Optimal Approximation Algorithm of Virtual Machine Placement for Data Latency Minimization in Cloud Systems

Jian-Jhih Kuo, Hsiu-Hsien Yang, and Ming-Jer Tsai
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan, ROC
E-mail: mjtsai@cs.nthu.edu.tw

***Abstract*—The MapReduce/Hadoop architecture has become very important and effective in cloud systems because many data-intensive applications are usually required to process big data. In such environments, big data is partitioned and stored over several data nodes; thus, the total completion time of a task would be delayed if the maximum access latency among all pairs of a data node and its assigned computation node is not bounded. Moreover, the computation nodes usually need to communicate with each other for aggregating the computation results; therefore, the maximum access latency among all pairs of assigned computation nodes also needs to be bounded. In the literature, it has been proved that the placement problem of computation nodes (virtual machines) to minimize the maximum access latency among all pairs of a data node and its assigned computation node and among all pairs of assigned computation nodes does not admit any approximation algorithm with a factor smaller than two, whereas no approximation algorithms have been proposed so far. In this paper, we first propose a 3-approximation algorithm for resolving the problem. Subsequently, we close the gap by proposing a 2-approximation algorithm, that is, an optimal approximation algorithm, for resolving the problem in the price of higher time complexity. Finally, we conduct simulations for evaluating the performance of our algorithms.**

I. Introduction

As data-intensive applications are increasingly moving to cloud systems, big data processing is becoming an important concern [1], [2]. In cloud systems, the processing framework of MapReduce/Hadoop [3], [4] is a very effective and appropriate method for executing these applications. In the MapReduce/Hadoop framework, big data is partitioned and stored over several data nodes in a cloud system. When a computation is requested, it is partitioned over several computation nodes to process the big data stored in several data nodes. Running parallel computations can significantly reduce the total completion time of a task. In an ideal environment, all the load-balanced tasks access and process the data locally (e.g., in the same rack/machine) and the performance is not degraded by unbalanced load, communication overhead, and access latency. However, this is impossible in the real world due to capability and capacity limitations of the heterogeneous nodes, considerable growth in computation and storage needs, etc. Therefore, the manner in which the problems of cloud systems should be addressed is attracting considerable attention.

When tasks and data are spread over numerous nodes, the completion time of slow progressing tasks becomes a performance bottleneck for big data processing applications. The completion time of a task could be delayed by unbalanced loaded machines/tasks, slow processors, network congestion, etc.; therefore, there are several different ways of dealing with the problem. Some previous works focus on balancing the load of each task [3], whereas others refer to job scheduling (i.e., delay the job until suitable resources are released) [5] and detecting the slower tasks and executing speculative tasks (which is a copy of the tasks) simultaneously to obtain the first return result [3], [6].

The placement of computation nodes[i] to achieve a smaller access latency between the computation node and its responsible data node is addressed in [2]. In a cloud system, an efficient placement algorithm for computation nodes does not only reduce the access latency but also reduce bandwidth consumption. This is because the smaller latency between two nodes usually indicates the fewer passed switches [7]. Moreover, in many applications, the computation nodes also need to communicate with each other. For example, in MapReduce, the computation results need to be aggregated. In [2], the placement problem of computation nodes to minimize the maximum access latency among all pairs of a computation node and its responsible data node and all pairs of computation nodes is introduced; meanwhile, the problem is shown to be NP-hard and not to admit any approximation unless NP = P; even if the access latency satisfies the triangle inequality[ii], the

This work was supported by the National Science Council of ROC under the contract No. NSC102-2221-E-007-063-MY3.

978-1-4799-3360-0/14/$31.00 ©2014 IEEE

[i] In a cloud system, the computations are moved closer to the data node instead of the data being moved to the computation node because moving big data could cause serious network congestion and long computation delay [1].

[ii] Usually, the access latency between two nodes in a network is dominated by the number of switches between the two nodes [7]–[10]. In many network architectures inside a data center, such as Tree, VL2, and Fat-Tree [7], the number of switches obeys the triangle inequality. Thus, it is reasonable to assume that the access latency satisfies the triangle inequality inside a data center [2], [11].

problem is still NP-hard and does not admit any approximation with a factor smaller than two unless NP = P. Thus far, given that the access latency satisfies the triangle inequality, no approximation algorithm has been proposed for the problem in the literature, thereby providing the motivation for this paper. For the problem, assuming that the access latency satisfies the triangle inequality, we initially propose a 3-approximation algorithm, and subsequently propose a 2-approximation algorithm, that is, an optimal approximation algorithm, in the price of higher time complexity.

The remainder of this paper is organized as follows. We study a scenario concerning the placement of computation nodes (virtual machines) for data nodes in a cloud system and introduce the problem of placing virtual machines for data nodes (VMPDN) based on the scenario studied in Section II. Subsequently, we propose 3-approximation and 2-approximation algorithms for VMPDN in Sections III and IV, respectively. We consider the variation of VMPDN and investigate the approximation algorithm in Section V. Using simulations, we evaluate the performance of our algorithms in Section VI. Related works are presented in Section VII. Finally, we conclude this paper in Section VIII.

II. Virtual Machine Placement for Data Node

Initially, we demonstrate the scenario, which is studied in [2], in Section II-A. Subsequently, based on the scenario described, we formally define the problem in Section II-B.

A. Scenario

Consider a network inside a data center comprising numerous data nodes (DNs) and computation nodes. Each computation node has several available virtual machines (VMs). The user stores his/her data in some DNs and could request a certain number of VMs to process his/her data. For simplicity, we assume that each DN only requires a VM to process the stored data[iii]. The location of each DN is allocated in advance and is fixed, that is, during the computation, the data will not migrate to another DN. The access latency between a VM and DN or between two VMs is determined by measurement and estimation during preprocessing. Our goal is to determine the placement (assignment) of VMs for DNs such that the maximum access latency among all pairs of a DN and its assigned VM and all pairs of assigned VMs (the VMs that are assigned to DNs in the assignment) is minimized.

B. The Problem and Hardness

We define VMPDN based on the studied scenario in Definition 1. The objective of VMPDN is to minimize the maximum access latency among all pairs of a DN and its assigned VM and all pairs of assigned VMs in a data center. In VMPDN, it is assumed that the access latency satisfies the triangle inequality. Two constraints are imposed on VMPDN. First, each DN must be assigned to a single VM[iii]. Second, two DNs should not be permitted to be assigned to a VM, that is, a VM must be assigned to at most one DN.

[iii] As a DN needs k ($k > 1$) VMs to process the data, our algorithms can also resolve the problem by making $k-1$ replicated copies of the DN in the instance of the problem.

