAware in all different scenarios, as both are policy-aware for VM placement while TrafficAware is not. Approximation performs better than Greedy, which indirectly demonstrates the energy-efficiency of its 2-approximation. Finally, we observe that both Approximation and TrafficAware yield less energy cost for VM pairs compared to ordered-policy (this is more evident for large l and large m). This is because unlike in ordered policy wherein each VM must traverse the MBs in a specific order, in unordered policy, VM pairs can choose the order of the MBs to traverse in order to reduce their energy cost.

V. Conclusions and Future Work.

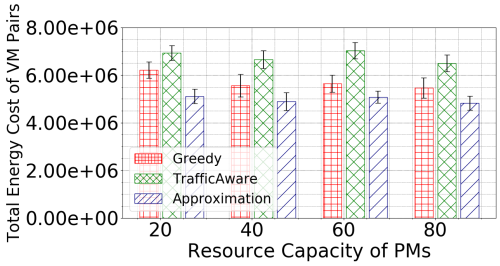
In this paper we have studied VMP^2, a new virtual machine placement problem in PDDCs. PDDCs have become important infrastructures for cloud computing as the MB-based policies provide the cloud user applications with security and performance guarantees. The key observation underlying VMP^2



(a) Varying l, number of VM pairs. $m = 3$, $rc = 40$.



(b) Varying m, number of MBs. $l = 1000$, $rc = 40$.



(c) Varying rc, resource capacity of PMs. $l = 1000$, $m = 3$.

Fig. 5. Performance comparison in unordered-policy.

is that current MBs are most hardware-based, therefore can not be moved around easily once deployed; in contrast, VMs are much easier and convenient for placement. We therefore uncover a new algorithmic problem that places VMs inside PDDC while respecting the existing MBs and policies, with the objective to minimize the total energy consumption of the VM applications. We solved VMP^2 by designing optimal and approximation algorithms under ordered- and unordered-policy, respectively. As the future work, we would like to study if the optimality and approximability of our algorithms still hold when different VMs require different amount of resources. We will also study in network function virtualization, when both MBs and VMs can be placed easily inside PDDCs, how to design a holistic and synergistic MB and VM placement to achieve ultimate energy-efficiency in PDDCs.

REFERENCES

[1] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. *SIGCOMM Comput. Commun. Rev.*, 38(4):63–74, August 2008.

[2] M. Alicherry and T.V. Lakshman. Optimizing data access latencies in cloud systems by intelligent virtual machine placement. In *Pro. of IEEE INFOCOM 2013*.

[3] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, 2010.

[4] D. Boru, D. Kliazovich, F. Granelli, P. Bouvry, and A. Y. Zomaya. Energy-efficient data replication in cloud computing datacenters. *Cluster Computing*, 18(1):385–402, 2015.

[5] B. Carpenter and S. Brim. Middleboxes: Taxonomy and issues, 2002. https://tools.ietf.org/html/rfc3234.

[6] R. Cohen, L. Lewin-Eytan, J. Seffi Naor, and D. Raz. Almost optimal virtual machine placement for traffic intense data centers. In *Proc. of IEEE INFOCOM 2013*.

[7] R. Guerzoni et al. Network functions virtualisation: an introduction, benefits, enablers, challenges and call for action, introductory white paper. In *SDN and OpenFlow World Congress*, 2012.

[8] S. K. Fayazbakhsh, L. Chiang, V. Sekar, M. Yu, and J. C. Mogul. Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags. In *Proc. of USENIX NSDI 2014*.

[9] A. Gember, P. Prabhu, Z. Ghadiyali, and A. Akella. Toward software-defined middlebox networking. In *Proc. of HotNets-XI 2012*.

[10] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella. Opennf: Enabling innovation in network function control. In *Proc. of the ACM SIGCOMM 2014*.

[11] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang. Joint vm placement and routing for data center traffic engineering. In *Proc. of IEEE INFOCOM 2012*.

[12] D. A. Joseph, A. Tavakoli, and I. Stoica. A policy-aware switching layer for data centers. In *Proc. of the ACM SIGCOMM 2008*.

[13] F. Lam and A. Newman. Traveling salesman path problems. *Mathematical Programming*, 113:39 – 59, 2008.

[14] C. Lan, J. Sherry, R. Popa, S. Ratnasamy, and Z. Liu. Embark: Securely outsourcing middleboxes to the cloud. In *Proc. of NSDI*, 2016.

[15] L. E. Li, V. Liaghat, H. Zhao, M. Hajiaghayi, D. Li, G. Wilfong, Y. R. Yang, and C. Guo. Pace: Policy-aware application cloud embedding. In *Proc. of IEEE INFOCOM 2013*.

[16] X. Li and C. Qian. A survey of network function placement. In *IEEE CCNC 2016*.

[17] X. Li, J. Wu, S. Tang, and S. Lu. Let's stay together: Towards traffic aware virtual machine placement in data centers. In *Proc. of IEEE INFOCOM 2014*.

[18] J. Liu, Y. Li, Y. Zhang, L. Su, and D. Jin. Improve service chaining performance with optimized middlebox placement. *IEEE Transactions on Services Computing*, 2015.

[19] Z. A. Mann. Allocation of virtual machines in cloud data centers - a survey of problem models and optimization algorithms. *ACM Comput. Surv.*, 48(1):11:1–11:34, 2015.

[20] S. Mehraghdam, M. Keller, and H. Karl. Specifying and placing chains of virtual network functions. In *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*, 2014.

[21] X. Meng, V. Pappas, and L. Zhang. Improving the scalability of data center networks with traffic-aware virtual machine placement. In *Proc. of IEEE INFOCOM 2010*.

[22] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys and Tutorials*, 18(1), 2015.

[23] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu. Simplefying middlebox policy enforcement using sdn. In *Proc. of the ACM SIGCOMM 2013*.

[24] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar. Making middleboxes someone else's problem: Network processing as a cloud service. In *Proc. of the ACM SIGCOMM 2012*.

[25] Z. Wang, Z. Qian, Q. Xu, Z. Mao, and M. Zhang. An untold story of middleboxes in cellular networks. *ACM SIGCOMM Comput. Commun. Rev.*, 41(4):374–385, 2011.

[26] Q. Zhang, L. Cheng, and R. Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18, 2010.

[27] Y. Zhang, N. Beheshti, L. Beliveau, G. Lefebvre, R. Manghirmalani, R. Mishra, R. Patney, M. Shirazipour, R. Subrahmaniam, C. Truchan, and M. Tatipamula. Steering: A software-defined networking for inline service chaining. In *Proc. of the IEEE ICNP 2013*.