Computer Networks 83 (2015) 59–75

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Operational cost minimization of distributed data centers through the provision of fair request rate allocations while meeting different user SLAs

Zichuan Xu, Weifa Liang [*]

The Australian National University, Canberra, ACT 0200, Australia

A R T I C L E I N F O

Article history:
Received 23 April 2014
Received in revised form 4 January 2015
Accepted 16 February 2015
Available online 6 March 2015

Keywords:
Operational cost minimization
Fair request rate allocation
Service Level Agreements
Distributed data centers
Electricity price diversity
Approximation algorithm

A B S T R A C T

Data centers as computing infrastructures for cloud services have been growing in both number and scale. However, they usually consume enormous amounts of electricity that incur high operational costs of cloud service providers. Minimizing these operational costs thus becomes one main challenge in cloud computing. In this paper, we study the operational cost minimization problem in a distributed cloud computing environment that not only considers fair request rate allocations among web portals but also meets various Service Level Agreements (SLAs) between users and the cloud service provider, with an objective to maximize the number of user requests admitted while keeping the operational cost minimized, by exploiting the electricity diversity. To this end, we first propose an adaptive operational cost optimization framework that incorporates time-varying electricity prices and dynamic user request rates. We then devise a fast approximation algorithm with a provable approximation ratio for the problem, by utilizing network flow techniques. Finally, we evaluate the performance of the proposed algorithm through experimental simulations, using real-life electricity price data sets. Experimental results demonstrate that the proposed algorithm is very promising, and the solution obtained is nearly optimal.

1. Introduction

Many cloud service providers, such as Amazon EC2, Google, and Microsoft Azure [4,19,33], provide global services through their distributed data centers. As agreed in their Service Level Agreements (SLAs), cloud service providers must guarantee that their services are within the specified tolerant response times by users. To this end, the cloud service providers usually over-provision their services by switching on more servers than needed. This however results in only 10–30% server utilization [6,7], leading to the waste of huge amounts of electricity. It is estimated that the total electricity bill for data centers globally in 2010 was over \$11 billion, and this figure is almost doubled every five years [23]. However, it is noticed that electricity prices in different time periods and regions are different [13,42], this creates great opportunities for cloud service providers to minimize the electricity bills of their data centers through allocating user requests to geographically cheaper-electricity data centers while still meeting various user SLA requirements.

To minimize the operational cost, there are two key issues to be addressed. One is that different user requests have different SLAs, how to perform an allocation that meets multiple SLA requirements is challenging. A solution is to allocate the requests with the same level SLA to one

* Corresponding author. Tel.: +61 261253019.
E-mail addresses: edward.xu@anu.edu.au (Z. Xu), wliang@cs.anu.edu.au (W. Liang).

http://dx.doi.org/10.1016/j.comnet.2015.02.028
1389-1286/© 2015 Elsevier B.V. All rights reserved.

data center. Thus, each data center can guarantee its service to the admitted requests with a given level SLA [14,15,22,37–39,43,44]. This solution however may not be applicable to large-scale data centers with multiple SLA requirements. The other is that user requests from different web portals at different regions must be fairly allocated to different data centers. Otherwise, a biased allocation may severely degrade the reputation of the cloud service provider in those under-allocated regions, thereby reducing the potential revenue of the cloud provider in these regions in future.

By exploring electricity diversity, the operational cost minimization problem in distributed cloud environments has been studied in the past [1,2,10,14,15,22,25,35–39,44,45]. Most of existing works dealt with a single level SLA and focused on developing exact and heuristic solutions. For example, the authors in [1,2,37–39,45] developed exact solutions, by formulating the problem as a Mixed Integer Programming (MIP). However, such exact algorithms are only limited in small size problem instances, as their running time is prohibitively expensive, with the growth of problem size. Furthermore, even if such an exact solution is found ultimately, it may not be applicable to due to time-varying electricity prices and user request rates. On the other hand, the authors in [25,26] focused on developing heuristics such as simulated annealing for the problem. These heuristic solutions do not guarantee how far they are from the optimal one. Unlike these mentioned works, in this paper we deal with the operational cost minimization problem that not only satisfies multi-level SLAs but also guarantees a "proportional fairness" criteria ensuring the same proportion of requests from different web portals to be allocated to different data centers. Specifically, the novelty of this paper is as follows. We consider multiple-level user SLA requirements and fair request rate allocation. To the best of our knowledge, no one has studied multi-level SLAs yet. We devised a very first fast, scalable approximate solution with a guaranteed approximation ratio, by adopting techniques that are essentially different from existing studies. That is, we reduce the problem with multi-level SLA constraints to another problem – the minimum cost multicommodity flow problem. The novelty of reducing the operational cost minimization problem in a distributed cloud environment to the minimum cost multicommodity flow problem in a flow network lies in the construction of the flow network. That is, how to transform the multi-SLA requirements of user requests into the edge capacity constraints of the flow network, and how to map the electricity and bandwidth costs of requests to the edge costs in the flow network.

The main contributions of this paper are summarized as follows. We first consider the operational cost minimization problem in a distributed cloud computing environment by incorporating time-varying electricity prices and user request rates. We then devise a fast approximation algorithm with a provable approximation ratio for the problem. We finally evaluate the performance of the proposed algorithm by simulations, using real-life electricity price data. Experimental results demonstrate that the proposed algorithm is very promising, and the solution obtained is fractional of the optimal.

The remainder of this paper is organized as follows. In Section 2, we first summarize related works, followed by introducing the system model, the problem definition, and an efficient approximation algorithm for the minimum cost multicommodity flow problem in Section 3. We then propose an optimization framework and devise a fast approximation algorithm in Section 4. We finally evaluate the performance of the proposed algorithm through experimental simulations in Section 5, and we conclude in Section 6.

2. Related work

Most existing studies aim to minimize the energy consumption of a single data center [12,14,15,27,30,32]. For example, Elnozahy et al. [12] investigated the problem of power-efficient resource management in a single-service environment for web-applications with a single level SLA on a homogeneous cluster to reduce its operational cost. They showed that the proposed algorithm can provide up to 29% energy savings. Lin et al. [30] observed that much power is wasted in maintaining excess service capacity during periods of predictably low load. They devised an online algorithm for minimizing the power consumption in a data center by dynamically sizing the number of switching on servers. Kliazovich et al. [24] developed energy-efficient schedulers for a data center by considering the tradeoff between job scheduling (to minimize the amount of computing servers) and traffic patterns (to avoid hotspots in the data center network). Tziritas et al. [41] devised an algorithm for distributed virtual machine placement on server clusters in a data center with the aim of minimizing the application resource utilization, without violating server capacity constraints.

Cloud service providers usually deploy their data centers in different geographical regions to meet ever-growing user demands and economically run their business. One promising approach of minimizing the operational cost of these data centers is to explore time-varying electricity prices in different regions, this opens up a new dimension in designing efficient algorithms for geographically distributed data centers [9,10,18,22,25,26,28,29,36–38,40,44,45]. For example, Qureshi et al. [36] characterized the energy expense per unit of computation due to fluctuating electricity prices by showing that the exploration of the difference of electricity prices may save millions of dollars per day through simulations. Liu et al. [28] improved the energy sustainability of distributed data centers by exploiting sources of renewable energy. Through incorporating the geographical load-balancing algorithm [29], they investigated the impact of geographical load-balancing on the operational cost and demonstrated the necessity of the storage of renewable energies. Ren et al. [40] addressed batch job scheduling in distributed data centers and proposed an online scheduling algorithm to minimize the energy cost while maintaining the rate fairness among different sources. Adnan *et. al* [1] formulated the problem of scheduling various workloads (interactive requests and batch jobs) to geographically distributed data centers as a mixed integer programming and provided an exact solution. Le et al. [25,26] proposed heuristic algorithms for load balancing distributions among data centers by jointly considering variable electricity prices and green energy sources. A general framework is proposed to manage the usage of "brown energy" produced via carbon-intensive means and green energy with the aim of reducing environmental effects [26]. Similarly, Chen et al. [10] proposed a job scheduling algorithm to minimize the brown energy consumption across geographically distributed data centers that are partially powered by green energy sources. Buchbinder et al. [9] devised an online algorithm for job migrations among multiple data centers with the aim of reducing the electricity bill. Their simulation results indicated that their solution is within 6% of the optimal offline solution. Guo et al. [22] initialized the study of reducing electricity bills via the use of energy storage, and solved the problem by the Lyapunov optimization technique.

So far we have focused on minimizing the operational cost of distributed data centers for user request processing. It must be mentioned that when processing user requests, the Quality of Service requirements (QoS) of the users must be taken into account, too. Queueing theory has widely been applied to model the average waiting time of request queues as the request SLAs [3]. Many existing studies in literature thus assumed that within each data center, there is only one centralized queue [15,22,14,44,38,37], which means that the data center can only process the requests with identical average delays. For example, Gandhi et al. [15] adopted a centralized queue for their server farm with the aim to minimize the mean response time, under capped power consumption. Zhang et al. [44,45] investigated the problem of geographical request allocation with the aim of maximizing the use of renewable energy under a given operation budget, they adopted $M/M/n$ and $G/G/m$ queueing theories to model the response time of a data center. Liu et al. [27] introduced a general queueing model to meet the specified SLA requests. Rao et al. [37,38] explored the electricity cost reduction for multiple data centers under multi-electricity-market environments, assuming that each data center has a centralized $M/M/n$ queue. They formulated the problem as a mixed integer programming and devised a flow-based heuristic [39]. There are also several studies [1,2] that considered different SLA requirements by different batch jobs. The authors formulated the problem of geographical job load balancing as a mixed integer program and provided heuristic solutions through dynamically executing, deferring, and job migrating.

3. Preliminaries

In this section we first introduce the system model and notations. We then define the operational cost minimization problem. We finally introduce a fast approximation algorithm for the minimum cost multicommodity flow problem which will be used later.

3.1. System model

We consider a distributed cloud computing environment that consists of a set of geographically distributed data centers $\mathcal{DC} = \{DC_i \mid 1 \leqslant i \leqslant N\}$ and a set of web portals $\mathcal{WP} = \{WP_j \mid 1 \leqslant j \leqslant M\}$, which is managed by a cloud service provider. Each data center DC_i is equipped with hundreds of thousands of homogeneous servers. Denote by $N_i (= |DC_i|)$ and μ_i the number of servers and the service rate of each server in DC_i for every i with $1 \leqslant i \leqslant N$. Each web portal WP_j serves as a front-end server that directly receives requests from the users, performs request rate allocations to data centers, assuming that there is a dedicated backbone link serving the communications between a data center and a web portal [46]. An example of the distributed cloud is shown in Fig. 1.

Each user sends its requests to its nearby web portal, and the requests at each web portal are then allocated to different data centers. Associated with each request, there is an *average* tolerant delay requirement which is the negotiated SLA between the user and the cloud service provider. Such a delay usually consists of the network latency incurred on the links between web-portals and data centers and the average delay waiting for being processed. Since the communications between web portals and data centers are carried out by the backbone links of the distributed cloud provider, the network latency between a web portal and a data center within a short time period is given and does not change over this time period. The network latency will be integrated into the proposed cost optimization framework later. In the rest of this paper, we focus only on differentiating average waiting delays at each data center.

Different users may have different SLAs, we here assume that there are k different levels of average delay requirements among user requests. We further assume that $k \ll \mu_i \cdot N_i$ for any i with $1 \leqslant i \leqslant N$. Denote by $r_j \in \mathbb{Z}$ the accumulative request rate at web portal WP_j that consists of k different levels of requests, i.e., $r_j = \sum_{l=1}^{k} r_j^{(l)}$, where $r_j^{(l)}$ $(\geqslant 0)$ is the number of requests with average delay requirement D_l and $D_1 < D_2 < \cdots < D_k$. We assume that time is slotted into *equal time slots*, and the accumulative request rate r_j at WP_j is measured at each time slot. We further assume that each time slot is sufficiently long so that servers are not switched on and off frequently, given the significant wear-and-tear costs of power-cycling. To differentiate requests with different average delay requirements, each data center stores its allocated requests temporarily into k $M/M/n$ queues [21] for these requests waiting for processing, where the lth queue accommodates the requests with average delay requirement $D_l, 1 \leqslant l \leqslant k$.