Definition 1. Let D and V be a set of DNs and a set of VMs, respectively, and let d_{ab} be the access latency between two VMs $a, b \in V$ or between a DN $a \in D$ and a VM $b \in V$. Given a graph $G = (D \cup V, E)$ with weights $d_{ab} \in \mathbb{R}^+$ associated with all $e_{ab} \in E$, where $a, b \in V$ or $a \in D, b \in V$, the **V**IRTUAL **M**ACHINE **P**LACEMENT for **D**ATA **N**ODE PROBLEM (VMPDN) asks for the assignment of DNs to VMs such that:

1) each DN is assigned to a single VM,
2) each VM is assigned to at most one DN, and
3) the maximum access latency in the assignment is minimized, where the access latency satisfies the triangle inequality.

VMPDN is formulated as a mixed integer program as follows.

$$\text{minimize} \quad d \tag{1a}$$

$$\text{subject to} \quad \sum_{j \in V} x_{ij} = 1, \ \forall i \in D \tag{1b}$$

$$\sum_{i \in D} x_{ij} \leq 1, \ \forall j \in V \tag{1c}$$

$$x_{ij}(d - d_{ij}) \geq 0, \ \forall i \in D, \forall j \in V \tag{1d}$$

$$(\sum_{i \in D} x_{ij^1})(\sum_{i \in D} x_{ij^2})(d - d_{j^1 j^2}) \geq 0, \quad \forall j^1, j^2 \in V, j^1 \neq j^2 \tag{1e}$$

$$x_{ij} \in \{0, 1\}, \ \forall i \in D, \forall j \in V \tag{1f}$$

$$d \geq 0 \tag{1g}$$

Note that the program is not a linear (or integer) program because a fraction cannot (or can) be assigned to variable x_{ij} (or d) due to constraints in 1f (or 1g). It should also be noted that $x_{ij} = 1$ if and only if DN i is assigned to VM j. The object function 1a accounts for the goal to minimize the maximum access latency d in the assignment. If constraints in 1b and 1f are satisfied, any DN is assigned to a single VM. If constraints in 1c and 1f are satisfied, an arbitrary VM is assigned to at most one DN. Constraints in 1d ensure that the access latency between an arbitrary pair of a DN and its assigned VM is not greater than d. Note that $\sum_{i \in D} x_{ij} > 0$ if and only if VM j is assigned to some DN. Thus, constraints in 1e ensure that the access latency between any two assigned VMs j^1, j^2 is at most d. The following result is shown in [2].

Theorem 1. *VMPDN is NP-hard and cannot be approximated within a factor of* $2 - \epsilon$ *for any* $\epsilon > 0$.

III. 3-Approximation Algorithm

Initially, we propose an algorithm (Algorithm 1) for VMPDN in Section III-A. Subsequently, Algorithm 1 is shown to be a 3-approximation algorithm for VMPDN in Section III-B.

A. The Algorithm

The primary idea of Algorithm 1 is to employ the *threshold* technique. Algorithm 1 initially sets the threshold t to the minimum weight of edges (lines $1-2$, $4-5$), increases the threshold t in each round (lines $4-5$), and employs Procedure 1 to examine whether there is a feasible assignment of DNs to VMs under the threshold t (lines $6-8$). If Procedure 1 can make an assignment of DNs to VMs as the threshold t is not greater than the maximum weight of edges, Algorithm 1 outputs the assignment made by Procedure 1 (lines $6-8$); otherwise, Algorithm 1 outputs no feasible solution (line 10).

Algorithm 1 3-Approximation Algorithm for VMPDN

Input: A graph $G = (D \cup V, E)$ with weights d_{ab} associated with all $e_{ab} \in E$
1: Index all edges in E in a nondecreasing order of the weights;
2: $m \leftarrow 0$;
3: **while** $m < |E|$ **do**
4: $m \leftarrow m + 1$;
5: $t \leftarrow$ the weight of the edge with index m;
6: **if** Procedure 1 with input parameters G, t, and the constraints of 3tVMPDN (i.e. constraints in 3a $-$ 3e) outputs an assignment A **then**
7: Return the assignment A;
8: **end if**
9: **end while**
10: Return no feasible solution;

Subsequently, we design an algorithm to solve VMPDN with a fixed threshold t (tVMPDN). For readability, tVMPDN is formulated as the following integer program.

$$\text{subject to} \quad \sum_{j \in V} x_{ij} = 1, \ \forall i \in D \tag{2a}$$

$$\sum_{i \in D} x_{ij} \leq 1, \ \forall j \in V \tag{2b}$$

$$x_{ij}(t - d_{ij}) \geq 0, \ \forall i \in D, \forall j \in V \tag{2c}$$

$$(\sum_{i \in D} x_{ij^1})(\sum_{i \in D} x_{ij^2})(t - d_{j^1 j^2}) \geq 0, \quad \forall j^1, j^2 \in V, j^1 \neq j^2 \tag{2d}$$

$$x_{ij} \in \{0, 1\}, \ \forall i \in D, \forall j \in V \tag{2e}$$

We undertake the design of an LP rounding algorithm for tVMPDN. Note that because tVMPDN has quadratic constraints in 2d, we cannot get an LP solution for tVMPDN using an LP algorithm. Thus, we need to remove constraints in 2d while ensuring that:

(P1) any feasible solution of tVMPDN is a feasible solution of the new problem and

(P2) the maximum access latency is bounded in the new problem.

Observe that if we directly remove constraints in 2d from tVMPDN, the access latency between any two assigned VMs cannot be bounded. Thus, we need to add new constraints to ensure that properties P1 and P2 are satisfied. Note that the access latency satisfies the triangle inequality; thus, the access latency between any two assigned VMs must be bounded by $3t$ if new constraints, that is, the access latency between an arbitrary assigned VM and an arbitrary DN is bounded by $2t$, are added. On the other hand, in tVMPDN, because both the access latency between any two assigned VMs and the access latency between an arbitrary DN and its assigned VM are bounded by t, the access latency between an arbitrary assigned VM and an arbitrary DN is bounded by $2t$. In other words, any feasible solution of tVMPDN satisfies the newly added constraints. This implies that any feasible solution of tVMPDN is a feasible solution of the new problem, termed 3tVMPDN, obtained from tVMPDN by replacing constraints in 2d with constraints in 3d. The integer program of 3tVMPDN is described as follows.

