Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 9084370, 15 pages
http://dx.doi.org/10.1155/2016/9084370

Research Article

A Unified Algorithm for Virtual Desktops Placement in Distributed Cloud Computing

Jiangtao Zhang,[1,2] Lingmin Zhang,[1,3] Hejiao Huang,[1,3] Xuan Wang,[1,4] Chonglin Gu,[1,3] and Zhixiang He[1,3]

[1] *School of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen 518055, China*
[2] *Public Service Platform of Mobile Internet Application Security Industry, Shenzhen 518057, China*
[3] *Shenzhen Key Laboratory of Internet of Information Collaboration, Shenzhen 518055, China*
[4] *Shenzhen Applied Technology Engineering Laboratory for Internet Multimedia Application, Shenzhen 518055, China*

Correspondence should be addressed to Xuan Wang; wangxuan@cs.hitsz.edu.cn

Received 7 October 2015; Revised 7 December 2015; Accepted 16 December 2015

Academic Editor: Kavun Sergii

Copyright © 2016 Jiangtao Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Distributed cloud has been widely adopted to support service requests from dispersed regions, especially for large enterprise which requests virtual desktops for multiple geodistributed branch companies. The cloud service provider (CSP) aims to deliver satisfactory services at the least cost. CSP selects proper data centers (DCs) closer to the branch companies so as to shorten the response time to user request. At the same time, it also strives to cut cost considering both DC level and server level. At DC level, the expensive long distance inter-DC bandwidth consumption should be reduced and lower electricity price is sought. Inside each tree-like DC, servers are trying to be used as little as possible so as to save equipment cost and power. In nature, there is a noncooperative relation between the DC level and server level in the selection. To attain these objectives and capture the noncooperative relation, multiobjective bilevel programming is used to formulate the problem. Then a unified genetic algorithm is proposed to solve the problem which realizes the selection of DC and server simultaneously. The extensive simulation shows that the proposed algorithm outperforms baseline algorithm in both quality of service guaranteeing and cost saving.

1. Introduction

Distributed cloud computing has been widely adopted to support service requests from dispersed regions by exploiting the differences of locations and various service capabilities. As one of the most promising services in cloud computing, Virtual desktop technology [1, 2] facilitates users accessing virtual machines (VMs) [3] named as virtual desktops (VDs), deployed in remote data centers (DCs) by the local thin clients. Compared with traditional personal computers or desktops, the local clients are equipped with less resources and there is no data or file saved in local clients, hence reducing the initial investment greatly and also realizing better confidentiality. Moreover, it can be more green [4]. VDs have attracted the interest of many CSPs such as Microsoft [5], VMware [6], Huawei [7], and other traditional telecom operators [8]. It has been widely used around the world [7].

The service delivery scheme is depicted in Figure 1. VDs are placed in geodistributed DCs closer to the branch companies (BCs). Users in each BC access cloud services through one specified gateway by using local thin clients. Because of the relatively small scale of distributed DC or the specific availability policy [9] (e.g., an upper limit of VMs of one BC is designated in one DC), it is possible that more than one DC is used to accommodate the VDs for a big BC [10]. The distributed DCs are often connected by dedicated high-speed links or expensive long distance links. Inside each DC, servers are normally networked in a tree-like topology. In the context of infrastructure as a service, VD is corresponding to VM (in this paper, we use servers and physical machines (PMs), VDs,

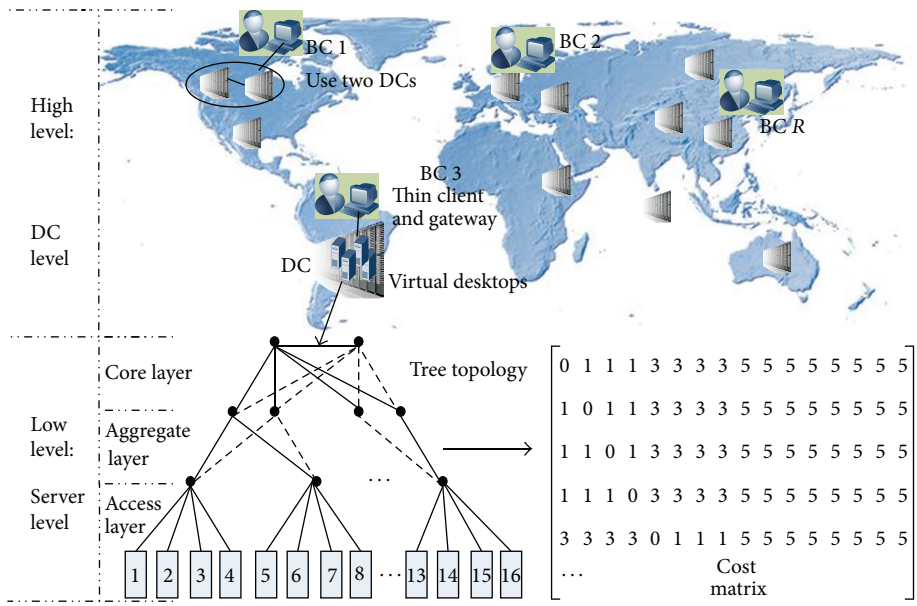


Figure 1: Virtual desktops in worldwide distributed cloud computing.

and VMs interchangeably). The traffic delay inside DC is mainly dominated by switches or routers in the path. The cost matrix [11] in Figure 1 demonstrates the number of switches between each PM for a tree-like topology, such as PortLand, VL2, and BCube.

CSP faces great challenges in VD service delivery. On the one hand, it should guarantee the service level agreement. Especially for the latency sensitive VDs, it should shorten the response time on the premise of keeping the delay within the threshold agreed. The delay consists of three parts. The first is the time between thin client and the gateway. The second is the time between the gateway and DC. The third is the queueing delay inside DC. Now that the first is fixed for each BC once the gateway is specified, so we consider the latter twos. For the second one, we only consider a one-way delay. Delay optimization drives the CSP to deploy VMs in DC closest to each BC. But maybe the closest one is not the best candidate from the economic perspective.

On the other hand, aiming to make maximum profit, CSP also strives to reduce cost as much as possible. At DC level, appropriate DC should be selected to pursue cheaper electricity price and less long distance inter-DC links consumption. Sometimes competition will occur where adjacent BCs compete for the resource of one DC. It complicates matters even more. Inside each DC, VMs should be consolidated and stay together so that fewer PMs are used, power is saved, and less inter-PM bandwidth is consumed. Because nodes, bandwidth, and power comprise 75% of DC cost [12], they are the prime consideration to optimize.

When CSP selects DCs with cheaper electricity price and closer to BC, it must consider the service type, resource capability of PMs inside each DC. The normal two-phase scheme selects DCs according to power and location objective, then determines PMs, and assigns VMs to the hosts. It is a natural but maybe not a good strategy, because the determination of DCs has limited the candidates of PMs to a large extent. Inside each DC, the assignment of the VMs to these PMs cannot guarantee the optimal overall cost. So the decision of DCs is also subject to the assignment of VMs to proper PMs inside each DC. In nature, there is a noncooperative relation which leads to a bilevel structure. Each level pursues different objectives. Some are consistent and others are not (it is detailed in Section 3). CSP should balance quality with cost by considering both DC and server selection simultaneously.

To the best of our knowledge, our work is the first to explore unified algorithms for multi-BC virtual desktops placement. The main contributions are as follows:

(1) We formulate the problem as multiobjective bilevel programming considering resource provision at both DC and server level.

(2) A novel unified algorithm, segmented two-level grouping genetic algorithm (STLGGA), is proposed. It can realize the selection of DC and server simultaneously. It also minimizes the response delay between VDs and BCs at the least cost consisting of bandwidth, server nodes, and power.

(3) Extensive simulation demonstrates that STLGGA outperforms the baseline algorithm for both multi-BC and single BC scenarios. For multi-BC, it achieves a 13% shorter delay, while saving power and resources by 21% and 6% on average, respectively.

The remainder of the paper is organized as follows. Related work is reviewed in Section 2. Section 3 formulates the problem and Section 4 presents a novel GA. It is evaluated in Section 5 and the whole paper is concluded in Section 6.

2. Related Work

Various objectives of virtual desktops placement have been explored recently. Most works take into consideration the resource efficiency inside DC [2, 14–16]. Man and Kayashima [14] use bin packing (BP) method to find the group of VDs which is suitable for a server so that minimal hosts are used. Makarov et al. [15] propose a tool for evaluating VDs. It focuses on assessing the effect of VD access protocol so as to guide the resource configuration according to the reaction time of the task execution. Deboosere et al. [2] study different aspects to optimize resources and user satisfaction. The resource requirement is predicted first, and then overbooking category is adopted and resource allocation and resource reallocation are used to achieve load balance or energy efficiency. This method provides a VDs resource management framework. VDs secure sharing is explored by [16]. But, in all these works, no location of VDs and inter-DC resource are considered.

Other works investigate VDs deployment across DCs. Kochut [17] prefers placing VDs across DCs in different location to lower the power consumption. But no service quality is touched. OpenFlow based mechanism is proposed to distribute VDs in multiple DCs so that the performance and scalability are maximized [18]. However, the authors focus on route setup, route selecting, and load balance between thin client and DC. The most similar work to ours is [1]. Latency sensitive VDs are optimized by exploiting the geodistributed DCs location. A greedy algorithm VMShadow is presented by migrating VD to its user while considering the overhead of migration. But it does not establish mapping between VDs and PMs. Furthermore, the resources at DC level and inside DC as well as energy cost are not involved.