Given $n^{(l)}$ servers in a data center with service rate μ, the request arrival rate $r^{(l)}$, and the probability $Pr^{(l)}$ of requests waiting in the lth queue, each request waits for $\frac{Pr^{(l)}}{n^{(l)}\mu - r^{(l)}}$ time on average to be served [21]. Then, each data center needs to determine the number of servers for processing the requests at each waiting queue. Denote by $r_{j,i}^{(l)}(t)$ the allocated request rate of average delay D_l from web portal WP_j to data center DC_i with $n_i^{(l)}$ servers assigned, denote by $R_i^{(l)}(t)$ the arrival rate of requests with delay D_l from all web portals to data center DC_i at time slot

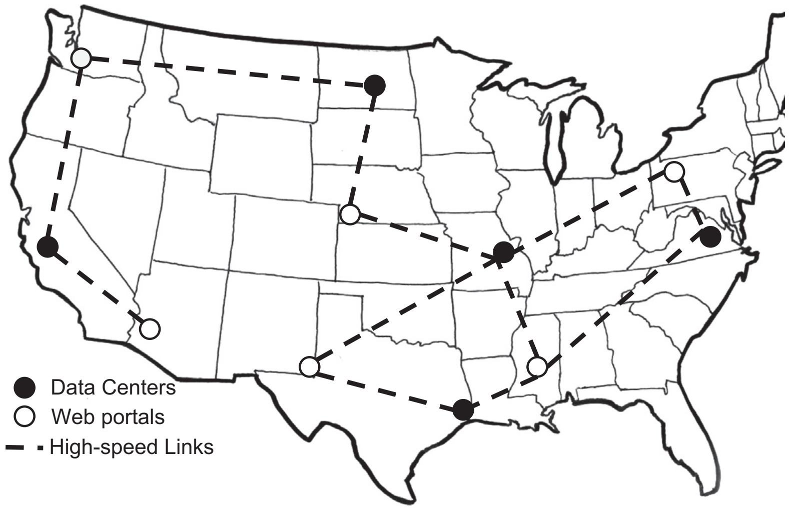


Fig. 1. An example of the distributed cloud.

t, and let $R_i^{(l)}(t) = \sum_{j=1}^{M} r_{j,i}^{(l)}(t)$. Then, the average waiting delay of requests with average delay D_l at DC_i is

$$D_i^{(l)}\left(n_i^{(l)}(t), R_i^{(l)}(t)\right) = \frac{1}{n_i^{(l)}(t)\cdot\mu_i - R_i^{(l)}(t)}, \quad (1)$$

where $\sum_{l=1}^{k} n_i^{(l)}(t) \leqslant N_i$. If there is only one request in each queue, the request will be processed immediately, which means that $D_i^{(l)}(N_i,1) \ll D_l$. By Eq. (1), we have $D_l \gg \frac{1}{N_i\mu_i - 1}$, which means that $\frac{1}{D_l} \ll N_i\mu_i$. In the rest of this paper we thus assume that $\frac{1}{D_l} \ll N_i\mu_i$ and $\frac{k}{D_l} \ll N_i\mu_i$ since $k \ll \mu_i \cdot N_i$.

The average delay requirement D_l of each admitted request in DC_i will be met if its average waiting time at DC_i is no greater than D_l, i.e.,

$$D_i^{(l)}\left(n_i^{(l)}(t), R_i^{(l)}(t)\right) \leqslant D_l, \quad \forall i, \quad 1 \leqslant i \leqslant N \text{ and } \forall l, \quad 1 \leqslant l \leqslant k. \quad (2)$$

where $R_i^{(l)}(t) = \sum_{j=1}^{M} r_{j,i}^{(l)}(t)$ for all i with $1 \leqslant i \leqslant N$.

3.2. Electricity cost model

The electricity cost of a data center DC_i during each time slot t is determined by the amount of power it consumed and the local electricity price during that time period. Let $p_i(t)$ and $E_i(t)$ be the unit-energy electricity price at the location of DC_i and the total energy consumption of DC_i in the time duration of time slot t, assuming that the load among the servers at DC_i is well balanced. $E_i(t)$ thus is proportional to the amount of energy consumption per request in data center DC_i. Let $\alpha_i^{(l)}$ be the amount of average energy consumed per request with the average delay requirement D_l, which is a constant. The energy consumption $E_i(t)$ and the electricity cost $p_i(t)\cdot E_i(t)$ in DC_i at time slot t are

$$E_i(t) = \sum_{j=1}^{M}\sum_{l=1}^{k} \alpha_i^{(l)} r_{j,i}^{(l)}(t), \quad (3)$$

and

$$p_i(t)\cdot E_i(t) = p_i(t)\sum_{j=1}^{M}\sum_{l=1}^{k} \alpha_i^{(l)} r_{j,i}^{(l)}(t). \quad (4)$$

The above electricity cost modeling is also adopted in [16,37–39]. Unless otherwise specified, in the rest of this paper we will drop the time slot parameter t from all formulas if no confusion arises.

3.3. Network bandwidth cost

Most existing studies assumed that allocating different user requests to different data centers is transparent (to users), and does not incur any cost on the bandwidth consumption as stateless requests from the web portals normally consist of simple jobs and can be enclosed by small data packages. However, a cloud service provider usually leases the bandwidth from ISPs for its data transfer between its web portals and its data centers, which does incur the service cost [34,46]. Similar to the pay-as-you-go policy for their computing resources, cloud service providers now are also embracing usage-based pricing policy for their bandwidth resources. For example, Rackspace [11] charges each instance $0.06 per hour for routing its traffic. We thus assume that the cost on bandwidth consumption between a web portal and a data center is proportional to the number of requests transferred between them. Also, it is very likely that the cloud service provider may lease bandwidths with different prices between its web portals and its data centers, according to the request load of the web portals and the price on the bandwidth consumption of a request. Let p_j^b denote the price of the bandwidth cost of allocating a request from WP_j to the data centers. Then, the cost of allocating the requests from web portal WP_j to all data centers is $p_j^b \sum_{i=1}^{N} r_{j,i}$.

3.4. Problem definition

Suppose that there is a cloud service provider managing a set of data centers $\mathcal{DC} = \{DC_i \mid 1 \leqslant i \leqslant N\}$ and a set of web portals $\mathcal{WP} = \{WP_j | 1 \leqslant j \leqslant M\}$ that are located in different geographic regions. Each DC_i contains N_i homogeneous servers with the average service rate μ_i of each server for all i with $1 \leqslant i \leqslant N$. User requests are first submitted to their nearby web portals, and then allocated to different data centers for processing.

Given a time slot t, let $r_j^{(l)}$ be the request rate with delay D_l from web portal WP_j for all l with $1 \leqslant l \leqslant k$, *the operational cost minimization problem for fair request rate allocation* in this distributed cloud computing environment is to fairly allocate a fractional request rate $r_{j,i}^{(l)} \in \mathbb{Z}$ of $r_j^{(l)}$ from each web portal WP_j to each data center DC_i such that *the system throughput*, $\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{l=1}^{k} r_{j,i}^{(l)}$, is maximized (or equivalently, the number of servers $n_i = \sum_{l=1}^{k} n_i^{(l)}$ at each DC_i to be switched on where $0 \leqslant n_i \leqslant N_i$), while the operational cost of achieving the system throughput $\sum_{i=1}^{N} p_i \sum_{j=1}^{M} \sum_{l=1}^{k} \alpha_i^{(l)} r_{j,i}^{(l)} + \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{l=1}^{k} p_j^b r_{j,i}^{(l)}$ is minimized, subject to that the SLA requirements of allocated requests, $\sum_{i=1}^{N} r_{j,i}^{(l)} \leqslant r_j^{(l)}$, and $\sum_{l=1}^{k}\sum_{i=1}^{N} r_{j,i}^{(l)} \leqslant r_j$, where $1 \leqslant i \leqslant N$ and $1 \leqslant j \leqslant M$.

The problem objective in this paper is to ensure that each web portal has the same proportion λ of numbers of requests with delay D_l to be served and the value of λ is as large as possible while the operational cost of processing the allocated requests is minimized, i.e., the objective is to maximize $\sum_{j=1}^{M}\sum_{l=1}^{k}\left(\lambda \cdot r_j^{(l)}\right) = \sum_{j=1}^{M} \lambda \cdot r_j = \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{l=1}^{k} r_{j,i}^{(l)}$, while the operational cost is minimized, where $r_j = \sum_{l=1}^{k} r_j^{(l)}$ and $0 < \lambda \leqslant 1$.

3.5. The minimum cost multicommodity flow problem [17]

Given a directed graph $G = (V, E; u, c)$ with capacities $u : E \mapsto \mathbb{R}^+$ and costs $c : E \mapsto \mathbb{R}^{\geqslant 0}$, assume that there are n source–destination pairs $(s_i, t_i; d_i)$ where d_i is the amount of demands to be routed from source node s_i to destination node t_i for all i with $1 \leqslant i \leqslant n$. Let $B\,(> 0)$ be a given budget and $|f_i|$ the amount of flow f_i sent from s_i to t_i. The *minimum cost multicommodity flow problem* in G is to find the largest λ such that there is a multicommodity flow f_i routing $\lambda \cdot d_i$ units of commodity i from s_i to t_i for each i with $1 \leqslant i \leqslant n$, subject to the flow constraint $\sum_{i=1}^{n} |f_i(e)| \leqslant u(e)$ and the budget constraint $\sum_{e \in E} c(e) \cdot |f(e)| \leqslant B$, where $f_i(e)$ is the flow of commodity i from s_i to t_i on edge $e \in E$ and $|f(e)| = \sum_{i=1}^{n} |f_i(e)|$.

The optimization framework for the minimum cost multicommodity flow problem given by Garg and Könemann is as follows. It first formulates the problem as a linear programming **LP**, then finds an approximate solution to the duality **DP** of the **LP** which returns an approximate solution to the original problem. Let $\mathcal{P}_i$ be the set of paths in G from s_i to t_i, and let $\mathcal{P} = \cup_{i=1}^{n} \mathcal{P}_i$. Variable f_p represents the flow on path $p \in \mathcal{P}$. The linear programming formulation of the problem **LP** is

$$\begin{aligned}
\textbf{LP}: \quad & \max \quad \lambda \\
\text{s.t.} \quad & \sum_{e \in p} f_p \leqslant u(e) && \forall e \in E, \\
& \sum_{i=1}^{n} \sum_{p \in \mathcal{P}_i} \sum_{e \in p} (f_p \cdot c(e)) \leqslant B \\
& \sum_{p \in \mathcal{P}_i} f_p \geqslant \lambda \cdot d_i && \forall i,\ 1 \leqslant i \leqslant n, \\
& f_p \geqslant 0 && \forall p \in \mathcal{P}, \\
& 0 \leqslant \lambda \leqslant 1.
\end{aligned}$$

The dual linear programming **DP** of the **LP** is described as follows, where $l(e)$ is the *length* on edge $e \in E$ and ϕ is treated as the *length* of the cost constraint B.

$$\begin{aligned}
\textbf{DP}: \quad & \min \quad D(l, \phi) = \sum_{e \in E} l(e) u(e) + B \cdot \phi \\
\text{s.t.} \quad & \sum_{e \in p} (l(e) + c(e) \cdot \phi) \geqslant z_i && \forall p \in \mathcal{P}_i, \\
& \sum_{i=1}^{k} d_i \cdot z_i \geqslant 1, \\
& l(e) \geqslant 0 && \forall e \in E, \\
& z_i \geqslant 0 && \forall i,\ 1 \leqslant i \leqslant n.
\end{aligned}$$

Garg and Könemann's optimization framework for **DP** proceeds in a number of *phases*. Each phase is composed of exactly n iterations, corresponding to the n commodities. Within each iteration, there are a number of *steps*. Initially, $l(e) = \frac{\delta}{u(e)}$ for each $e \in E$, $\phi = \delta / B$, and $z_i = \min_{p \in \mathcal{P}_i} \{l(p) + c(p)\phi\}$, where $c(p) = \sum_{e \in p} c(e)$, $\delta = \left(\frac{1-\epsilon}{|E|}\right)^{1/\epsilon}$, and ϵ is the increasing step of the length function in each iteration. In one phase, let i be the current iteration, the algorithm will route d_i units of flow of commodity i from s_i to t_i within a number of steps. In each step, it routes as much fraction of commodity i as possible along a shortest path p from s_i to t_i with length $l(p) + c(p)\phi$. The amount of flow on p is the minimum one, u_{min}, among the three values: the bottleneck capacity of p, the remaining demand d_i' of commodity i, and $B/c(p)$. Once the amount of the flow u_{min} has been routed on p, the dual variables l and ϕ are then updated: $l(e) = l(e)\left(1 + \epsilon \frac{u_{min}}{u(e)}\right)$ and $\phi = \phi\left(1 + \epsilon \frac{u_{min} \cdot c(p)}{B}\right)$. The algorithm continues until $D(l, \phi) \geqslant 1$. A feasible flow f' is finally obtained by scaling the flow f by $\log_{1+\epsilon} \frac{1}{\delta}$ [17].