$$\text{subject to} \quad \sum_{j \in V} x_{ij} = 1, \ \forall i \in D \tag{3a}$$

$$\sum_{i \in D} x_{ij} \leq 1, \ \forall j \in V \tag{3b}$$

$$x_{ij}(t - d_{ij}) \geq 0, \ \forall i \in D, \forall j \in V \tag{3c}$$

$$(\sum_{i \in D} x_{ij})(2t - d_{ij}) \geq 0, \ \forall i \in D, \forall j \in V \tag{3d}$$

$$x_{ij} \in \{0, 1\}, \ \forall i \in D, \forall j \in V \tag{3e}$$

Procedure 1 LP Rounding

Input: A graph $G = (D \cup V, E)$ with weights d_{ab} associated with all $e_{ab} \in E$, a threshold t, and constraints of a problem
11: Get the LP relaxation of the input problem by relaxing the input constraints $x_{ij} \in \{0, 1\}$ to $x_{ij} \in [0, 1]$;
12: Get a feasible LP solution x' for the LP relaxation of the input problem using an LP solver;
13: **if** no feasible LP solution exists **then**
14: Return no feasible solution;
15: **end if**
16: Construct a bipartite graph $G' = (D' \cup V', E')$ by Procedure 2 with input parameters G and x';
17: Find a matching M to saturate all nodes $i \in D'$ in G' using an algorithm for BMP;
18: **for** each edge between nodes i and j in the matching M **do**
19: Assign DN i to VM j;
20: **end for**
21: Return the assignment;

Procedure 1 is an LP rounding algorithm that is used to find a feasible solution of 3tVMPDN as the constraints of 3tVMPDN are input to Procedure 1. First, the integer program formulation of 3tVMPDN is changed to the linear program formulation by relaxing constraints in 3e to $x_{ij} \in [0, 1]$ (line 11). Subsequently, a feasible LP solution x' for the LP relaxation of 3tVMPDN is obtained by an LP solver (line 12). After x' is obtained, we find a feasible solution of 3tVMPDN based on Theorem 2. We first use x' to construct a bipartite graph $G' = (D' \cup V', E')$, where $D' = D$, $V' = \{j | \sum_{i \in D} x'_{ij} > 0\}$, and $E' = \{e_{ij} | x'_{ij} > 0\}$ by Procedure 2 (line 16). Observe that $\sum_{j \in V} x'_{ij} = 1$ for each $i \in D$ and $\sum_{i \in D} x'_{ij} \leq 1$ for each

Procedure 2 Bipartite_Graph_Construction

Input: A graph $G = (D \cup V, E)$ and an LP solution x'
22: **for** each DN $i \in D$ **do**
23: Create node i in D';
24: **end for**
25: **for** each VM $j \in V$ **do**
26: **if** $\sum_{i \in D} x'_{ij} > 0$ **then**
27: Create node j in V';
28: **end if**
29: **end for**
30: **for** each pair of nodes $i \in D'$ and $j \in V'$ **do**
31: **if** $x'_{ij} > 0$ **then**
32: Establish an edge e_{ij} in E';
33: **end if**
34: **end for**

$j \in V$. Thus, there is a fractional matching (see Definition 2) that saturates all nodes in D' (see Lemma 1). Then, we employ an algorithm for the BMP problem (see Definition 3) to find a matching M to saturate all $i \in D'$ in G' (line 17). By Theorem 2, M must exist. Finally, we obtain a feasible solution of 3tVMPDN by setting $x_{ij} = 1$ if edge e_{ij} is in M and setting $x_{ij} = 0$ if edge e_{ij} is not in M and obtain the corresponding assignment of DNs to VMs (lines 18 − 20).

Definition 2. [12] Given a graph $G = (V, E)$, a fractional matching is an assignment of fractional values to the edges in G such that, for each node, the sum of the values on the incident edges is at most 1. A node is said to be saturated in a fractional matching if the the sum of the values on the incident edges is exactly 1.

Definition 3. [13] Given a bipartite graph $G = (U \cup V, E)$, the **B**IPARTITE **M**ATCHING **P**ROBLEM (BMP) asks for the matching, a subset of E, such that each node in U is matched (saturated).

Theorem 2. [13] *Given any instance $G = (U \cup V, E)$ of BMP, if there is a fractional matching that saturates all nodes in U, then there is an (integral) matching that saturates all nodes in U.*

Example 1. Example of Algorithm 1 is illustrated in Fig. 1. In line 1, edges $e_{1\theta}$ and $e_{\theta\sigma}$ have the smallest index (1) and greatest index (25) because they have the smallest weight (10) and greatest weight (36), respectively. In line 5, t is initially set to 10. In line 6, Procedure 1 with G shown in Fig. 1(a), $t = 10$, and constraints in 3a − 3e as the input parameters is executed. Then, a feasible LP solution x' for the LP relaxation of 3tVMPDN is obtained in line 12. Note that $x'_{i\sigma} = 0$ for each $i \in D$. This must hold for $t = 10$; otherwise, constraints in 3d must be violated because $d_{1\sigma} = 28 > 2t = 20$. In line 16, a bipartite graph $G' = (D' \cup V', E')$, shown in Fig. 1(b), is constructed by Procedure 2 using x'. Note that $D' = D$ (lines 22 − 24); $\sigma \notin V'$ because $\sum_{i \in D} x'_{i\sigma} = 0$ (lines 25 − 29); and, $E' = \{e_{1\theta}, e_{1\tau}, e_{1\psi}, e_{2\pi}, e_{3\tau}, e_{3\psi}\}$ (lines 30 − 34). In line 17, a matching $M \subset E'$ is found, as illustrated in Fig. 1(b). In lines 18 − 20, DNs 1, 2, and 3 are assigned to VMs θ, π, and τ, respectively. Note that the maximum access latency in the assignment is $28 < 3t = 30$. ■

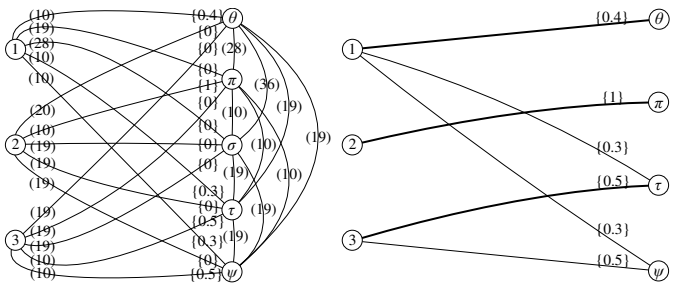

Fig. 1. Example of Algorithm 1. (a) The input graph $G = (D \cup V, E)$, where $D = \{1, 2, 3\}$, $V = \{\theta, \pi, \sigma, \tau, \psi\}$, and the weights of the edges in E are shown in parentheses. The value of variable x'_{ij} in a feasible LP solution x' is shown in a brace. (b) The constructed bipartite graph $G' = (D' \cup V', E')$. The edges in a matching in G' are shown in bold.