Only seldom papers [13, 19] consider the selection of DC and server simultaneously when placing VMs. Yao et al. [19] propose a two-time-scale Lyapunov optimization algorithm to reduce power cost for delay tolerant workloads. A multilevel group GA (MLGGA) is proposed to reduce carbon emission by exploiting green energy [13]. The main idea is DC consolidation and PM consolidation. Herein, the distributed DCs are viewed as the higher level group and servers in DC as lower level group. This scheme can group the items and is designed for multilevel BP. But it does not consider bandwidth optimization and quality of service guarantee. Both of them focus on the general VMs. Calyam et al. [10] present a utility driven model U-RAM to allocate resources for VDs so that the utility in each DC is maximized. The resource should be enough to ensure the timeliness user perceived and coding efficiency of VD access protocol. It adopts a two-phase scheme: the DCs are selected based on balance, power, or express migration. Then VMs are assigned to PMs in the selected DC. Inter-DC bandwidth cost is not considered. The overall resource cost in the two levels cannot be optimized integratedly. All works cannot capture the noncooperation relation between DC network and servers.

3. Formulation

Suppose there are R BCs and each requires I_r, $r = 1, 2, \ldots, R$ VMs, and $\sum_{r=1}^{R} I_r = I$. There are K candidate DCs. In each DC k, $k = 1, 2, \ldots, K$, there are J_k PMs. The total number of PMs in all DCs is J; that is, $\sum_{k=1}^{K} J_k = J$. The cost and quality aware multi-BC virtual desktops placement problem can be summarized as placing I VMs belonging to R BCs on J PMs which distribute in K DCs, so that the maximum distance between DCs and BCs being served is as short as possible, while minimizing the overall cost at both DC and server level, including power, network, and server. The problem is modeled as multiobjective bilevel programming (MOBLP).

3.1. Low Level Objective and Constraints. Low level is server level. It aims to select PMs in DC determined by high level (DC level, as illustrated in Figure 1). We will place I_k VMs in DC k. Note that maybe the VMs serving different BCs will be placed in one DC. The number of VMs is fixed; that is, $\sum_{k=1}^{K} I_k = \sum_{r=1}^{R} I_r$. Each VM i (V_i) requires H kinds of resources $V_i^1, \ldots, V_i^H$, such as network bandwidth, CPU, memory, and disk storage. P_j^k denotes PM j in DC k. It possesses H kinds of normalized resources $P_j^{k1}, \ldots, P_j^{kH}$ where $P_j^{kh} \in (0, 1]$ (the normalized weight for dimension h is $\omega_h = 1/(\max_{j,k} P_j^{kh})$). Specifically, we designate the first dimension resource as bandwidth (P_j^{k1} and V_i^1) and the second one as CPU (P_j^{k2} and V_i^2) for ease of notation. The communication traffic rate between V_h and V_o is t_{ho} and t_{hh} equals 0. Herein we suppose $t_{ho} = t_{oh}$ in that a bidirectional link normally has equal bandwidth in each direction. If $t_{ho} \neq t_{oh}$ we select the biggest one as the traffic. x_{ij}^k and y_j^k are both Boolean variables controlled by low level. x_{ij}^k indicates whether V_i is assigned to P_j^k. It equals 1 if V_i is assigned to P_j^k and 0 otherwise. y_j^k indicates whether P_j^k is active. It equals 1 if P_j^k is active and 0 otherwise.

The low level only considers resource cost of PM nodes (the former half part of $f(y, x)$) and bandwidth (the latter half part) inside each DC k:

$$f(y, x) = \sum_{j=1}^{J_k} y_j^k \sum_{h=2}^{H} c_h P_j^{kh} + \sum_{l=1}^{3} \sum_{j_1=1}^{J_k} \sum_{j_2=1}^{J_k} b_l \delta_l \left(P_{j_1}^k, P_{j_2}^k\right), \quad (1)$$

where c_h is the price for resource h. h is summed from 2 because the bandwidth cost has been calculated explicitly in the latter part. Multiplier y_j^k means that only cost of active PMs needs to be considered. b_l is the lth layer bandwidth price, where $l = 1, 2, 3$ represents bandwidth in access, aggregate, and core layer (Figure 1), respectively. Normally, $b_1 < b_2 < b_3$. $\delta_l(P_{j_1}^k, P_{j_2}^k)$ is the lth layer bandwidth consumption and defined as the sum of the lth layer traffic of VMs across PMs (2). So it equals zero for the same PMs (3). Consider

$$\delta_l \left(P_{j_1}^k, P_{j_2}^k\right) = \sum_{V_h \in P_{j_1}^k} \sum_{V_o \in P_{j_2}^k} t_{ho}^l \quad j_1 \neq j_2,\ l = 1, 2, 3, \quad (2)$$

$$\delta_l \left(P_{j_1}^k, P_{j_2}^k\right) = 0 \quad j_1 = j_2,\ l = 1, 2, 3. \quad (3)$$

In each PM, the resource capacity should be respected. Because if VMs are placed in one PM, then the inter-VM traffic is changed to intra-PM traffic and no outgoing bandwidth is occupied; the intra-PM traffic is subtracted. The bandwidth constraint is as follows:

$$\sum_{i=1}^{I_k} x_{ij}^k V_i^1 - \sum_{h=1}^{I_k}\sum_{o=1}^{I_k} x_{hj}^k x_{oj}^k t_{ho} \le P_j^{k1} \quad j = 1, \ldots, J_k. \tag{4}$$

The other resources constraint is

$$\sum_{i=1}^{I_k} x_{ij}^k V_i^h \le P_j^{kh} \quad h = 2, \ldots, H, \ j = 1, \ldots, J_k. \tag{5}$$

For each DC k, the low level programming is written as

$$\begin{aligned} \min_{y,x} \quad & (1) \\ \text{s.t.} \quad & (2), (3), (4), (5), \end{aligned} \tag{6}$$

$$y_j^k \ge \frac{\sum_{i=1}^{I_k} x_{ij}^k}{I_k} \quad j = 1, \ldots, J_k, \tag{7}$$

$$y_j^k \in \{0, 1\} \quad j = 1, \ldots, J_k, \tag{8}$$

$$\sum_{j=1}^{J_k} x_{ij}^k = 1 \quad i = 1, \ldots, I_k, \tag{9}$$

$$x_{ij}^k \in \{0, 1\} \quad i = 1, \ldots, I_k, \ j = 1, \ldots, J_k. \tag{10}$$

Constraint (7) states that a PM is viewed as active if it hosts at least one VM. Constraint (9) implies that all VMs should be assigned and a VM can only be placed in one PM.

3.2. High Level Objectives and Constraints. High level focuses on DC selection. It optimizes the delay, the overall power, and physical resource cost. Binary variable z_{kr} indicates whether there exists a PM which is used to serve BC r and the PM is in D_k. If it is true then $z_{kr} = 1$ and 0 otherwise. Once there exists one active PM in D_k, the D_k is viewed as active. So we have

$$\sum_{r=1}^{R} z_{kr} \ge \frac{\sum_{j=1}^{J_k} y_j^k}{J_k} \quad k = 1, \ldots, K. \tag{11}$$

Suppose the one-way delay between D_k and BC r being served is d_{kr}. It can be estimated by some extensively studied work [20]. We hope to ensure that all the delays are within a threshold ρ' which is the maximum delay permitted; that is,

$$z_{kr} d_{kr} \le \rho \le \rho' \quad r = 1, \ldots, R, \ k = 1, \ldots, K. \tag{12}$$

Multiplier z_{kr} means that we only care about the delay between active DC and the BC being served. At the same time we want to reduce the delay to a minimum. This is the first objective:

$$F_1(z, y, x) = \rho. \tag{13}$$

The second objective aims to optimize power cost of all the selected DCs by leveraging the geodiverse electricity price:

$$F_2(z, y, x) = \sum_{k=1}^{K} e_k \sum_{j=1}^{J_k} y_j^k \left(a \sum_{V_i \in P_j^k} V_i^2 + p_j^k \right), \tag{14}$$

where e_k is electricity price of D_k. a is the coefficient to reflect the relation between power and CPU load. p_j^k is the power consumption of P_j^k in idle or standby state. Because power grows largely positively proportional to CPU utilization [21], we use an affine function of CPU load ($\sum_{V_i \in P_j^k} V_i^2$) to estimate power cost. To make the power consumed and physical resource cost comparable, we follow the way of [12]. All the one-time purchased physical resource cost is amortized in a reasonable lifetime. So all the physical resource prices in the formulation are the amortized ones. Implicitly, in the formulation, we only balance the cost in the amortized period. The former half of $F_2(z, y, x)$ represents power cost caused by workload and the latter half is power in idle or standby state.

The third objective is the overall resource costs including PMs and bandwidth:

$$\begin{aligned} F_3(z, y, x) = & \sum_{k=1}^{K} \sum_{j=1}^{J_k} y_j^k \sum_{h=2}^{H} c_h P_j^{kh} \\ & + \sum_{k=1}^{K} \sum_{l=1}^{3} \sum_{j_1=1}^{J_k} \sum_{j_2=1}^{J_k} b_l \delta_l \left(P_{j_1}^k, P_{j_2}^k \right) \\ & + \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} b \delta \left(D_{k_1}, D_{k_2} \right). \end{aligned} \tag{15}$$

The former two parts are consistent with the objective of low level ($f(y, x)$). However, low level only considers resources inside each DC. High level considers the inter-DC bandwidth cost in the third part additionally. Inter-DC bandwidth definition is similar to inter-PM bandwidths (2), (3):

$$\delta \left(D_{k_1}, D_{k_2} \right) = \sum_{V_h \in D_{k_1}} \sum_{V_o \in D_{k_2}} t_{ho} \quad k_1 \ne k_2, \tag{16}$$

$$\delta \left(D_{k_1}, D_{k_2} \right) = 0 \quad k_1 = k_2. \tag{17}$$

The high level optimization can be summarized as multi-objective programming (MOP):

$$\begin{aligned} \min_{z,y,x} \quad & (13), (14), (15) \\ \text{s.t.} \quad & (2), (3), (11), (12), (16), (17), \end{aligned} \tag{18}$$

$$z_{kr} \in \{0, 1\} \quad r = 1, \ldots, R, \ k = 1, \ldots, K, \tag{19}$$

$$y_j^k \in \{0, 1\} \quad j = 1, \ldots, J_k, \ k = 1, \ldots, K, \tag{20}$$

where x and y are the solutions of the low level programming.