Theorem 1 (*see [17]*). *There is an approximation algorithm for the minimum cost multicommodity flow problem in directed graph $G = (V, E; u, c)$ with n commodities to be routed from their sources to their destinations, which delivers a solution with an approximation ratio of $(1 - 3\epsilon)$ while the associated cost is the minimum. The algorithm takes $O^*(\epsilon^{-2} m(n + m))$ time,*[1] *where $m = |E|$ and ϵ is a given constant with $0 < \epsilon \leqslant 1/3$.*

[1] $O^*(f(n)) = O\left(f(n) \log^{O(1)} n\right)$.

4. Algorithm for the operational cost minimization problem

In this section, we first propose a novel operational cost optimization framework for the operational cost minimization problem. We then develop a fast approximation algorithm with the guaranteed approximation ratio, based on the proposed optimization framework. We finally analyze the time complexity of the proposed algorithm and its approximation ratio.

4.1. The operational cost optimization framework

The basic idea behind the proposed framework is to reduce the problem to a minimum cost multicommodity flow problem in another flow network $G_{f,k} = (V_f, E_f; u, c)$ with cost function $c : E \mapsto \mathbb{R}^{\geqslant 0}$ and capacity function $u : E \mapsto \mathbb{R}^{+}$ as follows.

For each web portal WP_j, there are k corresponding nodes $WP_j^{(l)}$ in $G_{f,k}$ with each representing the number of requests with the average delay D_l in WP_j, where $1 \leqslant l \leqslant k$. For each data center DC_i, there are k corresponding nodes $DC_i^{(l)}$ in $G_{f,k}$ with each representing the number of servers in DC_i to be allocated for processing the requests of delay D_l, where $1 \leqslant l \leqslant k$. Also, the flow network $G_{f,k}$ contains a 'virtual source node' s_0 and a 'virtual destination node' t_0 to represent the source and the destination of all requests of web portals. The set of nodes in $G_{f,k}$ is $V_f = \{DC_i | 1 \leqslant i \leqslant N\} \cup \{DC_i^{(l)} | 1 \leqslant l \leqslant k,\ 1 \leqslant i \leqslant N\} \cup \{WP_j | 1 \leqslant j \leqslant M\} \cup \{WP_j^{(l)} | 1 \leqslant l \leqslant k,\ 1 \leqslant j \leqslant M\} \cup \{s_0, t_0\}$, and the set of edges in $G_{f,k}$ is $E_f = \{\langle s_0, WP_j\rangle | 1 \leqslant j \leqslant M\} \cup \{\langle WP_j, WP_j^{(l)}\rangle | 1 \leqslant l \leqslant k,\ 1 \leqslant j \leqslant M\} \cup \{\langle WP_j^{(l)}, DC_i^{(l)}\rangle | 1 \leqslant j \leqslant M,\ 1 \leqslant i \leqslant N,\ 1 \leqslant l \leqslant k\} \cup \{\langle DC_i^{(l)}, DC_i\rangle | 1 \leqslant l \leqslant k\} \cup \{\langle DC_i, t_0\rangle | 1 \leqslant i \leqslant N\}$, as illustrated in Fig. 2.

The cost of each edge in $\{\langle WP_j^{(l)}, DC_i^{(l)}\rangle | 1 \leqslant l \leqslant k,\ 1 \leqslant j \leqslant M, 1 \leqslant i \leqslant N\}$ is the sum of the network bandwidth cost and the electricity cost of processing a single request of delay D_l in data center DC_i. Zero costs are associated with all edges in $\{\langle s_0, WP_j\rangle | 1 \leqslant j \leqslant M\} \cup \{\langle WP_j, WP_j^{(l)}\rangle | 1 \leqslant l \leqslant k, 1 \leqslant j \leqslant M\} \cup \{\langle DC_i^{(l)}, DC_i\rangle | 1 \leqslant l \leqslant k, 1 \leqslant i \leqslant N\} \cup \{\langle DC_i, t_0\rangle | 1 \leqslant i \leqslant N\}$.

The edge capacity of each edge in $G_{f,k}$ is assigned as follows. The key here is to transfer the average delay requirements of different requests to the capacity constraints on edges of $G_{f,k}$. Consider the requests with delay D_l to be processed in data center DC_i, the number of waiting requests in the lth waiting queue of DC_i should be limited, as large numbers of waiting requests in the queue will lead to the average waiting time greater than D_l, the upper bound on the number of 'admissible' requests $R_i^{(l)}$ at DC_i thus is calculated by

$$R_i^{(l)} \leqslant \left\lfloor n_i^{(l)} \mu_i - \frac{1}{D_l} \right\rfloor. \tag{5}$$

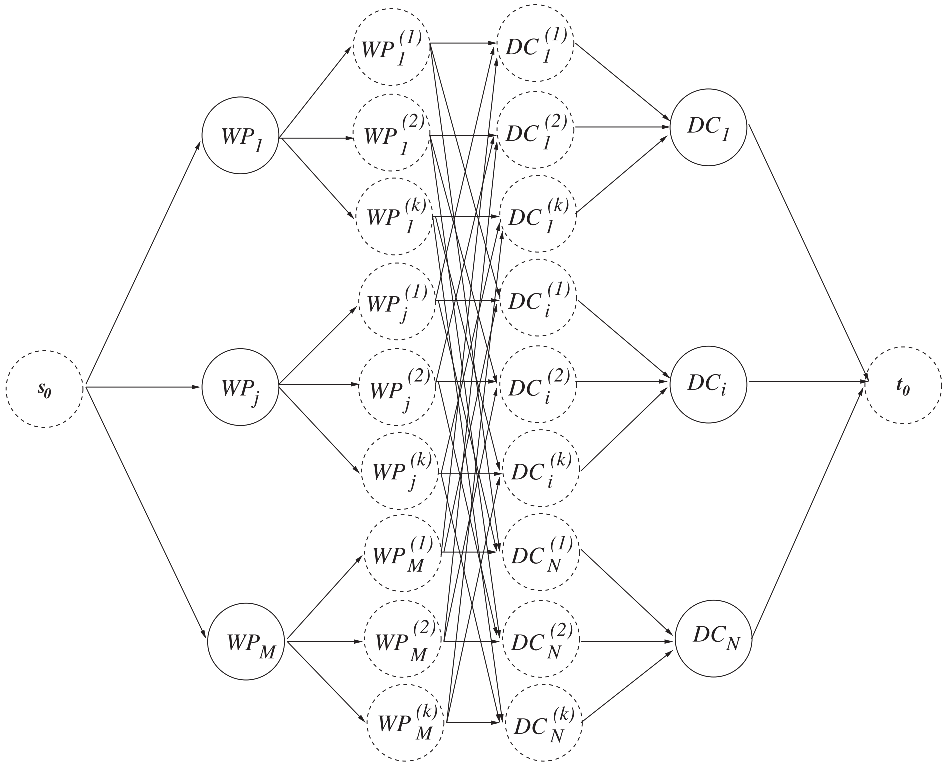


Fig. 2. $G_{f,k} = (V_f, E_f; u, c)$.

When all servers in data center DC_i are dedicated to process the requests with delay D_l ($n_i^{(l)} = N_i$), the maximum number of requests with delay D_l can be admitted by DC_i is $\lfloor N_i\mu_i - 1/D_l \rfloor$. The capacity on edge $\langle WP_j^{(l)}, DC_i^{(l)} \rangle$ and $\langle DC_i^{(l)}, DC_i \rangle$ thus is set as $u(WP_j^{(l)}, DC_i^{(l)}) = u\left(DC_i^{(l)}, DC_i\right) = \lfloor N_i\mu_i - 1/D_l \rfloor$ for each $l \in [1, k]$.

We now determine the capacity of every edge $\langle DC_i, t_0 \rangle$. Since each data center DC_i processes k types of requests with k different delays, the capacity of every edge $\langle DC_i, t_0 \rangle$ has k alternative capacity settings, i.e., $u(DC_i, t_0) = \lfloor \mu_i N_i - k/D_l \rfloor$ for all $l \in [1, k]$. For each of these k settings $l \in [1, k]$, assign each edge $\langle DC_i, t_0 \rangle$ a capacity of $\lfloor \mu_i N_i - k/D_l \rfloor$. We later discuss the relationship between these k capacity settings and a feasible solution to the problem. The capacities of all other edges in E_f are set to infinity.

Following the construction of $G_{f,k}$, the original problem can be transformed into a minimum cost multicommodity flow problem as follows. There are at most kM commodities (request rates with k delay requirements) to be routed. For a request rate r_j starting from WP_j, it consists of at most k request rates $r_j^{(l)} \geqslant 0$ of delay D_l, where $\sum_{l=1}^{k} r_j^{(l)} = r_j$. Each such request rate is considered as a commodity in $G_{f,k}$ to be routed to the destination node t_0. Thus, there are in total kM commodities in $G_{f,k}$ from M web portal nodes (source nodes) to be routed to their common destination node t_0, i.e., there are kM triples $\left(WP_j^{(l)}, t_0; r_j^{(l)}\right)$ in $G_{f,k}$ for all l with $1 \leqslant l \leqslant k$, where $WP_j^{(l)}$ is the source node, t_0 is the destination node, and $r_j^{(l)}$ is the amount of demands from the source node to be routed. Suppose that f is a flow from s_0 to t_0 that routes the kM commodities from different sources to their common destination t_0, and the constructed flow network $G_{f,k}$ satisfies the flow conservation constraint: for all $u, v \in V_f - \{s_0, t_0\}$, we have $\sum_{v \in V_f} f(v, u) = \sum_{v \in V_f} f(u, v)$. We then have the following lemma.

Lemma 1. *The constructed flow network $G_{f,k}$ has the following properties:*

(a) *The amount of (request) flow entering into node $DC_i^{(l)}$ for all i and l with $1 \leqslant i \leqslant N$ and $1 \leqslant l \leqslant k$, $\sum_{l=1}^{k} \left| f\left(DC_i^{(l)}, DC_i\right) \right| = |f(DC_i, t_0)|$, will be admitted by data center DC_i. The number of active servers needed for processing the admitted requests with the average delay D_l in DC_i is $n_i^{(l)} = \left\lfloor \frac{1}{\mu_i}\left(\left| f\left(DC_i^{(l)}, DC_i\right) \right| + \frac{1}{D_l} \right) \right\rfloor, 1 \leqslant l \leqslant k$.*

(b) *Let p and p' be two different paths of flow f that route two different SLA requests with delays D_l and $D_{l'}$ and $r^{(l)}$ and $r^{(l')}$ the admitted request rates routed by path p and p', respectively. Then, no requests in $r^{(l)}$ ($r^{(l')}$) will be sent to the l'th (lth) waiting queue of the same data center.*

(c) *Each path p from s_0 to t_0 in flow f corresponds to an allocation of a portion of requests with a delay requirement, and the cost of that portion of the flow along p corresponds to the operational cost of processing the portion of number of requests.*

Proof. We start by showing Property (a). Let $R_i^{(l)}(t)$ be the number of requests admitted by node $DC_i^{(l)}$ at time slot t through the edge $\left\langle WP_j^{(l)}, DC_i^{(l)} \right\rangle$. According to the construction of $G_{f,k}$, there is only one outgoing edge of node $DC_i^{(l)}$, i.e., $\left\langle DC_i^{(l)}, DC_i \right\rangle$, all these requests will flow into node DC_i through this edge. Thus, each request entering node $DC_i^{(l)}$ will be admitted by node DC_i. The number of such requests that will be processed by data center DC_i is $\left| f\left(DC_i^{(l)}, DC_i\right) \right|$, since there is only one outgoing edge of $DC_i^{(l)}$. That is,

$$R_i^{(l)}(t) = \left| f\left(DC_i^{(l)}, DC_i\right) \right|. \tag{6}$$

Recall that $D_i^{(l)}\left(n_i^{(l)}(t), R_i^{(l)}(t)\right)$ denotes the average delay of realized requests in $R_i^{(l)}(t)$. The number of active servers needed for processing these requests thus must be the number of servers that can exactly fulfill the average delay requirement of these requests, i.e., $D_i^{(l)}\left(n_i^{(l)}(t), R_i^{(l)}(t)\right) = D_l$. By (1), (2), and (6), we have

$$n_i^{(l)} = \left\lfloor \frac{1}{\mu_i}\left(\left| f\left(DC_i^{(l)}, DC_i\right) \right| + \frac{1}{D_l} \right) \right\rfloor, \quad 1 \leqslant l \leqslant k. \tag{7}$$

We then show Property (b) by showing that no requests of delay $D_{l'}$ will go through node $DC_i^{(l)}$ with $l' \neq l$. Following the construction of $G_{f,k}$, only the nodes in $\left\{ WP_j^l | 1 \leqslant j \leqslant M,\ 1 \leqslant l \leqslant k \right\}$ with delay D_l are connected to node $DC_i^{(l)}$, which means that each node $WP_j^{(l')}$ cannot connect to any node $DC_i^{(l)}$ in which the lth queue is located when $l' \neq l$. Thus, no requests in $r^{(l)}$ ($r^{(l')}$) will be sent to the l'th (lth) queue in the same data center.