B. The Analysis

Lemma 1. *There is a fractional matching that saturates all nodes in D' for the bipartite graph $G' = (D' \cup V', E')$ constructed by Procedure 2 as an LP solution for the LP relaxation of 3tVMPDN x' is input to Procedure 2.*

Proof: We assign fractional values to each edge in E' to obtain the desired fractional matching in G'. Note that in lines 30 − 34, an edge e_{ij} is established between $i \in D'$ and $j \in V'$ in G' if $x'_{ij} > 0$. Let x'_{ij} be assigned to the established edge e_{ij} between $i \in D'$ and $j \in V'$ in G'. By Definition 2, it suffices to show $\sum_{j \in V'} x'_{ij} = 1$ for all $i \in D'$ and $\sum_{i \in D'} x'_{ij} \leq 1$ for all $j \in V'$. Note that $D' = D$ and $V' = \{j | \sum_{i \in D} x'_{ij} > 0\} \subseteq V$; thus, we need only to show $\sum_{j \in V} x'_{ij} = 1$ for all $i \in D$ and $\sum_{i \in D} x'_{ij} \leq 1$ for all $j \in V$. This is clearly true since x' must satisfy constraints in 3a and 3b due to the fact that x' is an LP solution for the LP relaxation of 3tVMPDN. ■

Theorem 3. *If 3tVMPDN has a feasible solution in $G = (D \cup V, E)$ under the threshold t, Procedure 1 with input parameters G, t, and the constraints of 3tVMPDN must output a feasible solution of 3tVMPDN.*

Proof: Assume that there is a feasible solution of 3tVMPDN in G under the threshold t. Then, a feasible LP solution x' for the LP relaxation of 3tVMPDN can always be obtained using an LP solver in line 12. Thus, by Lemma 1, there is a fractional matching that saturates all nodes in D' for the bipartite graph $G' = (D' \cup V', E')$ constructed by Procedure 2 using x'. This implies that there is a matching M that saturates all nodes in D' for $G' = (D' \cup V', E')$ in line 17 by Theorem 2, and an assignment of DNs to VMs can be obtained accordingly in lines 18 − 20. To complete the proof, we need to show such an assignment satisfies: (c1) each DN $i \in D$ is assigned to a single VM, (c2) each VM $j \in V$ is assigned to at most one

DN, (c3) the access latency between any DN and its assigned VM is not greater than t, and (c4) the access latency between an arbitrary assigned VM and an arbitrary DN is not greater than $2t$. First note that each DN $i \in D'$ is assigned to a single VM and each VM $j \in V'$ is assigned to at most one DN since DN $i \in D'$ is assigned to VM $j \in V'$ if and only if e_{ij} is in the matching M (lines $18-20$) and the matching M saturates all nodes in D' (line 17). This implies c1 and c2 because $D' = D$ and $V' = \{j | \sum_{i \in D} x'_{ij} > 0\} \subseteq V$ (lines $22-29$). In addition, x' is an LP solution for the LP relaxation of 3tVMPDN (line 12), x' must satisfy constraints in 3c and 3d. If DN i is assigned to VM j, e_{ij} is in $M \subseteq E'$ (lines $18-20$). This implies $x'_{ij} > 0$ (lines $30-34$). According to constraints in 3c, we have $d_{ij} \leq t$. Thus, c3 holds. Besides, consider an arbitrary DN i and an arbitrary assigned VM j. Then, $j \in V'$; otherwise, VM j cannot be assigned to any DN. This implies $\sum_{i \in D} x'_{ij} > 0$ (lines $25-29$). According to constraints in 3d, we have $d_{ij} \leq 2t$. Hence, we have c4. ■

Lemma 2. *The maximum access latency in any feasible solution of 3tVMPDN under the threshold t is at most $3t$.*

Proof: Let assignment A be an arbitrary feasible solution of 3tVMPDN under the threshold t. Because the assignment A satisfies constraints in 3c, the maximum access latency among all pairs of a DN and its assigned VM in the assignment A is not greater than t. It suffices to show the maximum access latency among all pairs of assigned VMs in the assignment A is not greater than $3t$. Consider two arbitrary assigned VMs j and k. Let VM j be assigned to DN i in the assignment A. Then, the access latency between VMs j and k is not greater than the access latency between VM j and DN i (L_1) plus the access latency between DN i and VM k (L_2) because the access latency satisfies the triangle inequality. Since the assignment A satisfies constraints in 3c and 3d, $L_1 \leq t$ and $L_2 \leq 2t$, completing the proof. ■

Lemma 3. *Any feasible solution of tVMPDN is a feasible solution of 3tVMPDN.*

Proof: Let assignment A be an arbitrary feasible solution of tVMPDN. Then, the assignment A must satisfy constraints in 2a, 2b, 2c, 2d, and 2e. Since constraints in 2a, 2b, 2c, and 2e are equal to constraints in 3a, 3b, 3c, and 3e, respectively. We need only to show the assignment A satisfies constraints in 3d. It suffices to show that, in the assignment A, the access latency between an arbitrary assigned VM j and an arbitrary DN i is at most $2t$. Let DN i be assigned to VM k in the assignment A. If VM k is VM j, the access latency between DN i and VM j is at most t because the assignment A satisfies constraints in 2c; otherwise, the access latency between DN i and VM j is not greater than the access latency between DN i and VM k (L_1) plus the access latency between VM k and VM j (L_2) because the access latency satisfies the triangle inequality. Because the assignment A satisfies constraints in 2c and 2d, $L_1 \leq t$ and $L_2 \leq t$, completing the proof. ■

Lemma 4. *Procedure 1 is a polynomial-time algorithm.*

Proof: Clearly, in polynomial time, a bipartite graph can be constructed by Procedure 2 (line 16), and an assignment of DNs to VMs can be made (lines $18-20$). Besides, in polynomial time, a matching in a bipartite graph can be found [13] (line 17), and a feasible LP solution x' for the LP relaxation of the input problem can be obtained by a polynomial-time LP algorithm [14] (line 12). Thus, Procedure 1 can be executed in polynomial time. ■

Theorem 4. *Algorithm 1 is a 3-approximation algorithm of VMPDN.*

Proof: Algorithm 1 executes Procedure 1 at most $|E|$ times. Since Procedure 1 can be executed in polynomial time by Lemma 4, Algorithm 1 can be executed in polynomial time. Let d_{OPT} be the maximum access latency in the optimal solution of VMPDN. We need to show Algorithm 1 can always output an assignment of DNs to VMs with the maximum access latency of at most $3 \cdot d_{OPT}$. Clearly, there is an edge in the input graph G of Algorithm 1 such that d_{OPT} is equal to the weight of the edge. In addition, t is increased from the smallest to the greatest weight of the edges in G. By Theorem 3 and Lemma 2, the maximum access latency in any assignment output by Procedure 1 is smaller than $3 \cdot d_{OPT}$ as $t < d_{OPT}$. Thus, it suffices to show that (c1) Procedure 1 must output a feasible solution of 3tVMPDN as $t = d_{OPT}$. Since d_{OPT} is the maximum access latency in the optimal solution of VMPDN, there is a feasible solution of tVMPDN as $t = d_{OPT}$. By Lemma 3, there is a feasible solution of 3tVMPDN as $t = d_{OPT}$. Thus, by Theorem 3, c1 must hold. ■

IV. Optimal Approximation Algorithm

Theorem 1 indicates that an approximation algorithm with a factor smaller than two does not exist for VMPDN. Here, we propose a 2-approximation algorithm, that is, an optimal approximation algorithm, for VMPDN. The basic idea is to divide the original problem into $|V|$ small problems, compute a solution for each small problem, and subsequently choose the best solution. The steps are described in the following paragraphs.