Obviously, each level has its own objectives and constraints. The high level objective value depends on the optimal

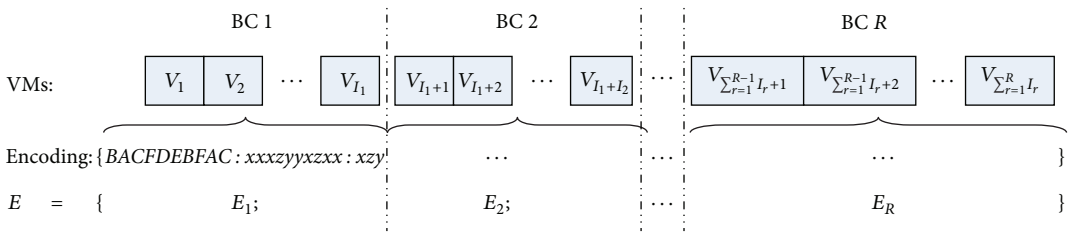


Figure 2: Segmented two-level grouping encoding.

solutions of the low level. Needless to say the first two parts of F_3 which are just what was pursued by the low level, the inter-DC bandwidth (F_3) and the overall power (F_2), are both subject to the optimal solution of the low server level. Inter-DC bandwidth consists of the traffic between VMs across DCs (16). The overall power is directly related to the CPU load and how many PMs are used. The low level optimizes itself under the determined high level decision variable (DC is determined by high level). In particular, there is a noncooperation relation between DC level and server level. For example, each DC only tries to minimize its resource consumed, but sometimes it is contrary to minimizing the inter-DC traffic because minimizing inter-DC traffic may need to consume more PM resources. This is just the case described by bilevel programming [22].

Note that the bandwidth related constraints, (2), (4), and (16) are nonlinear, MOBLP is nonlinear bilevel programming. Even linear bilevel programming, the simplest one of bilevel programming problems, is proved to be strong NP-hard. The problem formulated herein is NP-hard. GA has been demonstrated as a very efficient scheme to address bilevel programming [23, 24]. We resort to GA to solve it.

4. Algorithm

For minimization programming, suppose there are two vectors, x and y, with the same dimension n; we say x dominates y if and only if $x_i \leq y_i$ for any i and there exists at least $j \in 1, 2, \ldots, n$ so that $x_j < y_j$. Suppose the feasible solution set of multiobjective programming (MOP) is $\mathcal{F}$; then a solution x^* is Pareto optimal to MOP if there is no point $x \in \mathcal{F}$, so that $F(x)$ dominates $F(x^*)$, where $F = \{F_1, F_2, F_3\}$. In short, any decrease in one dimension of F must lead to the increase of at least one other dimension. We try to find the multiple Pareto solutions for MOBLP.

Distinctive encoding and decoding, initial population generation scheme, and genetic operators are designed to address the multiobjective bilevel resource provision problem.

4.1. Encoding and Decoding Scheme. In GA, it is very important to reflect the structure and information of the problem to be optimized in the genes of chromosome (it is assumed that readers are familiar with the structure of GA, otherwise please refer to [25] for details). Considering the characteristic of multi-BC and two levels of *MOBLP*, we propose a segmented two-level grouping encoding scheme as depicted in Figure 2. The entire candidate PMs and DCs are numbered first. The encoding gives the serial number (SN) of PM to which each VM is assigned and DC to which each PM belongs. It consists of R segments in series and is encoded as $E = E_1; E_2; \ldots; E_R$. Segments are separated by a semicolon and E_r corresponds to BC r for $r = 1, 2, \ldots, R$. The structure of E_r is the same as the encoding of MLGGA [13]. It comprises three parts. The first is SN of PMs used. The second is SN of DC to which each PM belongs. The third is DC in the second part after deleting the repeated ones. The three parts are isolated by a colon. For example, suppose there are DCs x, y, z, PMs $A \sim F$; the relation of belongingness of the PM and DC is as follows: $x = A, B, C$, $y = D, E$, $z = F$. BC 1 requires $I_1 = 10$ VMs and they are assigned to PM *BACFDEBFAC*. The corresponding DCs are *xxxzyyxzxx*. They are abbreviated as *xyz* by deleting the latter repeated ones. Then E_1 is *BACFDEBFAC*: *xxxzyyxzxx*: *xyz*.

The new encoding scheme lists the PMs and DCs to which the VMs of each BC are assigned. It can capture the placement of VMs of multiple BCs as a whole and facilitate the competitive scenario resolution. Therefore, better solutions can be found. It also remedies the incompetency of MLGGA which can only place VMs of one BC one time and achieves better performance as revealed in Section 5. The decoding is self-evident.

Distinctive initial population generation scheme and genetic operators are designed to address the multiobjective bilevel VDs placement problem.

4.2. Initial Population Generation. The Pareto solution aims to optimize each scalar objective F_i. So we strive to embody the optimum of each F_i in the initial population. Solutions for the solo member objective are produced so that the initial population has a rather good gene to be inherited by the offspring.

For F_1, the shortest delay VM placement algorithm (SD) is proposed in Algorithm 1. For BC r, only those DCs, delays between them, and BC r which do not exceed ρ are considered. Denote those candidate DCs for BC r as feasible DC set $\mathcal{F}_r$. For each BC, SD prefers placing VMs of this BC in closer DC in $\mathcal{F}_r$ until all VMs are assigned. This procedure is repeated for all BCs.

In Algorithm 1, we can replace the sorting criterion d_{kr} with electricity price e_k. Then we have another method which strives to place VDs in feasible DC with the lowest electricity price. We denote it as LeastPowerCost (LPC). LPC aims to optimize F_2.

Input: R: numbers of BCs
$\mathscr{F}_r$: feasible DC set for BC r and $|\mathscr{F}_r| = n_r$
V^r: VMs set for BC r
Output: VM placement solution encoding X
(1) **for** $r = 1, \ldots, R$ **do**
(2) Sort DCs in $\mathscr{F}_r$ according to d_{gk}. Closer DC is selected with higher probability and denote the sorted sequence as $D_1, D_2, \ldots, D_{n_r}$
(3) **for** $k = 1, \ldots, n_r$ **do**
(4) Randomly select non-assigned VM from V^r and put it into a random non-full PM in D_k until D_k is full or reach the upper limit
(5) $V^r \leftarrow (V^r \backslash$VMs assigned to $D_k)$
(6) **if** $|V^r| = 0$ **then**
(7) break
(8) **end if**
(9) **end for**
(10) **end for**
(11) encoding the solution as X according to Section 4.1

Algorithm 1: Shortest delay VM placement algorithm (SD).

For F_3, a modified first fit decreasing algorithm (MFFD, Algorithm 2) which takes into account both nodes and inter-PM bandwidth optimization is depicted in Algorithm 2. It strives to place VDs in the *smallest-cost*-cluster with the largest capacity in a DC which is the largest one in $\mathscr{F}_r$. The communication cost is defined as the number of switches or routers in the path [11]. The PMs with the same c cost are named as c-cluster. There are three kinds of clusters in topology in Figure 1. For example, PM1~PM4 is a 1-cluster. PM9~PM16 is a 3-cluster. PM5~PM8 and PM9~PM11 are both 5-clusters. Each time, we prefer the largest capacity cluster with the smallest cost, because bigger cost means more aggregate links and core links will be used. The overconsumption of these relatively scarce top layer links may further lead to congestion and communication delay. This tactic can reduce consumption of the higher layer bandwidth. In the process of placement, once a c-cluster is selected, another cluster with the same cost c will be selected in priority if both clusters can constitute a *second-smallest-cost*-cluster. This can ensure the effect of consolidation and save more links between clusters. For example, if PM1~PM4 is the cluster with the biggest capacity in all 1-cluster, it will be selected first and then PM5~PM8 cluster is considered unconditionally because they constitute a 2-cluster.

The capacity of a PM P_j^k is defined as that in [26]:

$$\text{cap} = \sum_{h=1}^{H} \omega_h P_j^{kh}, \tag{21}$$

that is, the sum of all the resources dimensions of all the PMs in the cluster. Or

$$\text{cap} = \sum_{h=1}^{H} \omega_h r_j^{kh}, \tag{22}$$

if there is at least one VM in this PM, where ω_h is the normalized factor for dimension h. r_j^{kh} is the h-dimension residue capacity of PM P_j^k. The reason is that, in the computer, if any dimension is used up, then the PM cannot support any more VMs. The corresponding cluster capacity is defined as the sum of the capacities of PMs in the cluster and similar for the capacity of DC. Capacity of VM V_i is similar to (21) except that P_j^{kh} is replaced by V_i^h.

MFFD strives to place VDs in the *smallest-cost*-cluster with the largest capacity in the largest DC chosen from $\mathscr{F}_r$.