We finally prove Property (c) by proving that the number of requests routed by each directed path p in $G_{f,k}$ from s_0 to t_0 will be admitted by a data center and their average delay requirements will be met. To this end, we treat each such a path p consisting of three segments: segment 1 starting from s_0 and ending at node $WP_j^{(l)}$, segment 2 starting from node $WP_j^{(l)}$ and ending at node $DC_i^{(l)}$, and segment 3 starting from node $DC_i^{(l)}$ and ending at node t_0. In segment 1, path p leaves from a distinct web portal node and then ends at one node in $\left\{ WP_j^{(l)} | 1 \leqslant j \leqslant M,\ 1 \leqslant l \leqslant k \right\}$. This means that these requests routed by path p are with the average delay requirement D_l, if the ending node in segment 1 is $WP_j^{(l)}$. Segment 2 leads the flow to a node $DC_i^{(l')}$ with $l' = l$. Finally, path p will direct its admitted requests to a single data center in segment 3 that corresponds to a request rate allocation. For the incurred operational cost of requests along path p, if the requests are admitted by DC_i, the corresponding flow will enter a node $DC_i^{(l)}$ for some l with $1 \leqslant l \leqslant k$. Following the construction of $G_{f,k}$, the operational cost incurred by processing these requests corresponding to the flow in p is the cost incurred along path p. $\square$

Recall that there are k different capacity settings for each edge from DC_i to t_0 in $G_{f,k}$ for all i with $1 \leqslant i \leqslant N$. We have an important observation about $G_{f,k}$ regarding the relationship between a flow f in $G_{f,k}$ from s_0 to t_0 and a corresponding feasible solution to the problem by the following lemma.

Lemma 2. *Let $G^{(l)}_{f,k}$ be the flow network when the edge capacity of each edge from DC_i to t_0 is assigned a value $\lfloor \mu_i N_i - k/D_l \rfloor$, i.e., $u(DC_i, t_0) = \lfloor \mu_i N_i - k/D_l \rfloor$ for all i and l with $1 \leqslant i \leqslant N$ and $1 \leqslant l \leqslant k$. Let $f^{(l)}$ be the flow in $G^{(l)}_{f,k}$ for the kM commodities. Then, (i) if $l = 1$, there is a feasible solution to the operational cost minimization problem. (ii) otherwise ($l \geqslant 2$), there may or may not have a feasible solution to the operational cost minimization problem.*

Proof. We show claim (i) by showing that the number of active servers n_i to process the requests entering each data center DC_i is no greater than the available number of servers N_i in it. Let $n^{(l)}_i$ be the number of servers in DC_i dedicated to the requests with delay D_l. Clearly,

$$n_i = \sum_{l=1}^{k} n^{(l)}_i \tag{8}$$

$$= \sum_{l=1}^{k} \left\lceil \frac{1}{\mu_i}\left(|f^{(l)}\left(DC^{(l)}_i, DC_i\right)| + \frac{1}{D_l}\right)\right\rceil \quad \text{by Property 1 of Lemma 1}$$

$$\leqslant \left\lceil \sum_{l=1}^{k} \frac{1}{\mu_i}\left(\left|f^{(l)}\left(DC^{(l)}_i, DC_i\right)\right| + \frac{1}{D_l}\right)\right\rceil = \left\lceil \frac{1}{\mu_i}|f^{(l)}(DC_i, t_0)| + \frac{1}{\mu_i}\sum_{l=1}^{k}\frac{1}{D_l}\right\rceil \tag{9}$$

$$\leqslant \frac{1}{\mu_i}|f^{(l)}(DC_i, t_0)| + \frac{1}{\mu_i}\sum_{l=1}^{k}\frac{1}{D_l} \tag{10}$$

$$\leqslant \frac{1}{\mu_i}u(DC_i, t_0) + \frac{1}{\mu_i}\sum_{l=1}^{k}\frac{1}{D_l} \quad \text{since } |f^{(l)}(DC_i, t_0)| \leqslant u(DC_i, t_0), \tag{11}$$

$$\leqslant \frac{1}{\mu_i}u(DC_i, t_0) + \frac{k}{\mu_i D_1} \quad \text{since } D_1 < D_l \text{ with } l \geqslant 2, \tag{12}$$

$$\leqslant N_i \quad \text{since } u(DC_i, t_0) = \left\lfloor N_i\mu_i - \frac{k}{D_1}\right\rfloor.$$

We now show claim (ii). Following equations and inequalities from (8)–(11), $n_i \leqslant \frac{1}{\mu_i}\left(u(DC_i, t) + \sum_{l=1}^{k}\frac{1}{D_l}\right)$. However, if the edge capacity of each edge from DC_i to t_0 is $\lfloor \mu_i N_i - k/D_l \rfloor$ for some l with $l \geqslant 2$, then there is no guarantee that $\frac{k}{D_l} \geqslant \sum_{l=1}^{k}\frac{1}{D_{l'}}$, since $\frac{1}{D_{l'}} \geqslant \frac{1}{D_l}$ for all l' with $l' \leqslant l$, but $\frac{1}{D_{l'}} < \frac{1}{D_l}$ for all l' with $l' > l$. That is, the number of active servers n_i needed in some DC_i may be greater than the number of servers N_i provided by the data center. □

4.2. Approximation algorithm

The proposed algorithm proceeds in k iterations with each for a different edge capacity setting, $u(DC_i, t_0) = \lfloor \mu_i N_i - k/D_l \rfloor$ for all i and l with $1 \leqslant i \leqslant N$ and $1 \leqslant l \leqslant k$. A maximum flow $f^{(l)}$ from s_0 to t_0 is then found which may or may not be a feasible solution to the problem by Lemma 2. To obtain a feasible solution to the problem, the flow $f^{(l)}$ needs to be scaled by an appropriate factor so that the resulting flow $f'^{(l)}$ is feasible as follows.

Algorithm 1. An approximation algorithm for the request allocation at time slot t.

```
Input: A set of data centers $\{DC_i | 1 \leqslant i \leqslant N\}$, a set of web portals $\{WP_j | 1 \leqslant j \leqslant M\}$, request arrival rate $r_j = \sum_{l=1}^{k} r^{(l)}_j$ at web portal $WP_j$ for each $j$ with $1 \leqslant j \leqslant M$, the set $\{p_i(t) | 1 \leqslant i \leqslant N\}$ of electricity prices for all data centers, the set $\{p^b_j | 1 \leqslant j \leqslant M\}$ of bandwidth costs, and the accuracy parameter $\epsilon$ with $0 < \epsilon \leqslant 1$.
Output: The minimum operational cost $C$, the request assignment $\left\{r^{(l)}_{j,i} | 1 \leqslant l \leqslant k,\ 1 \leqslant j \leqslant M,\ 1 \leqslant i \leqslant N\right\}$ such that $\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{l=1}^{k} r^{(l)}_{j,i}$ is maximized and the number of servers $n_i = \sum_{l=1}^{k} n^{(l)}_i$ ($\leqslant N_i$) in each $DC_i$ to be switched on, where $\sum_{i=1}^{N}\sum_{l=1}^{k} r^{(l)}_{j,i} \leqslant r_j$.
1: Construct an auxiliary flow network $G_{f,k}$, in which there are $kM$ commodities to be routed from their sources to the destination $t_0$, $\left\{\left(WP_j, t_0, r^{(l)}_j\right) | 1 \leqslant l \leqslant k, 1 \leqslant j \leqslant M\right\}$;
2: $B \leftarrow \sum_{j=1}^{M} r_j \cdot \left(\max_{1 \leqslant i \leqslant N}\{p_i(t)\}\max_{1 \leqslant l \leqslant k}\{\alpha^{(l)}_i\} + p^b_j\right)$ /* The upper bound on the total operational cost */;
3: $\mathcal{F} \leftarrow \emptyset$; /* the set of routing paths from $s_0$ to $t_0$ that corresponds to the feasible flow. */
4: for $l \leftarrow 1$ to $k$ do
5:    $u(DC_i, t_0) \leftarrow \lfloor \mu_i N_i - k/D_l \rfloor$;
6:    Let $f^{(l)}$ be the flow delivered by applying Garg and Könemann's algorithm for the minimum cost multicommodity flow problem in $G^{(l)}_{f,k}$, and let $f^{(l)}(e)$ be the fraction of the flow on each edge $e \in E_f$. Let $F^{(l)}_i$ be the amount of flow entering into node $DC_i$ in $G^{(l)}_{f,k}$;
7:    if $l = 1$ then
8:       $F^{(1)}_i \leftarrow \sum_{l'=1}^{k}\left|f^{(1)}\left(DC^{(l')}_i, DC_i\right)\right|$, for all $i$ with $1 \leqslant i \leqslant N$;
9:    else
10:      for each data center $DC_i$ do
11:         $N'_i \leftarrow \sum_{l'=1}^{k}\left\lceil \frac{\left|f^{(l)}\left(DC^{(l)}_i, DC_i\right)\right|}{\mu_i} + \frac{1}{\mu_i D_{l'}}\right\rceil$;
12:         If $N'_i > N_i$, $\rho_i \leftarrow \frac{F^{(1)}_i}{\left|\sum_{l'=1}^{k} f^{(l)}\left(DC^{(l')}_i, DC_i\right)\right|}$; otherwise, $\rho_i \leftarrow 1$;
13:      end for
14:      $\rho_{min} \leftarrow min\{\rho_i | 1 \leqslant i \leqslant N\}$;
15:      $|f'^{(l)}(e)| \leftarrow |f^{(l)}(e)|\rho_{min}, \forall e \in E_f$; /* scale the flow */
16:   end if
17:   Add the feasible flow $f'^{(l)}$ into $\mathcal{F}$;
18: end for
19: Choose flow $f'$ that delivers the maximum number of requests in $\mathcal{F}$ as the final feasible flow;
20: for each data center $DC_i \in \mathcal{DC}$ do
21:   $n_i \leftarrow \sum_{l=1}^{k}\left\lceil \frac{\left|f'\left(DC^{(l)}_i, DC_i\right)\right|}{\mu_i} + \frac{1}{\mu_i D_l}\right\rceil$; /* the number of servers in $DC_i$ to be switched on */
22: end for
23: $C \leftarrow \sum_{e \in E_f} c(e) \cdot |f'(e)|$;
```

Let $f^{(l)}$ be the flow from s_0 to t_0 in $G^{(l)}_{f,k}$ when $u(DC_i, t_0) = \lfloor \mu_i N_i - k/D_l \rfloor$. Let $F^{(l)}_i$ be the amount of fractional flow of $f^{(l)}$ entering node DC_i in $G^{(l)}_{f,k}$ with capacity $u(DC_i, t_0) = \lfloor \mu_i N_i - k/D_l \rfloor$ and N'_i the corresponding number of servers needed for processing the flow (or the corresponding request flow), while the available number of servers in DC_i is N_i $(< N'_i)$.

Denote by $\rho^{(l)}_i = \frac{F^{(1)}_i}{F^{(l)}_i}$ the flow ratio at DC_i when the edge capacity of each edge from DC_i to t_0 is $\lfloor \mu_i N_i - k/D_l \rfloor$, the value of the resulting flow entering into node DC_i, $F'^{(l)}_i = \rho^{(l)}_i F^{(l)}_i \leqslant F^{(1)}_i$, is feasible as there are enough numbers of available servers in DC_i for realizing the scaled flow, by Lemma 2. Notice that in terms of the "fair" request rate allocation, we scale not only the flow entering into node DC_i but also other flows from other web portals by the same ratio. Let ρ_{min} be the scale ratio, then

$$\rho_{min} = \min\{\rho^{(l)}_i \mid 1 \leqslant i \leqslant N\}. \tag{13}$$

A feasible flow with value $\rho_{min} \cdot |f^{(l)}|$ is then obtained, and the cost of the scaled flow is no greater than the optimal cost C^* of the optimal flow which will shown in Theorem 2 later. The detailed approximation algorithm is described in Algorithm 1.