First, we divide VMPDN into $|V|$ small problems. Each small problem is a *rooted* version of VMPDN, termed rVMPDN. In other words, in rVMPDN, the set of the assigned VMs must contain a specific VM r. Clearly, an algorithm for rVMPDN can be used to solve VMPDN by trying all possible $|V|$ roots. The mixed integer program of rVMPDN can be obtained from that of VMPDN by adding new constraints $\sum_{i \in D} x_{ir} = 1$. Then, we obtain rVMPDN with a fixed threshold t, termed trVMPDN, using the threshold technique. The integer program of trVMPDN can be obtained from that of tVMPDN by adding new constraints $\sum_{i \in D} x_{ir} = 1$.

Next, we need to remove the quadratic constraints in the integer program of trVMPDN, that is, $(\sum_{i\in D} x_{ij^1})(\sum_{i\in D} x_{ij^2})(t - d_{j^1 j^2}) \geq 0, \forall j^1, j^2 \in V, j^1 \neq j^2$, while ensuring that properties P1 and P2, described in Section III-A, are satisfied. Since removing the quadratic constraints could result in unbounded access latency between two assigned VMs, we need to add constraints to ensure that the access latency between any two assigned VMs is bounded by $2t$. In order to do so, we avoid that the access latency between VM r and an arbitrary assigned VM is greater than t by adding new constraints, $(\sum_{i\in D} x_{ij})(t - d_{rj}) \geq 0, \forall j \in V$. Note that because the access latency between any two assigned VMs are bounded by t in trVMPDN, any feasible solution of trVMPDN satisfies the newly added constraints. The integer program of the new problem, termed 2trVMPDN, is described as follows.

$$\text{subject to} \quad \sum_{j\in V} x_{ij} = 1, \ \forall i \in D \tag{4a}$$

$$\sum_{i\in D} x_{ij} \leq 1, \ \forall j \in V \tag{4b}$$

$$x_{ij}(t - d_{ij}) \geq 0, \ \forall i \in D, \forall j \in V \tag{4c}$$

$$(\sum_{i\in D} x_{ij})(t - d_{rj}) \geq 0, \ \forall j \in V \tag{4d}$$

$$\sum_{i\in D} x_{ir} = 1 \tag{4e}$$

$$x_{ij} \in \{0, 1\}, \ \forall i \in D, \forall j \in V \tag{4f}$$

Algorithm 2 2-Approximation Algorithm for VMPDN

Input: A graph $G = (D \cup V, E)$ with weights d_{ab} associated with all $e_{ab} \in E$

35: Index all edges in E in a nondecreasing order of the weights;
36: $m \leftarrow 0$;
37: **while** $m < |E|$ **do**
38: $m \leftarrow m + 1$;
39: $t \leftarrow$ the weight of the edge with index m;
40: $L_{min} \leftarrow 2t + 1$
41: **for** each VM r **do**
42: **if** Procedure 1 with input parameters G, t, and the constraints of 2trVMPDN (i.e. constraints in 4a − 4f) outputs an assignment A with the maximum access latency, L, less than L_{min} **then**
43: $A_{min} \leftarrow A$;
44: $L_{min} \leftarrow L$;
45: **end if**
46: **end for**
47: **if** $L_{min} \leq 2t$ **then**
48: Return the assignment A_{min};
49: **end if**
50: **end while**
51: Return no feasible solution;

The proposed 2-approximation algorithm for VMPDN is described in Algorithm 2. Similar to Algorithm 1, Algorithm 2 increases the threshold t from the smallest to the greatest weight of the edges in the input graph G. Under a certain threshold t, Algorithm 2 divides VMPDN into $|V|$ rVMPDN, computes the assignment of the corresponding 2trVMPDN of each rVMPDN using Procedure 1, and outputs the assignment with the minimum of the maximum access latency if there is an assignment output by Procedure 1 (lines 41 − 49).

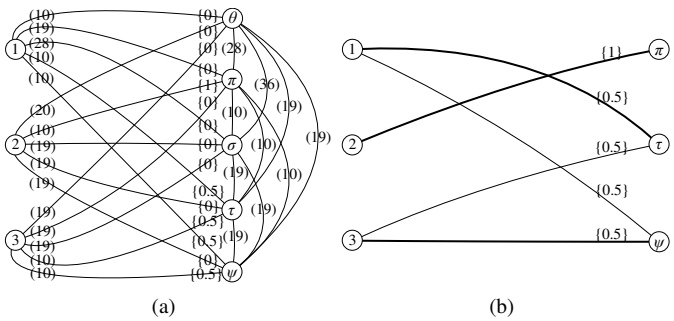

Fig. 2. Example of Algorithm 2. (a) The input graph $G = (D \cup V, E)$. (b) The constructed bipartite graph $G' = (D' \cup V', E')$.

Example 2. Example of Algorithm 2 is illustrated in Fig. 2. Initially, t is set to 10 in line 39. As $r = \theta$, Procedure 1 cannot find a feasible solution in line 42 because $d_{\theta j} > t = 10$ for each $j \in V - \{\theta\}$. As $r = \pi$, a feasible LP solution x' for the LP relaxation of 2trVMPDN is obtained when executing Procedure 1. Note that $x'_{i\theta} = 0$ for each $i \in D$ because $d_{\theta\pi} = 28 > t = 10$ and x' must satisfy constraints in 4d. Algorithm 2 finally assigns DNs 1, 2, and 3 to VMs τ, π, and ψ, respectively. Note that the maximum access latency in the assignment is $19 < 2t = 20$. ■