MFFD, SD, and LPC will be invoked x times, respectively, to produce $3 * x$ initial feasible solutions. This scheme can produce a rather large initial population and the three groups of population embody a relatively good assignment for the three objectives, respectively. Thus, the initial parents are endowed with some optimal property. In the latter crossover and mutation, though the initial solution will be replaced by a new one which dominates this solution, the size of the initial population remained at least $3 * x$ so that the GA can converge faster.

Lines (2)–(21) of MFFD describe the placement scheme for one BC. We denote them as MFFDOneB. Compared with line (2) where DC with a larger capacity is preferred, in MFFDOneB, the DC, which has the smallest residue capacity and has been used, has a higher priority for selection, so as to take full advantage of the residue capacity and reduce the number of DCs used. Lines (4)–(21) of MFFD describe the placement scheme inside one DC. We denote then as MFFDOneDC. MFFDOneDC mainly works for f, while SD, LPC, MFFD, and MFFDOneB work for F_i $(i = 1, 2, 3)$.

4.3. Crossover Operator. The crossover is applied to the segment one by one and every segment can produce an effect on other segments if the BCs corresponding to these segments compete for the same DC. Crossover operator is depicted in

Input: R: numbers of BCs
$\mathcal{F}_r$: feasible DC set for BC r and $|\mathcal{F}_r| = n_r$
C: cost matrix of each DC as illustrated in Figure 1
V^r: VMs set for BC r
Output: solution encoding X
(1) **for** $r = 1, \ldots, R$ **do**
(2) Sort DCs in $\mathcal{F}_r$ according to their capacity. Bigger DC is selected with higher probability and denote the sorted sequence as $D_1, D_2, \ldots, D_{n_r}$
(3) **for** $k = 1, \ldots, n_r$ **do**
(4) **while** ($|V^r| \neq 0$) and (exists P_j^k not be searched) **do**
(5) Find *smallest-cost*-PM cluster with the largest capacity. Suppose there are total m_k PMs in this cluster
(6) Sort these PMs in non-increase capacity (21) and (22) order as $P_1^k, P_2^k, \ldots, P_{m_k}^k$
(7) **for** $j = 1, \ldots, m_k$ **do**
(8) Select the biggest non-assigned VM from V^r and put it into P_j^k until P_j^k is full
(9) $V^r \leftarrow (V^r \backslash$VMs assigned to $P_j^k)$
(10) **if** $|V^r| = 0$ **then**
(11) break
(12) **end if**
(13) **end for**
(14) **end while**
(15) **if** $|V^r| = 0$ **then**
(16) break
(17) **end if**
(18) **end for**
(19) **if** ($|V^r| \neq 0$) and (all P_j^k is searched) **then**
(20) There is overflow, return FAILURE
(21) **end if**
(22) **end for**
(23) Encoding the solution as X according to Section 4.1

Algorithm 2: Modified first fit decreasing algorithm (MFFD).

Input: X, Y: two individuals
Output: X', Y': two individuals after crossover
(1) **for** $r = 1, \ldots, R$ **do**
(2) $E_r(X)$ crossover with $E_r(Y)$ according to MLGGA [13] with the exception that when "competition" occurs then the upper described scheme is used
(3) **end for**

Algorithm 3: Crossover operator.

Algorithm 3. $E_r(X)$ denotes the segment corresponding to BC r in encoding X. For each segment, the mechanism of the crossover operator in MLGGA is adopted. But to capture the scenario of multi-BC, we propose twofold exceptions.

First, the classic FFD used in MLGGA is replaced by MFFDOneB when placing VMs of one BC or MFFDOneDC when placing some VMs inside one DC, respectively.

Second, a new technique is recommended to address the "competition" case in the multi-BC scenario. In MLGGA, it tries to inherit the property of the parent and keep the VMs in the inserted group (PM or DC) unchanged. So it will clear the different resident VMs in the same group in the target chromosome in advance and keep the common VMs. For example, there are two chromosomes, P_1: *bacfdebfac*: *XXXZYYXZXX*: $X|Y|Z$, where DC Z contains VM 4 and VM 8, and P_2: *BFCFDBBCAE*: *xzxzyxxxxy*: $x|z|y$, where DC z contains VM 2 and VM 4. Here the same alphabet with different case represents the same PM or DC. Therefore, z and Z indicate the same DC. | means crossover point. For target chromosome P_1, when crossover operates, y will be replaced by Z. Now there are two same DC z and Z in the offspring of P_1. So VMs in z should be cleared so that VMs in Z are kept unchanged; that is, VM 8 in z will be reassigned by FFD and VM 4 is preserved. But in the multi-BC scenario, maybe z contains many VMs of other BCs. So when there are too many VMs in Z, z has not enough residue capacity to host these VMs. This is just the case of "competition" where BCs compete for the same PM or DC. We propose the competition

Input: X: input individual
$\mathcal{F}_r$: feasible DC set for BC r and $|\mathcal{F}_r| = n_r$
Output: X': individual after mutation
(1) **for** $r = 1, \ldots, n_r$ **do**
(2) Randomly select one DC k in the third part of $E_r(X)$ and change it to one DC with lower electricity price with higher probability in $\mathcal{F}_r$, denotes as k'
(3) **if** (k' is used by r) or (k' is used by another BC (such as r')) or (k' is shared by BC r and BC r') **then**
(4) Clear k'. Assign VMs of BC r in k to k' by MFFDOneDC in priority
(5) The overflow VMs of r, if any, are assigned to other DCs in the order of non-increase order of electricity price in $\mathcal{F}_r$. Inside each DC, the assignment is completed by MFFDOneDC
(6) The VMs of r originally in k' are assigned to k' by MFFDOneDC.
(7) The overflow VMs of r', if any, are preferentially assigned to k', then to other DCs in the order of non-increase order of electricity price in $\mathcal{F}'_r$. Inside each DC, the assignment is completed by MFFDOneDC
(8) **else**
(9) Place the VMs in k' by MFFDOneDC
(10) **end if**
(11) **if** All VMs are assigned successfully **then**
(12) Clear the VMs of BC r in DC k
(13) **else**
(14) Remain the original assignment before mutation unchanged
(15) **end if**
(16) **end for**

Algorithm 4: Mutation operator.

resolution scheme and allow one BC to drive out VMs of other BCs as follows. The resident VMs of the other BCs and this driving BC in z will be cleared first. Thus ensure VMs in Z are kept unchanged and the group property of the parent is inherited. Then the cleared VMs are reassigned in z by MFFDOneDC according to the following BC order: first this driving BC and then another randomly selected one. The procedure is repeated until all BCs are processed or z is full. At last, the overflow VMs are reassigned to the feasible BCs of the BC being served, by MFFDOneB. Crossover can reduce both delay and PM and network cost.

4.4. Mutation Operator. The mutation happens in the third part of each segment, that is, DC in $\mathcal{F}_r$, thus leading to VMs replacement in PMs belonging to the mutated DC. There are three possible scenarios for the mutation. The first is that DC mutates to an idle candidate. The second is that it mutates to a DC which is used by the same BC. The last is that the target DC has been used by other BCs, and therefore this DC/PM is competed for by two BCs. The latter two scenarios may coexist. Because the newly added VMs may cause a violation of capacity or exceed the upper limit of the BC, some VMs may overflow; that is, the resident VMs need to be reassigned. This facilitates the changeover of DCs/PMs for two BCs so that resources can be balanced between them. See Algorithm 4 for details. In line (7), the overflow VMs of BC r' are preferentially assigned to k' because maybe k' is sill not full after VMs of BC r are assigned.

Power cost optimization is mainly fulfilled by mutation operator in that the electricity price differs at DC level.

4.5. The Unified Genetic Algorithm: STLGGA. The unified GA, segmented two-level grouping GA (STLGGA), is depicted in Algorithm 5.

5. Simulation Results

DC network is simulated in a 1400 $*$ 1400 grid in x-y plane. Generally, the DCs hold the property of clustering. 80% of DCs follow a normal distribution and 20% of DCs are selected uniformly from the grid. The distance between DCs is E-distance and the number of PMs inside each DC follows U (30–50). Configurations of PMs are borrowed from IBM System x M5 server and System x3300 M4 server [27]. Four classes of PMs equipped with a 1 Gbps Ethernet card are simulated. Considering the proportional configuration of PMs, we simply give each class a price instead of giving every resource a unique price. For the resource requirements of VMs, we adopt the four kinds of configurations of Amazon m3-serials [28] (for consistency with PM, GiB is replaced

(1) Pareto solution set $PSet \leftarrow \emptyset$
(2) Generate $3 * x$ initial population by SD (Algorithm 1), LPC and MFFD (Algorithm 2), respectively. Denotes these population as $\mathscr{C}$
(3) **while** Stopping condition not satisfied **do**
(4) **Crossover**. Random select two individuals from $\mathscr{C}$ and denote as X^1, X^2. Applying Algorithm 3 to X^1, X^2. Produce two offspring Y, Z
(5) **Mutation**. Applying Algorithm 4 to Y, Z to produce Y', Z'
(6) **Update of** $\mathscr{C}$. For each X in $\mathscr{C}$,
If $F(Y') \leq F(X)$, then $X = Y'$
If $F(Z') \leq F(X)$, then $X = Z'$
The size of $\mathscr{C}$ is kept not less than $3 * x$
(7) **Update of** $PSet$.
Remove from $PSet$ all the points X if $F(X)$ is dominated by $F(Y')$ and $F(Z')$
Add $F(Y')$ to $PSet$ if not exist point X in $PSet$ so that $F(X)$ dominates $F(Y')$
Add $F(Z')$ to $PSet$ if not exist point X in $PSet$ so that $F(X)$ dominates $F(Z')$
(8) **end while**

Algorithm 5: The unified genetic algorithm (STLGGA).