The obtained solution can be further improved if possible. That is, the value of ρ_{min} can be maximized so that the value of the resulting flow $\rho_{min} \cdot |f^{(l)}|$ is still the value of a feasible solution to the problem. To this end, the algorithm proceeds iteratively until the increased percentage of the feasible flow at the current iteration is no greater than a given threshold $\xi, 0 < \xi < 1$, compared with the value in its last iteration through binary search. Given the value of ρ_{min} at the current iteration, we examine each DC_i to see whether the requested number of servers for the processing of admitted requests corresponding to the value $\rho_{min} \cdot F^{(l)}_i$ is available. If so, the next iteration proceeds. The algorithm terminates when either the ratio difference is bounded by ξ or there is at least one DC_i that does not support the scaled flow. Thus, the number of iterations of Algorithm 1 is bounded by $\left\lceil \log \frac{(1-\rho_{min})}{\xi} \right\rceil = \lceil \log(1 - \rho_{min}) - \log \xi \rceil = O(1)$. In the real implementation of the proposed algorithm, the initial value of ρ_{min} can be set $\min\left\{ \frac{\lfloor \mu_i N_i - k/D_1 \rfloor}{\lfloor \mu_i N_i - k/D_k \rfloor} \mid 1 \leqslant i \leqslant N \right\}$ instead of ρ_{min}.

4.3. Algorithm performance analysis

In this subsection we analyze the approximation ratio and time complexity of the proposed algorithm by following lemmas and a theorem.

Lemma 3. *Let $G^{(1)}_{f,k}$ be the flow network when the capacity of every edge from DC_i to t_0 is $\lfloor \mu_i N_i - k/D_1 \rfloor$, i.e., $u^{(1)}(DC_i, t_0) = \lfloor \mu_i N_i - k/D_1 \rfloor$ for all i with $1 \leqslant i \leqslant N$. Let $f^{(1)}$ and $F^{(1)}_i$ be the feasible flow from s_0 to t_0 in $G^{(1)}_{f,k}$ and the amount of flow in edge $\langle DC_i, t_0 \rangle$ of $f^{(1)}$. For all i with $1 \leqslant i \leqslant N$, if all data centers run in the full capacities, i.e., the number of requests is far larger than the processing capacities of all data centers, we have*

$$\frac{F^{(1)}_i}{u^{(1)}(DC_i, t_0)} \geqslant \frac{1}{2}. \tag{14}$$

Proof. Assume that the claim is not true, then edge $\langle DC_i, t_0 \rangle$ is not saturated, which means that more flow can go through the edge, this contradicts the fact that all servers in DC_i have been switched on already. As the value of flow is an integer (the number of requests), then, $u^{(1)}(DC_i, t_0) - F^{(1)}_i \geqslant 0$ and $F^{(1)}_i \geqslant 1$. For a small fraction x with $0 \leqslant x < 1$, we have $\frac{F^{(1)}_i}{u^{(1)}(DC_i,t_0)} \geqslant \frac{F^{(1)}_i}{F^{(1)}_i + x} \geqslant \frac{F^{(1)}_i}{F^{(1)}_i + 1} = \frac{1}{1+1/F^{(1)}_i} \geqslant \frac{1}{2}$. The lemma holds. □

Lemma 4. *Let $u^{(1)}(DC_i, t_0)$ and $u^{(l)}(DC_i, t_0)$ be the edge capacity of every edge from DC_i to t_0 when it is set to $\lfloor \mu_i N_i - k/D_1 \rfloor$ and $\lfloor \mu_i N_i - k/D_l \rfloor$ with $l \neq 1$, respectively. We have $u^{(l)}(DC_i, t_0) - u^{(1)}(DC_i, t_0) < \frac{k}{D_1} - \frac{k}{D_l} + 1$.*

Proof.

$$\begin{aligned}
u^{(l)}(DC_i, t_0) - u^{(1)}(DC_i, t_0) &= \left\lfloor N_i \mu_i - \frac{k}{D_l} \right\rfloor - \left\lfloor N_i \mu_i - \frac{k}{D_1} \right\rfloor \\
&\leqslant N_i \mu_i - \frac{k}{D_l} - \left\lfloor N_i \mu_i - \frac{k}{D_1} \right\rfloor \\
&< N_i \mu_i - \frac{k}{D_l} + -N_i \mu_i + \frac{k}{D_1} + 1 \\
&\quad \text{since } \left\lfloor N_i \mu_i - \frac{k}{D_1} \right\rfloor > N_i \mu_i - \frac{k}{D_1} - 1, \\
&= \frac{k}{D_1} - \frac{k}{D_l} + 1. \qquad \square
\end{aligned}$$

Theorem 2. *In a distributed cloud computing environment consisting of M web portals and N data centers, given different user requests with k different average delay requirements at each web portal, there is an approximation algorithm for the operational cost minimization problem. The approximation ratio on the system throughput delivered by the algorithm is $(1/2 - \epsilon')$ while the operational cost of achieving the system throughput is no more than the operational cost of the optimal solution C^*. The time complexity of the proposed algorithm is $O^*(\epsilon'^{-2} k^3 M^2 N^2)$, where ϵ' is a constant with $0 < \epsilon' \leqslant 1/2$.*

Proof. We first show that the proportionally accumulative number of requests $r_1, r_2, \ldots, r_M$ with delay requirements $D_1, D_2, \ldots, D_k$ from the M web portals will be admitted by the system while their SLAs will be met too. Since the edge capacity of each edge in $G_{f,k}$ from $DC^{(l)}_i$ to DC_i is calculated by bounding the number of requests in the lth waiting queue, we only need to show that the value of a feasible flow f' is no greater than the edge capacities and the number of servers to be switched on is no greater than the server capacity at each data center DC_i by two cases: (i) the capacity of each edge from node DC_i to t_0 is $\lfloor \mu_i N_i - k/D_1 \rfloor$ for all i with $1 \leqslant i \leqslant N$; or (ii) the capacity of each edge from DC_i to t_0 is $\lfloor \mu_i N_i - k/D_l \rfloor$ for all i with $1 \leqslant i \leqslant N$ and some l with $2 \leqslant l \leqslant k$.

By Lemma 2, the number of servers to be switched on n_i in case (i) is no greater than the available number of servers at DC_i. In case (ii), although the initial flow obtained $f^{l)}$ may not be feasible, it will be feasible by scaling a factor of ρ_{min}. Thus, all allocated requests with k different delays to different data centers will be processed by these data centers within the specified delays of the admitted requests.

We then analyze the approximation ratio of Algorithm 1. Let f' and f^* be a feasible and optimal solutions of the problem. Let f be the flow in $G^{(l)}_{f,k}$ delivered by Garg and Könemann's algorithm, then $|f| \geqslant (1-3\epsilon)|f^*|$ by Theorem 1, where ϵ is the accuracy parameter. Note that if f is not a feasible solution, it becomes feasible by scaling a factor of ρ_{min}. Thus, there is a feasible solution f' to the problem (i.e., the system throughput $|f'| \geqslant \rho_{min}(1-3\epsilon)|f^*|$).

The rest is to show that $\rho_{min} \geqslant 1/2$ by the following two cases: (i) all data centers are running in their full capacities (i.e., the number of requests at each data center is far greater than the processing capacity of the data center); (ii) each data center is under-utilized.

Recall that $F^{(l)}_i$ is the amount of fractional flow entering into node DC_i in $G^{(l)}_{f,k}$ of flow $f^{(l)}$.

For case (i), we have

$$\begin{aligned}
\rho_{min} &= \min\left\{\frac{F^{(1)}_i}{F^{(l)}_i} | 1 \leqslant i \leqslant N\right\} = \frac{F^{(1)}_{i_0}}{F^{(l)}_{i_0}} \quad \text{suppose} \quad \rho_{i_0} = \rho_{min},\\
&\geqslant \frac{F^{(1)}_{i_0}}{u^{(l)}(DC_{i_0}, t_0)} \quad \text{since} \quad F^{(l)}_{i_0} \leqslant u^{(l)}(DC_{i_0}, t_0),\\
&= \frac{F^{(1)}_{i_0}}{\left\lfloor N_{i_0}\mu_{i_0} + \frac{k}{D_l}\right\rfloor} \quad \text{since} \quad u^{(l)}(DC_{i_0}, t_0) = \left\lfloor N_{i_0}\mu_{i_0} + \frac{k}{D_l}\right\rfloor,\\
&\geqslant \frac{F^{(1)}_{i_0}}{\left\lfloor N_{i_0}\mu_{i_0} + \frac{k}{D_1}\right\rfloor} \quad \text{since} \quad D_l \geqslant D_1,\\
&\geqslant \frac{\frac{1}{2}u^{(1)}(DC_{i_0}, t_0)}{u^{(1)}(DC_{i_0}, t_0)} \quad \text{by Lemma 3}\\
&= \frac{1}{2}.
\end{aligned}$$

For case (ii), if $F^{(l)}_i \leqslant u^{(l)}(DC_i, t_0) - \sum_{l=1}^{k}\frac{1}{D_l} + \frac{k}{D_k}$ for any i with $1 \leqslant i \leqslant N$, we have

$$\begin{aligned}
n_i &\leqslant \frac{1}{\mu_i}\left(|f^{(l)}(DC_i, t_0)| + \sum_{l=1}^{k}\frac{1}{D_l}\right) \quad \text{following Equations from (8)–(10)},\\
&= \frac{1}{\mu_i}\left(F^{(l)}_i + \sum_{l=1}^{k}\frac{1}{D_l}\right) \quad \text{since } F^{(l)}_i = |f^{(l)}(DC_i, t_0)|,\\
&\leqslant \frac{1}{\mu_i}u^{(l)}(DC_i, t_0) - \frac{1}{\mu_i}\left(\sum_{l=1}^{k}\frac{1}{D_l} - \frac{k}{D_k}\right) + \frac{1}{\mu_i}\sum_{i=1}^{k}\frac{1}{D_l}\\
&= \frac{1}{\mu_i}u^{(l)}(DC_i, t_0) + \frac{k}{\mu_i D_k} \leqslant \frac{1}{\mu_i}u^{(l)}(DC_i, t_0) + \frac{k}{\mu_i D_l} \quad \text{since } D_l \leqslant D_k,\\
&\leqslant N_i.
\end{aligned}$$

This implies that the number of active servers needed at each data center DC_i will not exceed its capacity N_i. Therefore, $\rho_{min} = 1$. Otherwise we have $u^{(l)}(DC_i, t_0) - u^{(1)}(DC_i, t_0) < \frac{k}{D_1} - \frac{k}{D_l} + 1$ by Lemma 4. This implies that the amount of flow at each flow path corresponding to the feasible flow $f^{(1)}$ increases by at most $\frac{k}{D_1} - \frac{k}{D_l} + 1$. Thus, $F^{(l)}_i$ is at most $\frac{k}{D_1} - \frac{k}{D_l} + 1$ larger than $|F^{(1)}_i|$, i.e., $F^{(l)}_i \leqslant F^{(1)}_i + \frac{k}{D_1} - \frac{k}{D_l} + 1$. We then have

$$\begin{aligned}
\rho_{min} &= min\left\{\frac{F^{(1)}_i}{F^{(l)}_i} \mid 1 \leqslant i \leqslant N\right\} = \frac{F^{(1)}_{i_0}}{F^{(l)}_{i_0}} \quad \text{suppose } \rho_{i_0} = \rho_{min},\\
&\geqslant \frac{F^{(l)}_{i_0} - \frac{k}{D_1} + \frac{k}{D_l} - 1}{F^{(l)}_{i_0}} \quad \text{since } F^{(l)}_{i_0} \leqslant F^{(1)}_{i_0} + \frac{k}{D_1} - \frac{k}{D_l} + 1,\\
&= 1 - \frac{\frac{k}{D_1} - \frac{k}{D_l} + 1}{F^{(l)}_{i_0}}\\
&\geqslant 1 - \frac{\frac{k}{D_1} - \frac{k}{D_l} + 1}{u^{(l)}(DC_{i_0}, t_0) - \sum_{l=1}^{k}\frac{1}{D_l} + \frac{k}{D_k}} \quad \text{since } F^{(l)}_{i_0}\\
&> u^{(l)}(DC_i, t_0) - \sum_{l=1}^{k}\frac{1}{D_l} + \frac{k}{D_k},\\
&\geqslant 1 - \frac{\frac{k}{D_1} - \frac{k}{D_l} + 1}{u^{(l)}(DC_{i_0}, t_0) - \frac{k}{D_1} + \frac{k}{D_k}} \quad \text{since } D_1 \leqslant D_l,\\
&\geqslant 1 - \frac{\frac{k}{D_1} - \frac{k}{D_l} + 1}{u^{(l)}(DC_{i_0}, t_0) - \left(\frac{k}{D_1} - \frac{k}{D_k} + 1\right)}\\
&\geqslant 1 - \frac{\frac{k}{D_1} - \frac{k}{D_k} + 1}{u^{(l)}(DC_{i_0}, t_0) - \left(\frac{k}{D_1} - \frac{k}{D_k} + 1\right)} \quad \text{since } D_k \geqslant D_l, .
\end{aligned} \tag{15}$$

Let $y = \frac{k}{D_1} - \frac{k}{D_k} + 1$. Since we assume that $\frac{k}{D_l} \ll N_i\mu_i, u^{(l)}(DC_{i_0}, t_0) = \lfloor N_i\mu_i - k/D_l\rfloor \gg y$. Without loss of generality, let $u^{(l)}(DC_{i_0}, t_0) = \beta y$, then $\beta \geqslant 3$ as β is at least the same order of magnitudes as $N_i \cdot \mu_i$. By Eq. (15), we have

$$\rho_{min} \geqslant 1 - \frac{1}{\frac{u^{(l)}(DC_{i_0}, t_0)}{y} - 1} = 1 - \frac{1}{\beta - 1} \geqslant 1/2.$$

Thus, the approximation ratio of the proposed algorithm is $\rho_{min}(1-3\epsilon) = 1/2 - \epsilon'$, where $\epsilon' = (3/2)\epsilon$.