Clearly, Algorithm 2 executes Procedure 1 at most $|E| \cdot |V|$ times; thus, Algorithm 2 can be executed in polynomial time by Lemma 4. In addition, in any feasible solution of 2trVMPDN under the threshold t, the maximum access latency among all pairs of a DN and its assigned VM is not greater than t (due to constraints in 4c) and the maximum access latency among all pairs of assigned VMs is not greater than $2t$ (because the maximum access latency between VM r and any assigned VM is not greater than t due to constraints in 4d); that is, the maximum access latency in any feasible solution of 2trVMPDN under the threshold t is at most $2t$. Let d_{OPT} be the maximum access latency in the optimal solution of VMPDN. Then, given that $t = d_{OPT}$, at least one of $|V|$ trVMPDN has a feasible solution. Let trVMPDN with a specific VM r' have a feasible solution. Since any feasible solution of trVMPDN is a feasible solution of 2trVMPDN based on the similar arguments of Lemma 3, 2trVMPDN with a specific VM r' has a feasible solution as $t = d_{OPT}$. Thus, as $t = d_{OPT}$ and $r = r'$, Procedure 1 must output a feasible solution of 2trVMPDN with the maximum access latency of at most $2 \cdot d_{OPT}$ using the similar argument of Theorem 3. Since the threshold t is increased from the smallest to the greatest weight of the edges in G, the maximum access latency in the assignment output by Algorithm 2 is bounded by $2 \cdot d_{OPT}$. Thus, we have Theorem 5. According to Theorem 1, Algorithm 2 is an optimal approximation algorithm for VMPDN. We omit the proof of Theorem 5 due to its similarity with that of Theorem 4.

Theorem 5. *Algorithm 2 is a 2-approximation algorithm for VMPDN. Specifically, Algorithm 2 is an optimal approximation algorithm for VMPDN.*

V. The Problem Variant

We study the variation of VMPDN, termed sVMPDN, which is also introduced and shown not to admit any approximation algorithm with a factor smaller than two in [2]. The objective of sVMPDN is to minimize the maximum access latency among all pairs of a DN and its assigned VM while ensuring that the maximum access latency among all pairs of assigned VMs (maximum inter-VM latency) is bounded by a specific value s. In other words, sVMPDN bounds the maximum inter-VM latency at a specific value s, rather than minimizing it. We can obtain the mixed integer program of sVMPDN from that of VMPDN by replacing constraints in 1e with $(\sum_{i\in D} x_{ij^1})(\sum_{i\in D} x_{ij^2})(s - d_{j^1j^2}) \geq 0, \forall j^1, j^2 \in V, j^1 \neq j^2$. Similar to the design of the 2-approximation algorithm for VMPDN, we introduce a new problem, termed tr2sVMPDN, with the integer program described in the following.

$$\text{subject to} \quad \sum_{j\in V} x_{ij} = 1, \ \forall i \in D \tag{5a}$$

$$\sum_{i\in D} x_{ij} \leq 1, \ \forall j \in V \tag{5b}$$

$$x_{ij}(t - d_{ij}) \geq 0, \ \forall i \in D, \forall j \in V \tag{5c}$$

$$(\sum_{i\in D} x_{ij})(s - d_{rj}) \geq 0, \ \forall j \in V \tag{5d}$$

$$\sum_{i\in D} x_{ir} = 1 \tag{5e}$$

$$x_{ij} \in \{0, 1\}, \ \forall i \in D, \forall j \in V \tag{5f}$$

Using a similar argument for sketching the proof of Theorem 5, as the constraints of tr2sVMPDN (i.e., constraints in 5a − 5f), instead of the constraints of 2trVMPDN, are input to Procedure 1, Algorithm 2 can always output an assignment where the maximum access latency among all pairs of a DN and its assigned VM is not greater than t (due to constraints in 5c) and the maximum access latency among all pairs of assigned VMs is not greater than $2s$ (because the maximum access latency between VM r and any assigned VM is not greater than s due to constraints in 5d). Thus, with a slight modification, Algorithm 2 can obtain an optimal solution of sVMPDN through a violation of the maximum inter-VM latency constraint by a factor of at most two, as described in Theorem 6. The proof of Theorem 6 is omitted due to its similarity with those of Theorems 4 and 5.

Theorem 6. *Algorithm 2 is a $(1, 2)$-approximation algorithm for sVMPDN if the constraints of tr2sVMPDN, instead of the constraints of 2trVMPDN, are input to Procedure 1.*

VI. Numerical Result

We conduct the simulations based on four network architectures inside a data center, including Tree, VL2, Fat-Tree, and BCube, discussed in [7]. For each network architecture, we first generate a topology with 1,024 racks, where p_0 and p_1 are set to 16 and 4, respectively, in the construction of Tree, p_0 is set to 32 in the construction of VL2, k is set to 16 in the construction of Fat-Tree, and k and n are set to 1 and 32, respectively, in the construction of BCube. Subsequently, we randomly assign each of the DNs and available VMs to one rack, where the numbers of the DNs and available VMs are 40 and 120, respectively. To study the impact of the distribution of DNs and VMs on the performance of the proposed algorithms, DNs and VMs are assigned to one of the first 16 racks, 64 racks, 256 racks, and 1,024 racks. The access latency between a DN (or VM) and a VM is randomly chosen from the interval $[0.9, 1.1]$ times the estimated access latency between them. The estimated access latency between a DN (or VM) and a VM is evaluated by the sum of the switch delay and the physical link delay [8]. The default size of a TCP packet is 1,500 bytes. Assume that the link bandwidth is 10Gbps. Then, the switch delay of transferring a TCP packet is approximately 1.214 μs [9]. In addition, between the node (DN or VM) and the access switch, between the access switch and the aggregate switch, and between the aggregate switch and the core switch, the link lengths are approximately 10 m, 25 m, and 25 m, respectively, according to [15], and thus, the physical link delay of transferring a TCP packet is approximately 1.207 μs, 1.283 μs, and 1.283 μs, respectively, according to [8], [10].

TABLE I
Maximum Number of Switches between Two Racks

Topology/Rack Interval	$[1, 16]$	$[1, 64]$	$[1, 256]$	$[1, 1024]$
Tree	1	3	3	5
VL2	1	5	5	5
Fat-Tree	3	3	5	5
BCube	1	3	3	3

We evaluate the performance of Algorithms 1 and 2 on the maximum access latency among all pairs of a DN and its assigned VM and all pairs of assigned VMs. In addition, we compare our algorithms with the lower bound of the optimal solution (LB) evaluated by the smallest threshold t such that a feasible LP solution for the LP relaxation of 2trVMPDN can be obtained by Procedure 1. The empirical data is obtained by averaging the data of 100 assignment of racks to DNs and VMs for each network topology. In addition, we also evaluate the worst (maximum) ratios of the maximum access latencies of Algorithms 1 and 2 to LB among 100 assignment of racks to DNs and VMs.