Table 1: PMs resource configurations and VMs resource requirements.

	CPU	Memory	Price		vCPU	Memory
PM	4 Cores	32 GB	1	VM	1 vCPUs	3.75 GB
	8 Cores	64 GB	2		2 vCPUs	7.5 GB
	24 Cores	192 GB	6		4 vCPUs	15 GB
	32 Cores	256 GB	8		8 vCPUs	30 GB

by GB). m3-serials are designed for general purpose and are very suitable for VDs. Table 1 lists the details. For each DC, we simulate a tree-like topology (Figure 1). Each core switch administrates 15 aggregate switches. Each aggregate switch administrates 2 access switches. 5 PMs are connected to the same access switch.

Depending on VMWare [29], a virtual machine cannot have more vCPUs than the number of logical cores of the host. The number of logical cores is equal to the number of the physical cores if hyperthreading is disabled or at most twice that number of the physical cores if hyperthreading is enabled. So we suppose there is a one-to-one relation between vCPU and physical core.

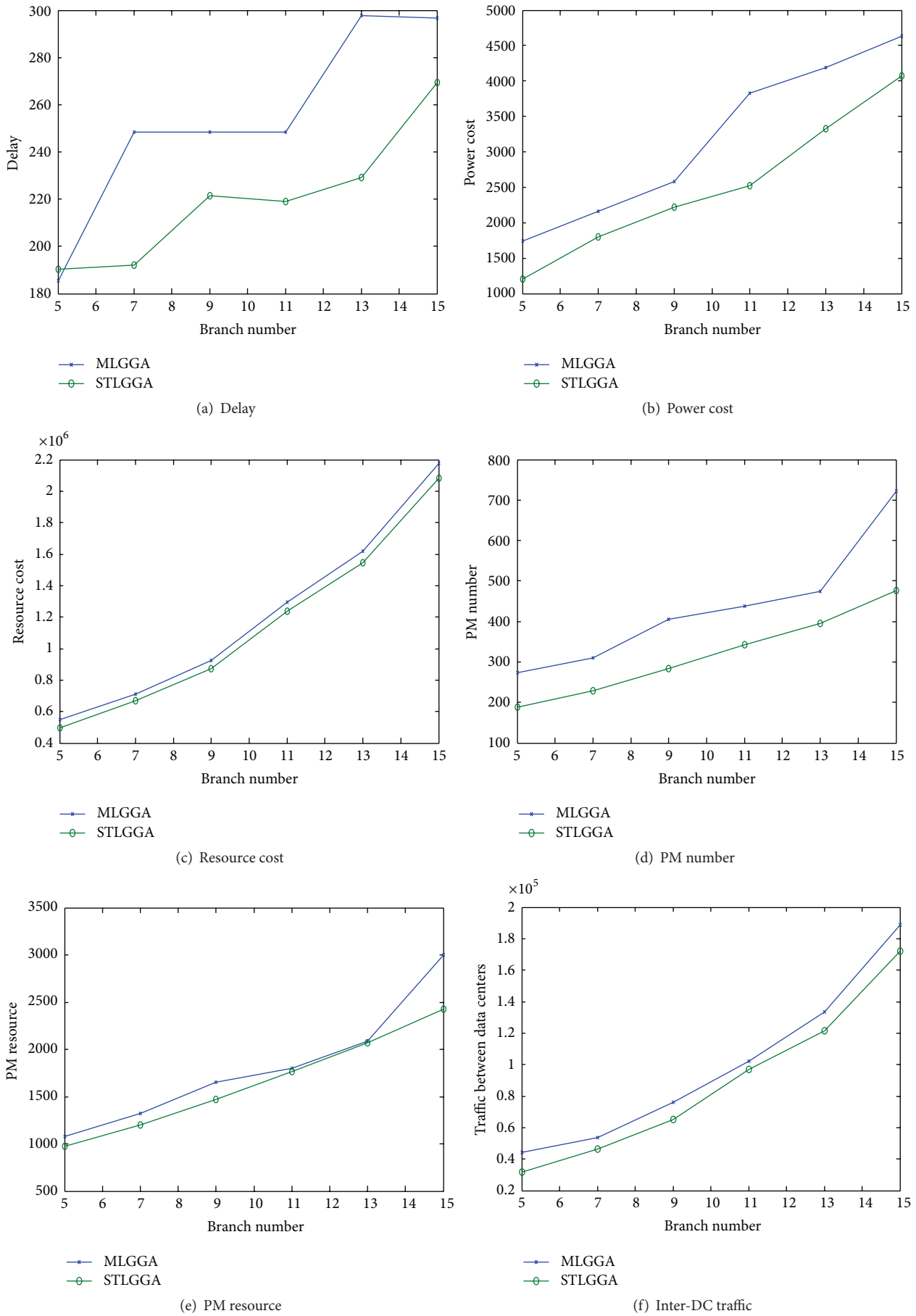
For multi-BC scenario, the number of VDs each BC requires is chosen uniformly between 20 and 300. Traffic between VDs follows U (0-1) Mbps [9]. For the VDs in the same BC, all of them communicate. For the VDs belonging to different BCs, only 10% VDs communicate. The bandwidth prices $b_1 = 0.01$, $b_2 = 0.03$, and $b_3 = 0.05$ are attached to access, aggregate, and core layer, respectively. The long distance inter-DC bandwidth price is $b = 0.1$.

The electricity price pool is from the data of August, 2015, of EIA [30]. Each simulated DC is equipped with a random price selected from the pool. The transmission delay is measured by distance; herein we use 300 as the threshold. We assume the queueing delay is same and therefore it is omitted. We adopt the idle or standby power consumption p_j^k as 60% of the peak power [21].

The initial solution size of STLGGA is $3 * x$ and $x = 20$. Our simulation is realized with Matlab. All numerical experiments stop after 30 thousand iterations. In average, it takes about 105 seconds and is a little slower than MLGGA which will take about 81 seconds as claimed in [13]. This is because STLGGA need to deal with the competition scenario. In all the simulations, the numerical results are the average of all the Pareto solutions for multiobjective programming.

We use a latest proposed unified algorithm, MLGGA [13], as the baseline. Because each time it can only place the VMs of one BC, the objective F_3 cannot be calculated. Therefore, F_1 and F_2 are used to calculate the fitness values. MLGGA is invoked for a randomly selected BC r in $\mathscr{F}_r$ to optimize F_1 and F_2. A solution is chosen randomly from the Pareto solutions as the assignment scheme of r. Then, other BCs are traversed on the basis of the remaining resource after the deployment of VDs of previous BCs, until the VMs of all BCs are assigned. Now F_3 can be calculated based on the results. The simulation results are detailed in Section 5.1. We also compare the two algorithms for one BC case in Section 5.2, where each algorithm pursues three objectives defined in MOBLP. Both scenarios validate the effectiveness of STLGGA.

5.1. Simulation Results for Multi-BC Scenario. To investigate the scale efficiency of STLGGA, we vary the number of BCs from 5 to 15. Figure 3 plots the solution quality compared with MLGGA. The three objectives achieved are shown



(a) Delay

(b) Power cost

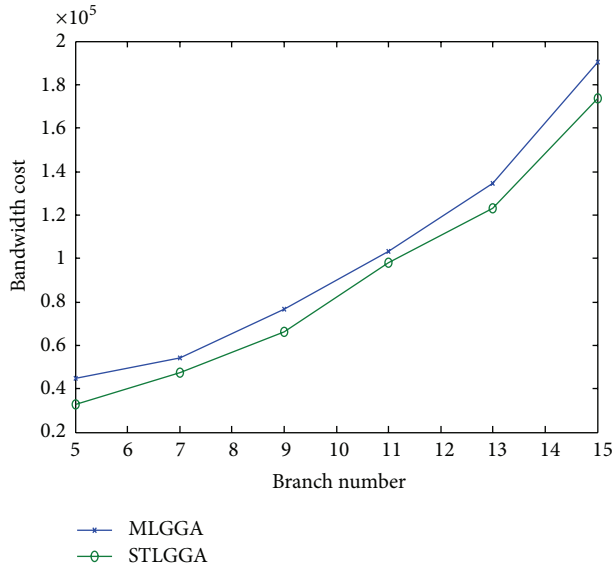
(c) Resource cost

(d) PM number

(e) PM resource

(f) Inter-DC traffic

Figure 3: Continued.



(g) Total bandwidth cost inside DC

FIGURE 3: The efficiency comparison with the number of BCs increasing.

in Figures 3(a)~3(c). With the number of BCs increasing, delay, accompanied with power and resources, increases for both algorithms. STLGGA results in an average of 13% shorter delay, which also means the communication of users supported by VDs deployed in different DCs becomes much faster. When STLGGA is used, power and resources are saved by 21% and 6% on average, respectively.

Resource efficiency is detailed in Figures 3(d)~3(g). STLGGA uses fewer PMs. The average reduction is about 27%, that is, 118 PMs (Figure 3(d)). Because of the heterogeneity of PMs, we also compare the PM resource consumed. STLGGA leads to 1% ~19% resource cost saving. On average, about 9% of cost is saved (Figure 3(e)). Fewer PMs indicate that less power is needed to keep PMs active. Therefore, power efficiency is improved and the total power is reduced. This further backs up Figure 3(b). On average, STLGGA also leads to a reduction of 13% of the expensive inter-DC traffic (Figure 3(f)). Because the total traffic between all VMs is a determined value, STLGGA saves more expensive long distance inter-DC bandwidth by converting more inter-DC traffic to intra-DC traffic at the cost of relatively cheaper intra-DC bandwidth, including access, aggregate, and core layer one. The traffic across the three layers produced by STLGGA is more than what was produced by MLGGA. But the cost of the total required bandwidth produced by STLGGA inside DC is much less than what was produced by MLGGA, that is, an average reduction of about 13%. It is consistent with our purpose (objective f) to optimize the bandwidth cost inside DC (Figure 3(g)).