The rest is to show the cost of the feasible solution f'. Let $\mathcal{P}^*(WP^{(l)}_j, t_0)$ be the set of routing paths of the optimal flow $f^{*(l)}_j$ and let $\mathcal{P}^*$ the union of sets $\mathcal{P}^*(WP^{(l)}_j, t_0)$ where each routing path $p \in \mathcal{P}^*(WP^{(l)}_j, t_0)$ routes a part of the demands $r^{(l)}_j$ with delay D_l from $WP^{(l)}_j$ to t_0. Then, $\mathcal{P}^* = \cup_{j=1}^{M}\cup_{l=1}^{k}\mathcal{P}^*\left(WP^{(l)}_j, t_0\right)$. Let $\mathcal{P}''\left(WP^{(l)}_j, t_0\right)$ be a subset of $\mathcal{P}^*\left(WP^{(l)}_j, t_0\right)$ such that the value of the flow of all routing paths in the set is equal to $|f|$, i.e., there is the same fraction of demands d_j with delay D_l as the feasible flow $f^{(l)}_j$ routed from $WP^{(l)}_j$ to t_0, and let $f''(e)$ be the flow on each edge e in paths of $\mathcal{P}''\left(WP^{(l)}_j, t_0\right)$. Note that one of the routing paths in $\mathcal{P}''\left(WP_j^{(l)}, t_0\right)$ may need to reduce the value of its flow in order to reach the value $|f_j^{(l)}|$. We then have

$$\sum_{e \in E_f} |f(e)| \cdot c(e) \leqslant \sum_{j=1}^{M}\sum_{l=1}^{k} \sum_{e \in p \in \mathcal{P}''\left(WP_j^{(l)}, t_0\right)} |f''(e)| \cdot c(e), \qquad (16)$$

as the feasible flow f is a minimum cost multicommodity flow, where $|f''(e)|$ is the sum of flows in $\cup_{j=1}^{M}\mathcal{P}''\left(WP_j^{(l)}, t_0\right)$ on edge $e \in E_f$.

Meanwhile, following the minimum cost multicommodity flow definition,

$$\sum_{j=1}^{M}\sum_{l=1}^{k} \sum_{e \in p \in \mathcal{P}''(WP_j^{(l)}, t_0)} |f''(e)| \cdot c(e)$$
$$\leqslant \sum_{j=1}^{M}\sum_{l=1}^{k} \sum_{e \in p \in \mathcal{P}^*(WP_j^{(l)}, t_0)} |f^*(e)| \cdot c(e) = C^*. \qquad (17)$$

Combining inequalities (16) and (17), we have

$$\sum_{e \in E_f} |f(e)| \cdot c(e) \leqslant \sum_{j=1}^{M}\sum_{l=1}^{k} \sum_{e \in p \in \mathcal{P}''(WP_j^{(l)}, t_0)} |f''(e)| \cdot c(e) \leqslant C^*. \qquad (18)$$

Flow f is scaled down by ρ_{min} to make it a feasible solution f'. The cost of the feasible solution thus is $\rho_{min}\sum_{e \in E_f}|f(e)| \cdot c(e)$, which is no greater than C^* since $1/2 \leqslant \rho_{min} \leqslant 1$. The time complexity of the proposed algorithm is analyzed as follows. Since the flow network $G_{f,k}(V_f, E_f; u, c)$ consists of $|V_f| = (k+1)(M+N)+2$ nodes and $|E_f| = (k+1)(M+N)+kMN$ edges, its construction takes $O(|V_f|+|E_f|)$ time. Following Theorem 1, the Garg and Könemann's algorithm takes $O^*(\epsilon^{-2}k^2M^2N^2)$ time, where the kM commodities are routed from their sources to the virtual destination node t_0. Since Algorithm 1 is invoked k times with each corresponding one of the k different capacity assignments of the edges from node DC_i to node t_0, the proposed algorithm thus takes $O^*(\epsilon'^{-2}k^3M^2N^2)$, where $\epsilon' = (3/2)\epsilon$. $\square$

5. Performance evaluation

In this section we evaluate the performance of the proposed algorithm through experimental simulations, using real-life electricity price data sets and various request arrival patterns.

5.1. Simulation environment settings

We consider a distributed cloud computing environment consisting of five data centers and six web portals. Following the similar settings in existing studies [5,38,39], each data center hosts 15,000–25,000 homogeneous servers with an identical operating power of 350 W. The service rate of each server is represented by the number of requests processed per second, which varies from 2.55 to 3.75 [38,39]. The bandwidth cost of allocating a request to a data center is from \$0.04 per hour to \$0.08 per hour [4,11,20]. The maximum request rates in web portals vary from 27,900 to 92,000, which is derived by scaling the request rates in [38] according to the processing ability of the data centers. Tables 1 and 2 summarize the information of the data centers and web portals, respectively. The running time is obtained based on a desktop with 2.66 GHz Intel Core 2 Quad CPU and 8 GB RAM. The accuracy parameter ϵ in Garg–Könemann's approximation framework is 0.1 in the default setting, and the difference threshold ξ in binary search for an appropriate ρ is set to 0.001. There are 5 different average delay requirements ($k = 5$) from 1 to 5 milliseconds (ms). Unless otherwise specified, we will adopt these default parameter settings in our simulation.

The request rate model: the request rate of a web portal within a day usually starts to rise at around 9:00 am, reaches a peak at around 12:00 pm and levels off after 6:00 pm. This request arrival pattern is similar to a lognormal process [8], which is referred to as *the lognormal request arrival pattern*, as shown in Fig. 3(a). On the other hand, due to some unexpected social events, a web portal may experience "burst" requests at a certain time period, and it is very likely that such burst requests exceed the processing capability of all data centers. We refer to this type of request arrival pattern as *the uniform request arrival pattern*, illustrated in Fig. 3(a). We will evaluate the performance of the proposed algorithm based on both types of request arrival patterns. In our simulation, the arrival rate curves of requests from M web portals are shifted according to their time zones.

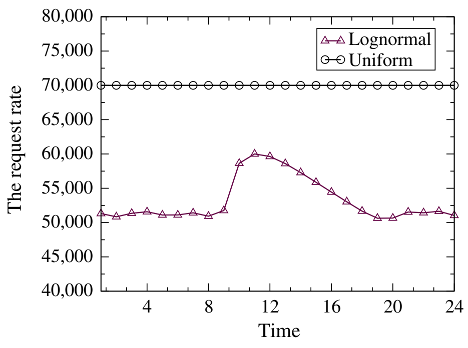
Electricity prices for data centers: following existing studies [38,39], the average electricity price per hour at each data center is derived from the raw data that is available from US government agencies [13,42], and the prices at

Table 1
Data centers.

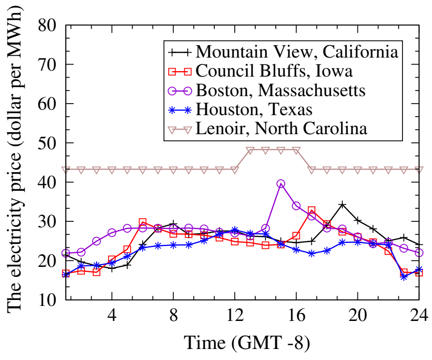
Data centers	Location	Number of machines	Service rate (req/second)	Operating power (W)
DC_1	Mountain View, CA	10,000	2.15	350
DC_2	Council Bluffs, IA	15,000	2.75	350
DC_3	Boston, MA	20,000	3	350
DC_4	Houston, TX	25,000	3.25	350
DC_5	Lenoir, NC	25,000	2.75	350

Table 2
Web portals.

Web portals	Location	Maximum request rate	Bandwidth cost (\$/req h)
WP_1	Seattle	99,200	0.05
WP_2	San Francisco	62,000	0.04
WP_3	Dallas	33,480	0.03
WP_4	Chicago	27,900	0.04
WP_5	New Mexico	27,900	0.06
WP_6	Denver	33,480	0.05



(a) Two types of request arrival patterns at each web portal



(b) Electricity prices of different data centers in Table 1

Fig. 3. Request arrival patterns and electricity prices.

all data centers are shifted according to their time zones. Fig. 3(b) depicts the electricity price curves of the five data centers listed in Table 1.

To evaluate the performance of the proposed algorithm against the optimal system throughput and the operational cost achieving the optimal system throughput, we consider two extreme cases of the problem: Case (i) the number of requests from all web portals is beyond the processing capability of all data centers; or Case (ii) the number of requests from all web portals is within the processing capability of all data centers. Recall that our objective is to maximize the system throughput, while the operational cost of realizing the achieved system throughput. For case (i), we focus on maximizing the system throughput by switching all servers on. Although switching some servers off sometimes may reduce the operational cost, the system throughput will drop, too. We thus formulate the system throughput maximization problem as `MILP(1)`, whose solution is *an upper bound* on the optimal system throughput of the operational cost minimization problem. For case (ii), since all requests can be processed by the data centers, the minimum operational cost is achieved by switching some servers off in some data centers. We thus formulate this cost minimization problem as `MILP(2)`, whose solution is *a lower bound* on the optimal operational cost of the problem of concern as it does not take the fairness of request rate allocation into consideration.

$$
\begin{aligned}
\texttt{MILP(1)}:\quad \max\ & \sum_{j=1}^{M}\sum_{i=1}^{N}\sum_{l=1}^{k} r_{j,i}^{(l)} \\
\text{s.t.}\ & \sum_{i=1}^{N} r_{j,i}^{(l)} \leqslant r_j^{(l)}, \quad \forall j,\ 1\leqslant j\leqslant M, \\
& \forall i,\ 1\leqslant i\leqslant N \text{ and } \forall l,\ 1\leqslant l\leqslant k \\
& \sum_{j=1}^{M} r_{j,i}^{(l)} \leqslant \mu_i n_i^{(l)} - \tfrac{1}{D_l}, \quad \forall i,\ 1\leqslant i\leqslant N \text{ and } \forall l, \\
& 1\leqslant l\leqslant k \\
& \sum_{l=1}^{k} n_i^{(l)} = N_i, \quad \forall i,\ 1\leqslant i\leqslant N \\
& n_i^{(l)} \in \mathcal{N}, \quad \forall i,\ 1\leqslant i\leqslant N \text{ and } \forall l,\ 1\leqslant l\leqslant k.
\end{aligned}
$$

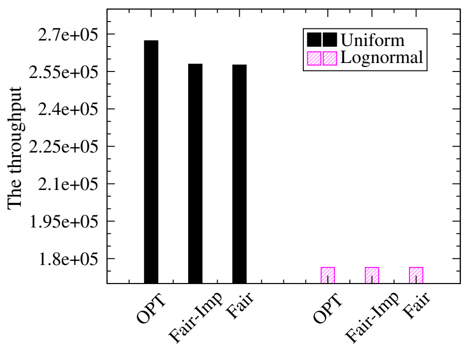
$$
\begin{aligned}
\texttt{MILP(2)}:\quad \min\ & \sum_{i=1}^{N} p_i \sum_{j=1}^{M}\sum_{l=1}^{k} \alpha_i^{(l)} r_{j,i}^{(l)} + \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{l=1}^{k} p_j^b r_{j,i}^{(l)} \\
\text{s.t.}\ & \sum_{i=1}^{N} r_{j,i}^{(l)} = r_j^{(l)}, \quad \forall j,\ 1\leqslant j\leqslant M, \\
& \text{and } \forall l,\ 1\leqslant l\leqslant k \\
& \sum_{j=1}^{M} r_{j,i}^{(l)} \leqslant \mu_i n_i^{(l)} - \tfrac{1}{D_l}, \\
& \forall i,\ 1\leqslant i\leqslant N \text{ and } \forall l,\ 1\leqslant l\leqslant k \\
& \sum_{l=1}^{k} n_i^{(l)} \leqslant N_i, \quad \forall i,\ 1\leqslant i\leqslant N \\
& n_i^{(l)} \in \mathcal{N}, \quad \forall i,\ 1\leqslant i\leqslant N \text{ and } \forall\ l, \\
& 1\leqslant l\leqslant k.
\end{aligned}
$$

Notice that cases (i) and (ii) correspond to the uniform and lognormal request arrival patterns, respectively. In our simulation, we solve `MILP(1)` and `MILP(2)` using lp_solve [31]. For the sake of convenience, in the rest of discussions we use **OPT** to represent these two optimal solution estimations if no confusion arises. To facilitate our evaluation, we consider a relaxation of `MILP(1)` and `MILP(2)` where the number of servers needed to process requests with delay requirement $D_l, n_i^{(l)}$, is a real value. It must be emphasized that such an estimation on the optimal system throughput is very conservative, as the optimal system throughput can be much lower than this estimated value.