Fig. 3 shows that Algorithm 1 performs as well as Algorithm 2 in terms of the maximum access latency in almost all cases, and that the maximum access latencies of Algorithms 1 and 2 approach LB. In addition, broadly speaking, as the rack range increases, the maximum access latencies of Algorithms 1 and 2 tend to increase in all network architectures. Specifically, both Algorithms 1 and 2 have approximately the same maximum access latency as the rack range is $[1, 64]$ and $[1, 256]$ in Tree; $[1, 64]$, $[1, 256]$, and $[1, 1024]$ in VL2 and BCube; $[1, 16]$ and $[1, 64]$ or $[1, 256]$ and $[1, 1024]$ in Fat-Tree. These observations reveal that the maximum access latencies of Algorithms 1 and 2

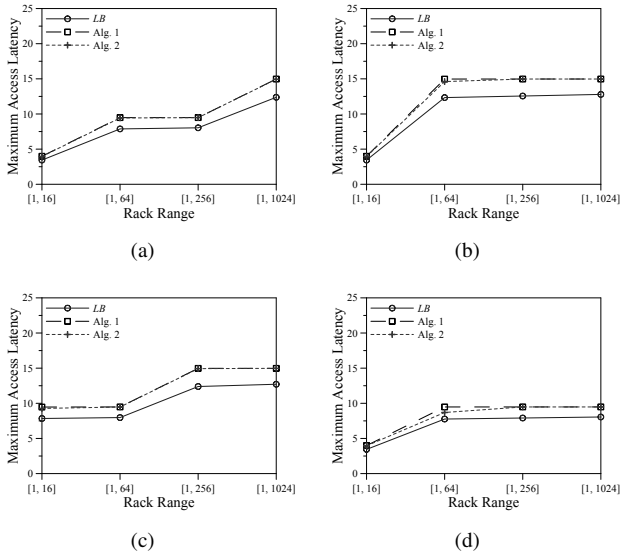

Fig. 3. Impact of the rack range on the maximum access latencies of Algorithms 1 and 2 and LB in the (a) Tree, (b) VL2, (c) Fat-Tree, and (d) BCube architectures.

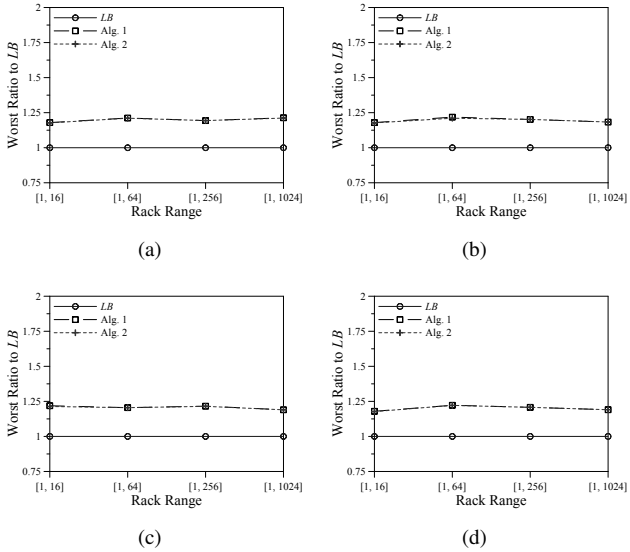

Fig. 4. Impact of the rack range on the worst ratios of the maximum access latencies of Algorithms 1 and 2 to LB in the (a) Tree, (b) VL2, (c) Fat-Tree, and (d) BCube architectures.

are proportional to the maximum number of switches between two racks in these network architectures shown in Table I. This is reasonable because the maximum access latency between two DNs or VMs is dominated by the number of switches rather than the physical link length between them because the difference in the physical link delay of 10 m and 25 m links in the transfer of a TCP packet is only 0.076 μs, whereas the switch delay is approximately 1.214 μs. Moreover, Fig. 4 shows that the worse ratios of the maximum access latencies of Algorithms 1 and 2 to LB range from 1.17 to 1.22.

VII. Related Work

Several assignment methods are proposed in the literature. The generalized assignment problem (GAP) [16] is the first assignment problem with load (or capacity) constraints. GAP is shown to be NP-hard in [17], and an approximation algorithm with a cost not greater than the optimal solution and the capacity of each facility violated by a factor of at most two is proposed in [18]. Unlike GAP, in which the total weights of edges (the total cost) is minimized, [19] and [20] provide the variants of GAP to maximize the total weights of edges (the total profit). In addition, the facility location problem (FLP) has the best approximation ratio $O(logn)$ on the cost that does not obey the distance metric (triangle) inequality (the non-metric cost) [21]. Furthermore, the capacitated FLP [22]–[24], in which the demand and capacity constraint is considered, has a constant approximation ratio on the metric cost. Note that there are no constraints on the distance between any two assigned facilities in GAP, FLP, and their variants; in contrast, constraints are imposed on the access latency between any two assigned VMs in our problem. Thus, the methods of GAP, FLP, and their variants cannot be used to obtain the approximation algorithms for our problem.

In addition, there are several algorithms for VM placement and migration [7], [11], [25]–[29]. In [7], [25], VM placement/migration algorithms are proposed to reduce bandwidth consumption and prevent the network congestion problem by placing the VMs with relatively heavier traffic as close as possible. In [26], the proposed VM migration algorithm considers not only the hop distance between the VMs but also the link capacity such that the network congestion can be mitigated in the routing paths between two communicating VMs. In [11], the authors consider a VM placement problem across data centers (or inside a data center) where the resource of each data center (or rack/machine) is limited. Moreover, assuming that the access latency satisfies the triangle inequality, a 2-approximation algorithm is proposed for minimizing the maximum access latency between the relative VMs. In [27], a VM placement algorithm is introduced to consolidate the live VMs such that the number of the used computation nodes is minimized while ensuring that the probability of the total variable bandwidth consumption of the consolidated VMs exceeding the bandwidth limitation of the computation node is bounded by a factor of user-defined parameter p ($0 < p < 1$). In [28], the authors model the problem of consolidating the VMs according to the resources of computation nodes as an NP-hard problem and resolve this problem by the existing integer program solver or approximation algorithms. However, none of these VM placement/migration algorithms address the access latency between the data and computation nodes; thus, none of them are suitable for our problem. In [2], the access latency between the data and computation nodes is considered first. The authors introduce the problem that has been investigated in this paper. They show the lower bound on the approximation factor of the approximation algorithm for the problem (as the access latency satisfies the triangle inequality); however, the proposed heuristic does not bound the maximum access latency in the assignment generated. In this paper, we fill this gap by providing an approximation algorithm with a

factor equal to the known lower bound.