Suppose 5 BCs apply for VDs. We study VDs placement with different scales of DCs varied from 4 to 60. The capacity of DC and the number of VDs requested for each BC remain as before. Figure 4 demonstrates the three objectives. On average, 5% delay is shortened. 10% power and 5% resource cost are saved, respectively. The detailed resource comparison results show the same tendency as Figures 3(d)~3(f) and are omitted here.

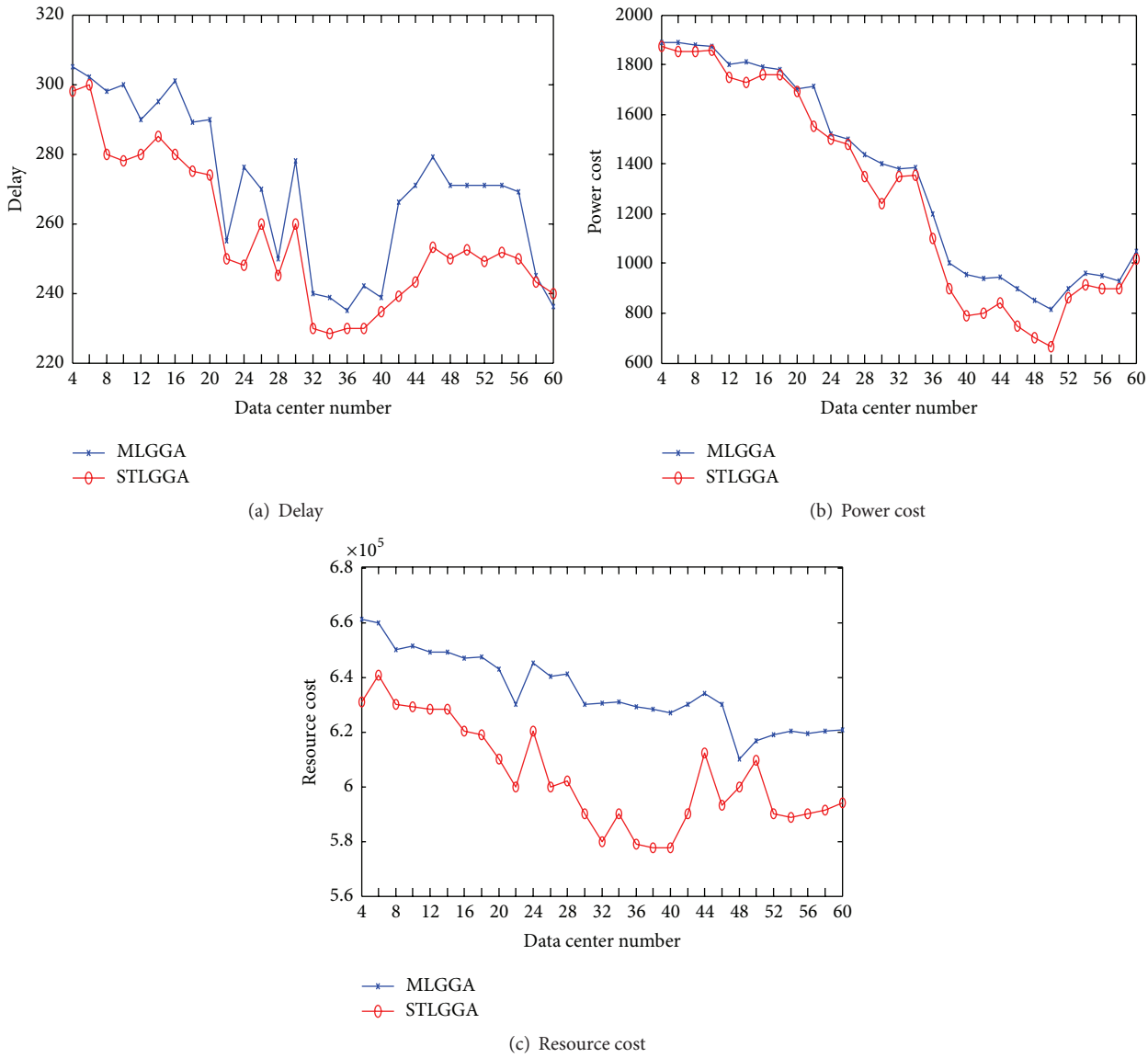
It is noted that, naturally, it is expected that the solution quality will improve because as the number of DCs increases, there are more candidates. But in reality, the Pareto solution tries to balance the three objectives and the figures appear in a nonsmooth phenomenon. The delay displays an uptrend when DCs increase from 35 to 50 (Figure 4(a)), accompanied by the declining of power cost (Figure 4(b)) and resource cost (Figure 4(c)). But when delay decreases as DCs increased from 55 to 60, the latter two objectives go upward. This is due to the random number of PMs in DC, the location diversity, and random power assignment. It is also observed that STLGGA still performs much better than MLGGA in the latter two objectives. This comes at a cost of a little bigger delay when 60 DCs are searched. Generally, with the number of DCs increasing, all the three figures show a downtrend.

5.2. Simulation Results for Single BC Scenario. We also examine the VDs placement when there is just one BC. The number of VDs being applied for varies from 500 to 1000. This time, MLGGA is invoked to optimize F_1, F_2, and F_3 simultaneously within the feasible DCs, now that F_3 can be calculated. The average results of the Pareto solutions for both algorithms are reported in Figure 5. Similar to the results of multi-BC, STLGGA outperforms MLGGA for the three objectives, as well as for PM number, PM resources, inter-DC traffic, and total required bandwidth cost.

This further validates the idea that STLGGA not only works well for multi-BC, but also does for a single BC.

6. Conclusion

Considering the bilevel resource provision for the deployment of virtual desktops of multi-BC in distributed cloud,



(a) Delay

(b) Power cost

(c) Resource cost

Figure 4: The efficiency comparison with the number of DCs increasing.

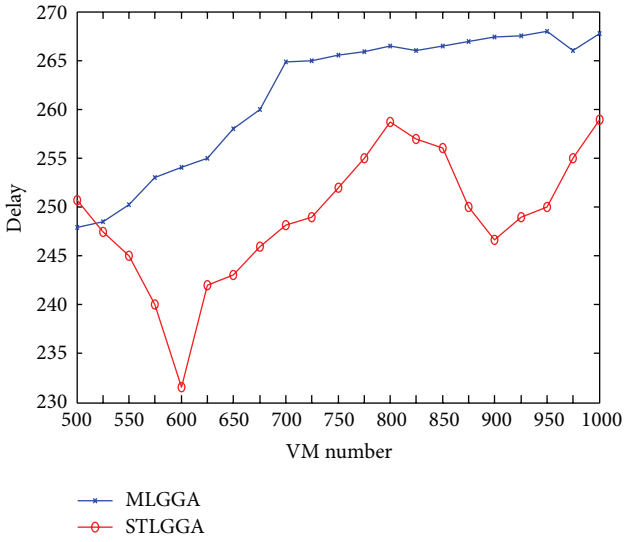
service delay, power efficiency, and cost optimization are explored in this paper. The problem is formulated as multiobjective bilevel programming which captures the noncooperative relation of DC network level and server level. So it can facilitate the optimization of nodes and bandwidth cost of both levels without violating the delay threshold, while striving to further minimize the maximum delay of each BC. Because of the NP-hard nature of the problem, a segmented two-level group GA is proposed. Novel coding, initial population production, and operators schemes are tailored to address the problem. The effectiveness of the algorithm is validated by extensive simulations. The algorithm outperforms the baseline algorithm in both multi-BC and single BC scenarios.

Though we focus on VDs deployment, it is just one applicable object of the proposed formulation and algorithm. They can also be applied to the placement of VMs to support any location-sensitive or delay-sensitive services [31] in distributed clouds, such as VOD [32] and big data [33].

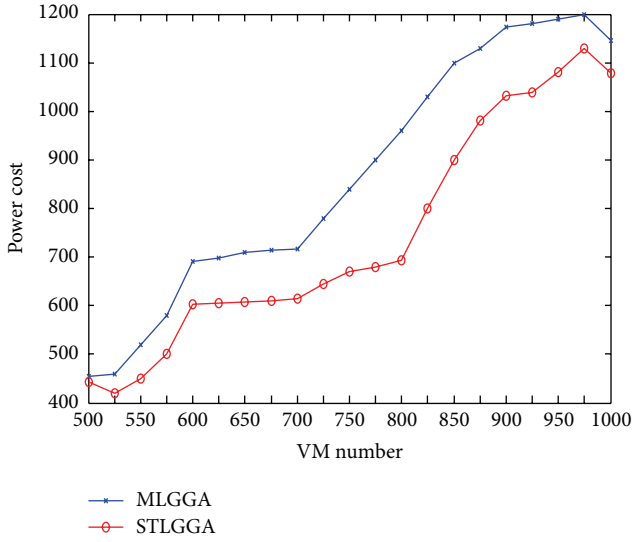
In this paper, we only consider different electricity prices of DCs in energy cost optimization. But it cannot reflect the utilization of renewable energy. Because renewable energy, such as solar, wind, and tidal energy, is varying with time and regions, VDs can be migrated to exploit them more efficiently within the delay threshold [34, 35]. In our future work, we aim to utilize more renewable energy by leveraging the wide-spanned distributed DCs over the globe, so that not only economic, but also social benefit can be achieved.