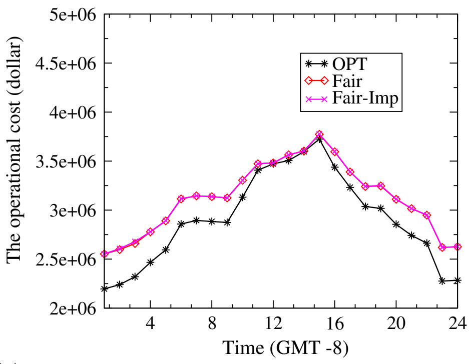
5.2. Algorithm performance

We first investigate the performance of the proposed approximation algorithm and its improved variant including the system throughput, the operational cost, and the number of servers switched on to achieve the system throughput. For the sake of convenience, we refer to the proposed approximation algorithm and its improved variant with a threshold ξ as algorithm `Fair` and algorithm `Fair-Imp`, respectively.

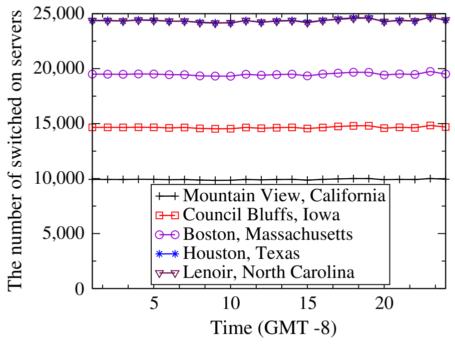
Fig. 4(a) shows that the system throughput of algorithms `Fair-Imp` and `Fair` are 257,913 and 257,582 respectively, in comparison with the optimal one,



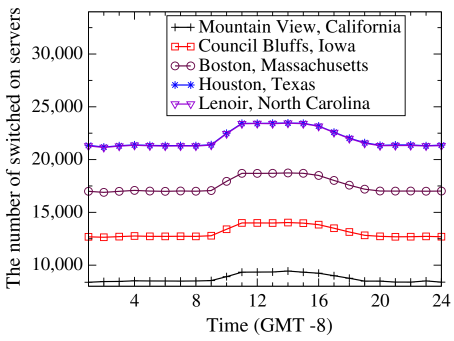
(a) The system throughput



(b) The operational cost under the lognormal request arrival pattern



(c) The number of switched on servers of algorithm `Fair` under the uniform request arrival pattern



(d) The number of switched on servers of algorithm `Fair` under the lognormal request arrival pattern

Fig. 4. The performance of algorithm `Fair` and algorithm `Fair-Imp`.

267,295, under the uniform request arrival pattern. The average approximation ratio of algorithm `Fair` is 0.963 which is much greater than its analytical counterpart $(1/2 - \epsilon')$. Fig. 4(b) plots the curves of the operational costs that achieve the corresponding system throughput by algorithms `Fair`, `Fair-Imp` and `MILP(2)`, respectively, from which it can be seen that the operational costs are much higher than that of the optimal operational cost, since the operational cost by algorithm `MILP(2)` is a lower bound on the optimal operational cost of the problem of concern. It also can be seen that the operational cost by algorithm `Fair-Imp` is marginally better than that by algorithm `Fair`. This is because the only difference between them is the choice of an appropriate ρ. Thus, in the rest of discussions we focus only on the performance of algorithm `Fair`.

Fig. 4(c) and (d) plot the number of servers switched on during a 24-h period to achieve the system throughput in Figs. 4(a), by algorithm `Fair`. Specifically. Fig. 4(c) shows that under the uniform request arrival pattern, almost all servers are switched on as the total number of requests from all web portals is larger than the aggregate processing capability of all data centers. For example, the average number of switching on servers at Lenoir is around 9960 which is very close to its capacity 10,000. The reason why not all servers are switched on is to meet the fairness on request rate allocations among different web portals. Specifically, to achieve the fairness, the proposed algorithm first conservatively scales down every incoming request flow to every data center including those having enough servers to process all admitted requests, by a factor ρ_{min}. If we relax the fairness requirement or increase the accuracy of the threshold of ρ_{min}, the gap can be further reduced. It must be mentioned that these small number of non-switching on servers (40 out of 10,000) are acceptable by cloud service providers, since keeping a small number of servers unallocated to requests is usually helpful in satisfying peak demands of already allocated requests. Fig. 4(d) demonstrates the number of switched-on servers under the lognormal request arrival pattern.

We then evaluate the impact of electricity price diversity on the number of switched-on servers by algorithm `Fair`. We observe in Fig. 4(c) and (d) that the number of switching on servers in each data center is mainly affected by the request arrival patterns. Such influence offsets the impacts of electricity prices. To show the impacts of electricity prices, we keep the request arrival rate unchanged during a 24-h period and reduce the request arrival rate to the half of uniform request arrival pattern in Fig. 3(a). Fig. 5 clearly indicates that the number of switching on servers at Lenoir drops at 1:00 pm with the growth of the electricity price at that time slot. Similarly, the number of switching on servers at Mountain View reaches its bottom at 7:00 pm since the electricity price reaches its peak at that time.

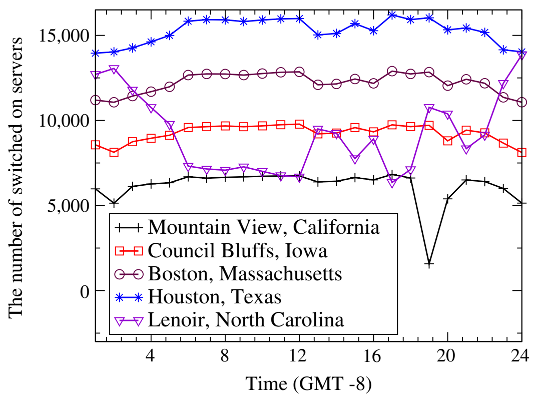


Fig. 5. The number of switching on servers with half numbers of requests of the uniform request arrival pattern.

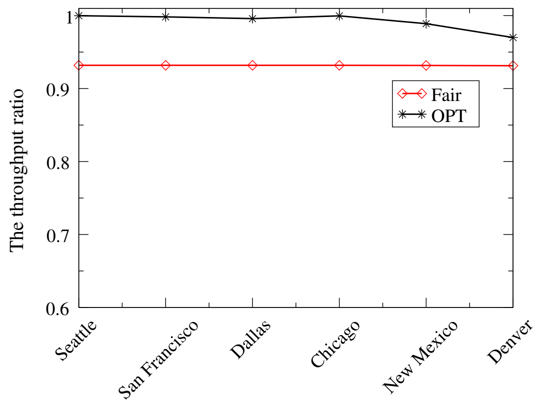


Fig. 6. The percentage of number of requests from each web portal is allocated.

We finally evaluate the fairness of request rate allocation in the solution delivered by algorithm Fair. Notice that all requests under the lognormal request arrival pattern can be processed immediately as the aggregate processing capability of all data centers exceeds the accumulative load of all requests from all web portals. Thus, we here only consider the uniform request arrival pattern. Fig. 6 plots the request rate allocation of requests from different web portals, from which it can be confirmed that the solution delivered by algorithm Fair does allocate the requests from different web portals fairly, whereas the OPT algorithm does not consider the fairness at all.

5.3. Impacts of other parameters on algorithm performance

In this subsection we investigate the impacts of other parameters on algorithm performance. We start by studying the impact of parameter k on the system throughput by algorithm Fair, by varying its value from 2 to 16 while assuming that the range of k-level delays is between 1 ms and 5 ms. Fig. 7 implies that the average system throughput during a 24-h period decreases with the growth of k. For the system throughput, each edge in $G_{f,k}^{(l)}$ from DC_i to t_0 is the bottleneck capacity edge whose capacity, $\lfloor \mu_i N_i - k/D_l \rfloor$, will determine the amount of flow entering into data center DC_i. With the growth of k, the edge capacity decreases, so does the system throughput. In terms of the running time of algorithm Fair, it takes k iterations with each for a different edge capacity assignment, i.e., $\lfloor \mu_i N_i - k/D_l \rfloor$ for each edge from DC_i to t_0 for all l with $1 \leqslant l \leqslant k$. With the increase of k, the number of iterations increases, which leads to a longer running time of algorithm Fair. Table 3 lists the average running time of algorithm Fair and algorithm MILP(1) for a 24-h period, from which it can be seen that with the growth of k, the running time of MILP(1) becomes prohibitively high, e.g., it takes around 20 min when $k = 3$ while the solution is not achievable when $k \geqslant 4$. In contrast, algorithm Fair only takes several seconds even when $k \geqslant 10$, which exhibits high scalability.

We then investigate the impact of request distributions of different delays on the operational cost by considering three request distributions: (1) the uniform distribution where the number of requests with different delays is identical; (2) the random distribution where the number of requests with different delays is randomly generated; and (3) the geometric distribution where $r_j^{(l)} = 2r_j^{(l+1)}$ for each web portal $WP_j, 1 \leqslant j \leqslant M$.

Fig. 8 illustrates the impact of request distributions on the operational cost under both uniform and lognormal

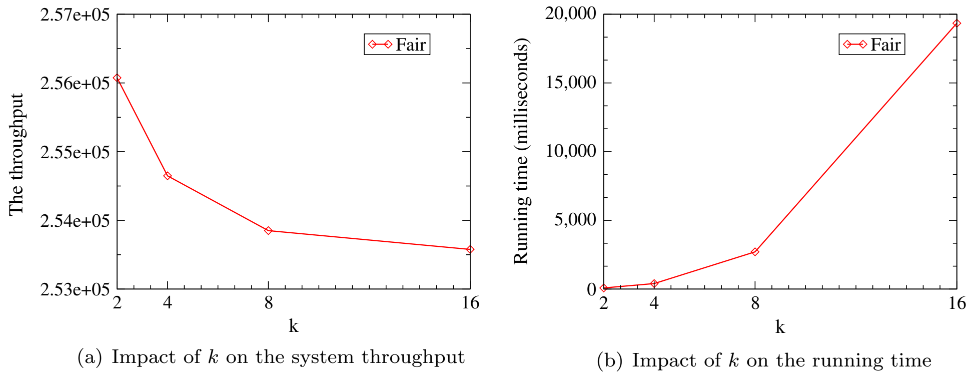
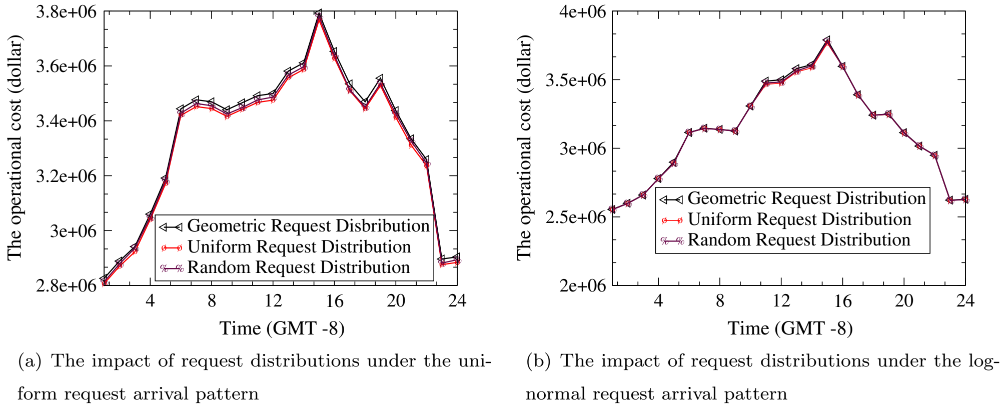


Fig. 7. The impact of k on the system throughput and the running time of algorithm Fair.