In [30], the authors argue that the data locality may not be relevant anymore because with 100G or more adapters, the performance bottleneck will be at the disk I/O rather than the network interface. However, unless we can avoid the oversubscription on the link capacities in the network inside a data center, the network bottleneck can cause high access latency for remote data. Moreover, some researchers suggest that disks be replaced with flash memory or even DRAM as the primary storage in the future [31]. In such environments, the performance bottleneck of data access could reemerge in the network interface, and an efficient VM placement algorithm continues to be required for minimizing the access latency between data and computation nodes.

VIII. Conclusion

In this paper, we investigated the placement problem of VMs for DNs, termed VMPDN, and its variant, termed sVMPDN, which are initially introduced and shown not to admit any approximation algorithm with a factor smaller than two in [2]. VMPDN asks for the assignment of DNs to VMs with the minimum maximum access latency among all pairs of a DN and its assigned VM and among all pairs of assigned VMs such that each DN is assigned to a single VM and each VM is not assigned to more than one DN. In sVMPDN, the maximum access latency among all pairs of a DN and its assigned VM is minimized while ensuring that the maximum access latency among all pairs of assigned VMs (maximum inter-VM latency) is bounded. To the best of our knowledge, no approximation algorithms have been proposed for VMPDN and sVMPDN in the literature so far. We initially proposed a 3-approximation algorithm (Algorithm 1), and subsequently proposed a 2-approximation algorithm (Algorithm 2), that is, an optimal approximation algorithm, for VMPDN in the price of higher time complexity. Furthermore, we also showed that with a slight modification, Algorithm 2 can become a $(1, 2)$-approximation algorithm for sVMPDN. In other words, Algorithm 2 can be slightly modified for obtaining an optimal solution of sVMPDN through a violation of the maximum inter-VM latency constraint by a factor of at most two.

We conducted simulations to evaluate the performance of Algorithms 1 and 2 based on four network architectures inside a data center, including Tree, VL2, Fat-Tree, and BCube, discussed in [7]. The simulation results showed that both Algorithms 1 and 2 make approximately optimal assignments of DNs to VMs. In addition, the simulation results also showed that although Algorithm 1 has a worse approximation factor, Algorithm 1 performs as well as Algorithm 2 in terms of the maximum access latency in almost all cases.

References

[1] Applying the cloud to big data storage. [Online]. Available: http://www.appistry.com/sites/default/files/downloads/Applying_the_Cloud_to_Big_Data_Storage.pdf

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

[3] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, pp. 107–113, 2008.

[4] Hadoop. [Online]. Available: http://hadoop.apache.org/

[5] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg, "Quincy: fair scheduling for distributed computing clusters," in *ACM SIGOPS*, 2009.

[6] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, and I. Stoica, "Improving MapReduce performance in heterogeneous environments," in *USENIX OSDI*, 2008.

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

[8] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*, 5th ed. Addison-Wesley Publishing Company, 2009.

[9] Understanding switch latency. [Online]. Available: http://www.cisco.com/en/US/prod/collateral/switches/ps9441/ps11541/white_paper_c11-661939.html

[10] Speed reduction by distance. [Online]. Available: http://www.numion.com/calculators/Distance.html

[11] M. Alicherry and T. Lakshman, "Network aware resource allocation in distributed clouds," in *IEEE INFOCOM*, 2012.

[12] L. Lovász, "On the ratio of optimal integral and fractional covers," *Discrete Math.*, vol. 13, pp. 383–390, 1975.

[13] L. Lovász and M. D. Plummer, *Matching Theory*. Elsevier Science Ltd, 1986.

[14] R. Monteiro and I. Adler, "Interior path following primal-dual algorithms. part I: Linear programming," *Math. Program.*, vol. 44, pp. 27–41, 1989.

[15] Data centre link lengths. [Online]. Available: http://www.ieee802.org/3/NGBASET/public/nov12/flatman_01a_1112_ngbt.pdf

[16] G. T. Ross and R. M. Soland, "A branch and bound algorithm for the generalized assignment problem," *Math. Program.*, vol. 8, pp. 91–103, 1975.

[17] M. L. Fisher, R. Jaikumar, and L. N. Van Wassenhove, "A multiplier adjustment method for the generalized assignment problem," *Manage. Sci.*, vol. 32, pp. 1095–1103, 1986.

[18] D. B. Shmoys and E. Tardos, "An approximation algorithm for the generalized assignment problem," *Math. Program.*, vol. 62, pp. 461–474, 1993.

[19] C. Chekuri and S. Khanna, "A PTAS for the multiple knapsack problem," in *ACM-SIAM SODA*, 2000.

[20] L. Fleischer, M. X. Goemans, V. S. Mirrokni, and M. Sviridenko, "Tight approximation algorithms for maximum general assignment problems," in *ACM-SIAM SODA*, 2006.

[21] D. S. Hochbaum, "Heuristics for the fixed cost median problem," *Math. Program.*, vol. 22, pp. 148–162, 1982.

[22] M. Pál, E. Tardos, and T. Wexler, "Facility location with nonuniform hard capacities," in *IEEE FOCS*, 2001.

[23] R. Levi and D. B. Shmoys, "LP-based approximation algorithms for capacitated facility location," in *ACM-SIAM SODA*, 2004.

[24] M. H. Bateni and M. T. Hajiaghayi, "Assignment problem in content distribution networks: Unsplittable hard-capacitated facility location," in *ACM-SIAM SODA*, 2009.

[25] V. Shrivastava, P. Zerfos, K.-W. Lee, H. Jamjoom, Y.-H. Liu, and S. Banerjee, "Application-aware virtual machine migration in data centers," in *IEEE INFOCOM*, 2011.

[26] J. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint VM placement and routing for data center traffic engineering," in *IEEE INFOCOM*, 2012.

[27] D. Breitgand and A. Epstein, "Improving consolidation of virtual machines with risk-aware bandwidth oversubscription in compute clouds," in *IEEE INFOCOM*, 2012.

[28] H. Yanagisawa, T. Osogami, and R. Raymond, "Dependable virtual machine allocation," in *IEEE INFOCOM*, 2013.

[29] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Almost optimal virtual machine placement for traffic intense data centers," in *IEEE INFOCOM*, 2013.

[30] G. Ananthanarayanan, A. Ghodsi, S. Shenker, and I. Stoica, "Disk-locality in datacenter computing considered irrelevant," in *USENIX HotOS*, 2011.

[31] J. Ousterhout, P. Agrawal, D. Erickson, C. Kozyrakis, J. Leverich, D. Mazières, S. Mitra, A. Narayanan, D. Ongaro, G. Parulkar, M. Rosenblum, S. M. Rumble, E. Stratmann, and R. Stutsman, "The case for RAMCloud," *Commun. ACM*, vol. 54, pp. 121–130, 2011.