Conflict of Interests

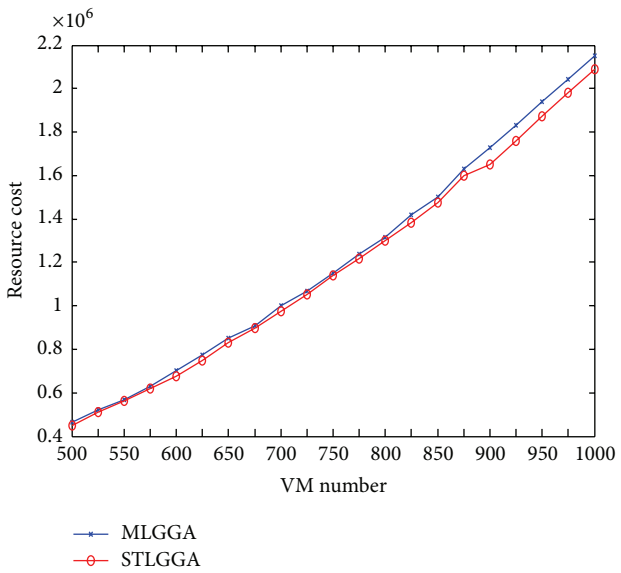
The authors declare that there is no conflict of interests regarding the publication of this paper.



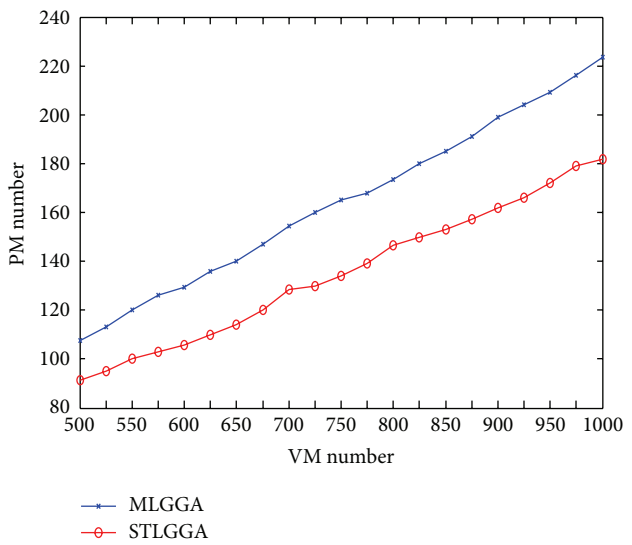
(a) Delay



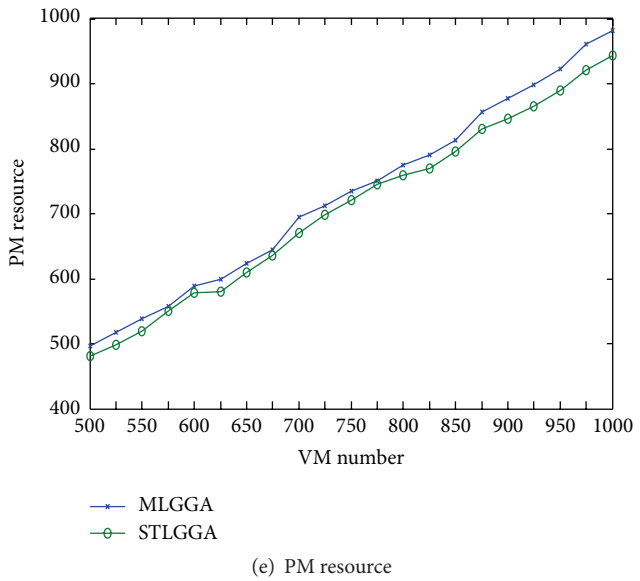
(b) Power cost



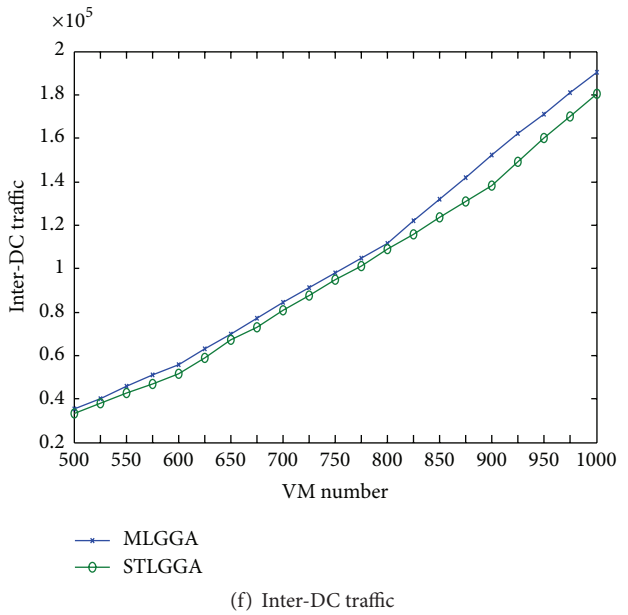
(c) Resource cost



(d) PM number

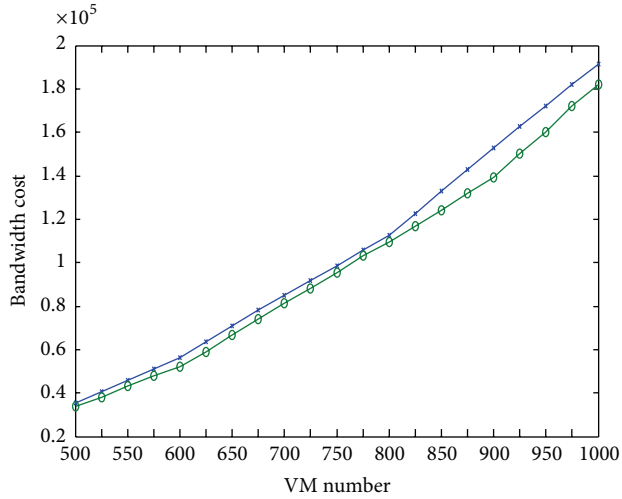


(e) PM resource



(f) Inter-DC traffic

Figure 5: Continued.



(g) Total bandwidth cost inside DC

Figure 5: The efficiency comparison for single BC.

Acknowledgments

This work is supported in part by National High Technology Research and Development Program of China (no. 2015AA016008), National Science and Technology Major Project (no. JC201104210032A), National Natural Science Foundation of China (no. 11371004, 61402136), Natural Science Foundation of Guangdong Province, China (no. 2014A030313697), International Exchange and Cooperation Foundation of Shenzhen City, China (no. GJHZ20140422173959303), Shenzhen Strategic Emerging Industries Program (no. ZDSY20120613125016389), Shenzhen Overseas High Level Talent Innovation and Entrepreneurship Special Funds (no. KQCX20150326141251370), Shenzhen Applied Technology Engineering Laboratory for Internet Multimedia Application of Shenzhen Development and Reform Commission (no. [2012]720), and Public Service Platform of Mobile Internet Application Security Industry of Shenzhen Development and Reform Commission (no. [2012]900).

References

[1] T. Guo, V. Gopalakrishnan, K. K. Ramakrishnan, P. Shenoy, A. Venkataramani, and S. Lee, "VMShadow: optimizing the performance of virtual desktops in distributed clouds," in *Proceedings of the Proceedings of the 4th Annual Symposium on Cloud Computing (SOCC '13)*, p. 42, ACM, Santa Clara, Calif, USA, October 2013.

[2] L. Deboosere, B. Vankeirsbilck, P. Simoens, F. De Turck, B. Dhoedt, and P. Demeester, "Efficient resource management for virtual desktop cloud computing," *The Journal of Supercomputing*, vol. 62, no. 2, pp. 741–767, 2012.

[3] P. Barham, B. Govic, K. Fraser et al., "Virtual machine monitors: xen and the art of virtualization," in *Proceedings of the ACM Symposium on Operating System Principles (SOSP '03)*, vol. 36, pp. 164–177, Bolton Landing, NY, USA, August 2003.

[4] C. R. Gil, P. V. Prieto, M. Silva, and A. Teixeira, "Virtual desktop infrastructure (VDI) technology: FI4VDI project," in *New Perspectives in Information Systems and Technologies, Volume 2*, vol. 276 of *Advances in Intelligent Systems and Computing*, pp. 35–42, Springer, Berlin, Germany, 2014.

[5] Microsoft, "Desktops," http://msdn.microsoft.com/en-us/library/windows/desktop/ms682573.aspx.

[6] VMware, "Desktop solution," http://www.vmware.com/products/horizon-air-desktops/.

[7] Huawei, "Desktop solution," http://e-file.huawei.com/en/solutions/technical/cloud-computing/desktop-cloud.

[8] N. Chen, B. Xu, D. Hu, and Y. Wan, "On user capacity optimization strategy of server computing based desktop cloud," in *Proceedings of the 1st International Workshop on Cloud Computing and Information Security*, Atlantis Press, Shanghai, China, November 2013.

[9] M. Alicherry and T. V. Lakshman, "Network aware resource allocation in distributed clouds," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '12)*, pp. 963–971, IEEE, Orlando, Fla, USA, March 2012.

[10] P. Calyam, R. Patali, A. Berryman, A. M. Lai, and R. Ramnath, "Utility-directed resource allocation in virtual desktop clouds," *Computer Networks*, vol. 55, no. 18, pp. 4112–4130, 2011.

[11] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proceedings of the IEEE INFOCOM*, pp. 1–9, IEEE, San Diego, Calif, USA, March 2010.

[12] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 68–73, 2008.

[13] F. F. Moghaddam, R. F. Moghaddam, and M. Cheriet, "Carbon-aware distributed cloud: multi-level grouping genetic algorithm," *Cluster Computing*, vol. 18, no. 1, pp. 477–491, 2015.

[14] C. L. T. Man and M. Kayashima, "Virtual machine placement algorithm for virtualized desktop infrastructure," in *Proceedings of the IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS '11)*, pp. 333–337, Beijing, China, September 2011.

[15] M. Makarov, P. Calyam, A. Sukhov, and V. Samykin, "Time-based criteria for performance comparison of resource-intensive user tasks in virtual desktops," in *Proceedings of the International Conference on Computing, Networking and Communications (ICNC '14)*, pp. 112–116, IEEE, Honolulu, Hawaii, USA, February 2014.

[16] J. Li, Y. Jia, L. Liu, and T. Wo, "CyberLiveApp: a secure sharing and migration approach for live virtual desktop applications in a cloud environment," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 330–340, 2013.

[17] A. Kochut, "Power and performance modeling of virtualized desktop systems," in *Proceedings of the IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '09)*, pp. 1–10, IEEE, London, UK, September 2009.

[18] P. Calyam, S. Rajagopalan, A. Selvadhurai et al., "Leveraging OpenFlow for resource placement of virtual desktop cloud applications," in *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM '13)*, pp. 311–319, IEEE, Ghent, Belgium, May 2013.

[19] Y. Yao, L. Huang, A. Sharma, L. Golubchik, and M. Neely, "Power cost reduction in distributed data centers: a two-time-scale approach for delay tolerant workloads," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 200–211, 2014.

[20] T. Ng and H. Zhang, "Predicting internet network distance with coordinates-based approaches," in *Proceedings of the IEEE 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '02)*, vol. 1, pp. 170–179, 2002.

[21] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2, pp. 13–23, 2007.

[22] G. B. Richardson, "The theory of the market economy," *Revue Économique*, vol. 46, no. 6, pp. 1487–1496, 1995.

[23] V. Oduguwa and R. Roy, "Bi-level optimisation using genetic algorithm," in *Proceedings of the IEEE International Conference on Artificial Intelligence Systems (ICAIS '02)*, pp. 322–327, Divnomorskoe, Russia, 2002.

[24] H. Sun, Z. Gao, and J. Wu, "A bi-level programming model and solution algorithm for the location of logistics distribution centers," *Applied Mathematical Modelling*, vol. 32, no. 4, pp. 610–616, 2008.

[25] L. Davis, *Handbook of Genetic Algorithms*, vol. 115, Van Nostrand Reinhold, New York, NY, USA, 1991.

[26] J. Zhang, Z. He, H. Huang, X. Wang, C. Gu, and L. Zhang, "SLA aware cost efficient virtual machines placement in cloud computing," in *Proceedings of the IEEE International Performance Computing and Communications Conference (IPCCC '14)*, pp. 1–8, Austin, Tex, USA, December 2014.

[27] IBM, "Ibmpminstance," http://www-03.ibm.com/systems/x/hardware/tower/index.html.

[28] Amazon, "Amazonvminstance," http://aws.amazon.com/ec2/instance-types/.

[29] VMware, "vCPU," http://pubs.vmware.com/vsphere-50/index.jsp#com.vmware.vsphere.vm_admin.doc_50/GUID-13AD347E-3B77-4A67-B3F4-4AC2230E4509.html.

[30] EIA, "Usapowerprice," http://www.eia.gov/state/data.cfm?sid=CT.

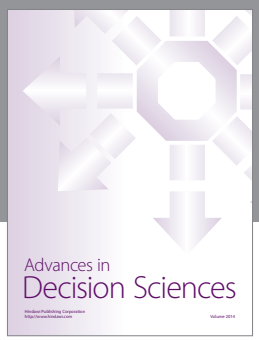
[31] I. L. Bedhiaf, R. B. Ali, and O. Cherkaoui, "On the problem of mapping virtual machines to physical machines for delay sensitive services," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM '12)*, pp. 2628–2633, IEEE, Anaheim, Calif, USA, December 2012.

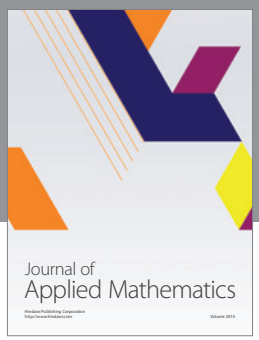
[32] V. Aggarwal, X. Chen, V. Gopalakrishnan, R. Jana, K. Ramakrishnan, and V. A. Vaishampayan, "Exploiting virtualization for delivering cloud-based IPTV services," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS '11)*, pp. 637–641, Shanghai, China, April 2011.

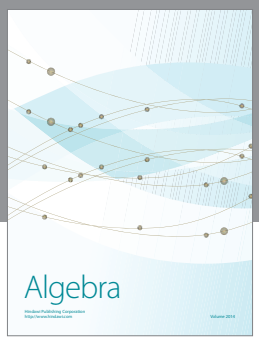
[33] L. Zhang, C. Wu, Z. Li, C. Guo, M. Chen, and F. C. M. Lau, "Moving big data to the cloud: an online cost-minimizing approach," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 12, pp. 2710–2721, 2013.

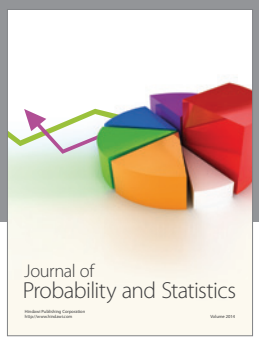
[34] C. Clark, K. Fraser, S. Hand et al., "Live migration of virtual machines," in *Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pp. 273–286, USENIX Association, 2005.

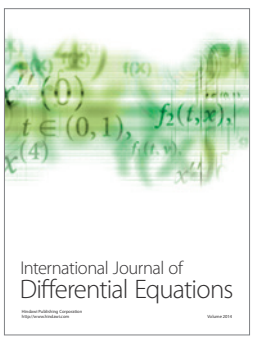
[35] W. Deng, F. Liu, H. Jin, B. Li, and D. Li, "Harnessing renewable energy in cloud datacenters: opportunities and challenges," *IEEE Network*, vol. 28, no. 1, pp. 48–55, 2014.

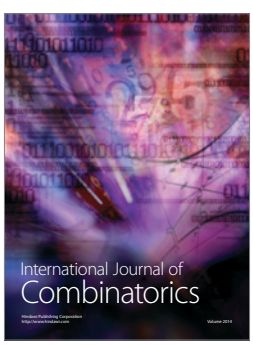


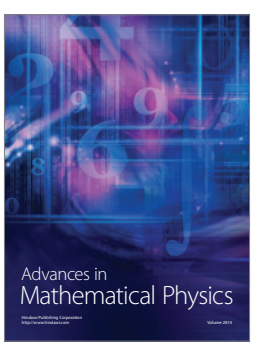


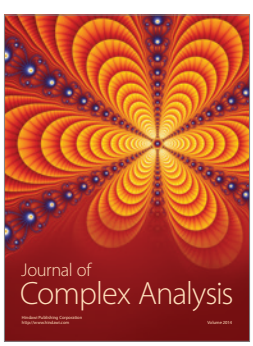


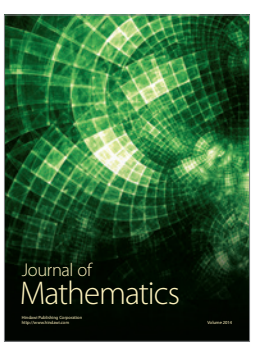


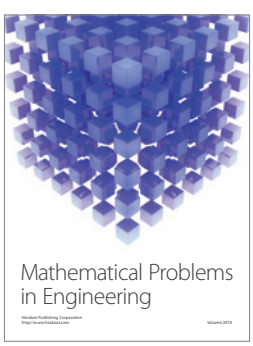


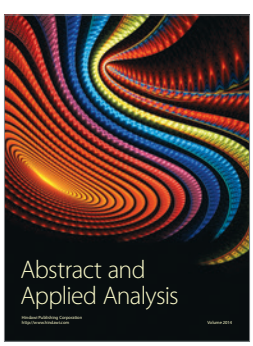


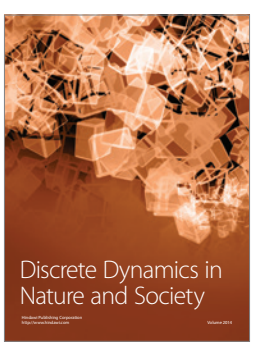


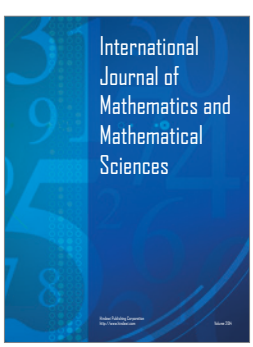


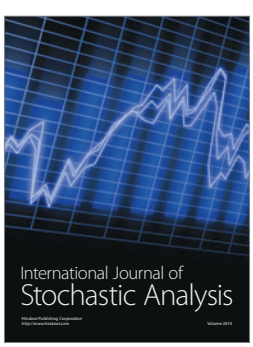












Hindawi

Submit your manuscripts at
http://www.hindawi.com