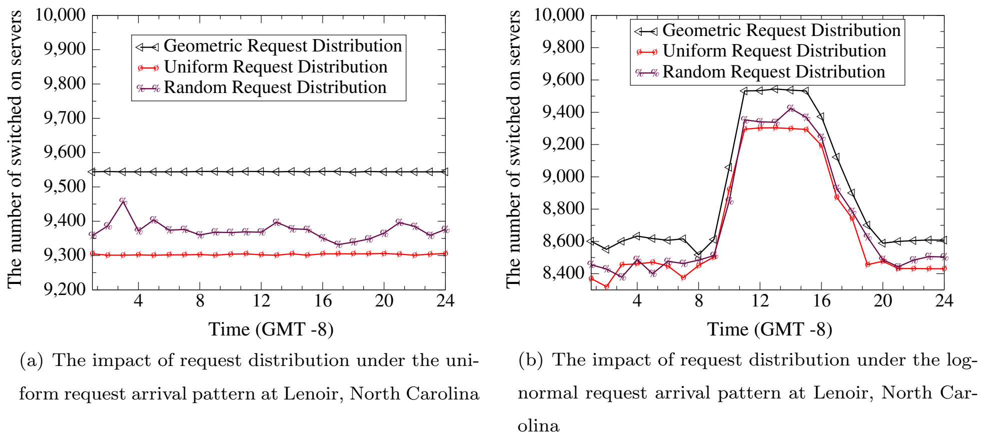
Table 3
The average running time (ms) during a 24-h period of algorithms `Fair` and `MILP(1)` in the cloud computing environment of five data centers and six web portals.

Algorithm	$k = 2$	$k = 3$	$k = 4$	$k = 8$	$k = 16$
`Fair`	80.8	172.6	399.9	2707.1	19345.1
`MILP(1)`	33,836	1,254,200	–	–	–



(a) The impact of request distributions under the uniform request arrival pattern
(b) The impact of request distributions under the log-normal request arrival pattern

Fig. 8. The impact of request distribution on the operational cost of algorithm `Fair`.



(a) The impact of request distribution under the uniform request arrival pattern at Lenoir, North Carolina
(b) The impact of request distribution under the log-normal request arrival pattern at Lenoir, North Carolina

Fig. 9. The impact of request distribution on the number of active servers of algorithm `Fair`.

request arrival patterns. Fig. 8(a) implies that the operational cost of the geometric request distribution is much higher than those of uniform and random ones since the number of requests in the geometric request distribution with lower delays are much larger than the one with higher delays, while these lower delay requests normally consume much more server resources that leads to a much higher operational cost. Although this trend is not obvious under the lognormal request arrival pattern as shown in Fig. 9(b), the operational cost of the geometric request distribution is still greater than those of the other two request distributions.

Fig. 9 illustrates the number of servers switching on at Lenoir, North Carolina under both uniform and lognormal request arrival patterns. Other data centers have the similar behaviors as the one at Lenoir, so we here only give the one at Lenoir, from which it can be seen that the number of switched on servers of the geometric request distribution is higher than the uniform and random request distributions due to types of requests with lower delay requirements need more servers to be switched on.

5.4. Scalability of the proposed algorithm

We finally investigate the scalability of algorithm `Fair` in a large-scale distributed cloud computing environment consisting of 10–20 data centers and 8–18 web portals, which are randomly deployed in some of 48 states in the United States with at most one data center per state. We evaluate the scalability of algorithm `Fair` on the number of data centers by varying the number of data centers from 10 to 20 while keeping the number of web portals fixed at

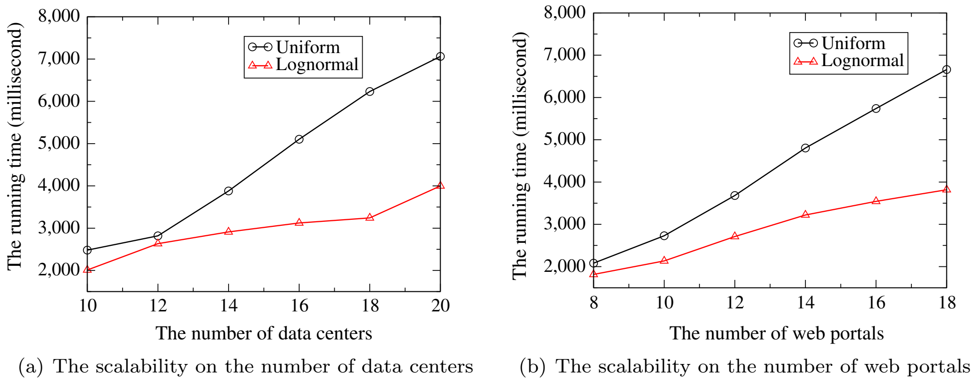


Fig. 10. The scalability of algorithm `Fair` by varying problem sizes.

6 and setting ϵ to 0.1. The electricity price of each data center is randomly selected from a prior given set of electricity prices and the time zone of each data center is randomly assigned in the range from GMT-8 to GMT-5. We also evaluate the scalability of algorithm `Fair` on the number of web portals by varying the number of web portals from 8 to 18 while keeping the number of data centers fixed at 10. The request rates under both lognormal and uniform request arrival patterns are properly scaled according to the total processing rate of data centers. The results are shown in Fig. 10, from which it can be seen that the running time of algorithm `Fair` increases slightly with the increase of problem size. Fig. 10(a) illustrates that algorithm `Fair` takes at most 4 s and 7 s to perform fair request rate allocations for both uniform and lognormal request arrival patterns in a 24-h period.

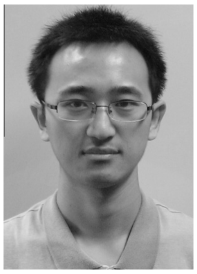
6. Conclusion

In this paper, we studied the operational cost minimization problem in a distributed cloud computing environment that not only provides fair rate allocations among web portals but also meets multi-level user SLA requirements, by exploiting time-varying electricity prices and user request rates, for which we first proposed an adaptive operational cost optimization framework. We then devised a fast, scalable approximation algorithm with a provable approximation ratio for the problem. We finally conducted extensive experiments by simulations to evaluate the performance of the proposed algorithm, using real-life electricity price data traces. Experimental results demonstrate that the proposed algorithm is very promising, and the solution obtained is fractional of the optimum.

References

[1] M.A. Adnan, R. Sugihara, R. Gupta, Energy efficient geographical load balancing via dynamic deferral of workload, in: Proc. of CLOUD'12, IEEE, 2012.
[2] M.A. Adnan, R. Sugihara, Y. Ma, R. Gupta, Energy-optimized dynamic deferral of workload for capacity provisioning in data centers, in: Proc. of IGCC'13, IEEE, 2013.
[3] F. Ahmad, T.N. Vijaykumar, Joint optimization of idle and cooling power in data centers while maintaining response time, in: Proc. of ASPLOS'10, ACM, ACM, 2010.
[4] Amazon EC2. <http://aws.amazon.com/ec2>.
[5] Amazon Data Center Size. <http://huanliu.wordpress.com/2012/03/13/amazon-data-center-size/>.
[6] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, A view of cloud computing, Commun. ACM 53 (2010) 50–58.
[7] L.A. Barroso, U. Hölzle, The case for energy-proportional computing, IEEE Comput. 40 (2007) 33–37.
[8] T. Benson, A. Anand, A. Akella, M. Zhang, Understanding data center traffic characteristics, ACM SIGCOMM Comput. Commun. Rev. 40 (2010) 92–99.
[9] N. Buchbinder, N. Jain, I. Menache, Online job-migration for reducing the electricity bill in the cloud, in: Proc. of IFIP NETWORKING'11, LNCS, vol. 6640, 2011, pp. 172–185.
[10] C. Chen, B. He, X. Tang, Green-aware workload scheduling in geographically distributed data centers, in: Proc. of CloudCom'12, IEEE, 2012.
[11] Cloud Load Balancer Pricing. <http://www.rackspace.com.au/cloud/load-balancers/pricing>.
[12] E. Elnozahy, M. Kistler, R. Rajamony, Energy-efficient server clusters, in: Proc. of PACS'02, LNCS, vol. 2325, 2003.
[13] Federal Energy Regulatory Commission. <http://www.ferc.gov/>.
[14] A. Gandhi, M. Harchol-Balter, R. Das, C. Lefurgy, Optimal power allocation in server farms, in: Proc. of SIGMETRICS'09, ACM, 2009.
[15] A. Gandhi, V. Gupta, M. Harchol-Balter, A. Kozuch, Optimality analysis of energy-performance trade-off for server farm management, J. Perform. Eval. 67 (2010) 1155–1171. Elsevier.
[16] Y. Gao, Z. Zeng, X. Liu, P.R. Kumar, The answer is blowing in the wind: analysis of powering Internet data centers with wind energy, in: Proc. of INFOCOM'13, IEEE, 2013.
[17] N. Garg, J. Könemann, Faster and simpler algorithms for multicommodity flow and other fractional packing problems, in: Proc. of FOCS'98, IEEE, 1998.
[18] Í. Goiri, K. Le, T.D. Nguyen, J. Guitart, J. Torres, R. Bianchini, GreenHadoop: leveraging green energy in data-processing frameworks, in: Proc. of EuroSys'12, ACM, 2012.
[19] Google Data Centers. <http://www.google.com/about/datacenters/>.
[20] Google App Engine Pricing. <https://cloud.google.com/pricing/>.
[21] D. Gross, J.F. Shortle, J.M. Thompson, C.M. Harris, Fundamentals of Queueing Theory, Wiley Press, 2008.
[22] Y. Guo, Z. Ding, Y. Fang, D. Wu, Cutting down electricity cost in Internet data centers by using energy storage, in: Proc. of GLOBECOM'11, IEEE, 2011.
[23] J. Kaplan, W. Forrest, N. Kindler, Revolutionizing Data Centre Energy Efficiency, McKinsey, Technique Report, 2008.
[24] D. Kliazovich, P. Bouvry, S.U. Khan, DENS: data center energy-efficient network-aware scheduling, Clust. Comput. 16 (1) (2013) 65–75. Springer.
[25] K. Le, R. Bianchini, M. Martonosi, T.D. Nguyen, Cost- and energy-aware load distribution across data centers, in: Proc. of HOTPOWER'09, ACM, 2009.
[26] K. Le, R. Bianchini, T.D. Nguyen, O. Bilgir, M. Martonosi, Capping the brown energy consumption of internet services at low cost, in: Proc. of IGCC'10, IEEE, 2010.
[27] H. Liu, Y. Chen, C. Bash, A. Wierman, D. Gmach, Z. Wang, M. Marwah, C. Hyser, Renewable and cooling aware workload management for sustainable data centers, in: Proc. of SIGMETRICS'12, ACM, 2012.

[28] Z. Liu, M. Lin, A. Wierman, S. Low, L.L.H. Andrew, Geographical load balancing with renewables, ACM SIGMETRICS Perform. Eval. Rev. 39 (2011) 62–66.
[29] Z. Liu, M. Lin, A. Wierman, S. Low, L.L.H. Andrew, Greening geographical load balancing, in: Proc. of SIGMETRICS'11, ACM, 2011.
[30] M. Lin, A. Wierman, L. Andrew, E. Thereska, Dynamic right-sizing for power-proportional data centers, in: Proc. of INFOCOM'11, IEEE, 2011.
[31] lp_solve. <http://lpsolve.sourceforge.net/>.
[32] T. Lu, M. Chen, L.L.H. Andrew, Simple and effective dynamic provisioning for power proportional data centers, IEEE Trans. Parall. Distrib. Syst. 24 (6) (2013). IEEE.
[33] Microsoft Cloud Power. <http://www.microsoft.com/en-au/cloud/default.aspx>.
[34] D. Niu, C. Feng, B. Li, Pricing cloud bandwidth reservations under demand uncertainty, in: Proc. of SIGMETRICS'12, ACM, 2012.
[35] H. Qian, D. Medhi, Server operational cost optimization for cloud computing service providers over a time horizon, in: Proc. of HOT-ICE'11, ACM, 2011.
[36] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, B. Maggs, Cutting the electric bill for Internet-scale systems, in: Proc. of SIGCOMM'09, ACM, 2009.
[37] L. Rao, X. Liu, M.D. Ilic, J. Liu, Distributed coordination of Internet data centers under multiregional electricity markets, Proc. IEEE 100 (2011) 269–282.
[38] L. Rao, X. Liu, L. Xie, W. Liu, Minimizing electricity cost: optimization of distributed Internet data centers in a multi-electricity-market environment, in: Proc. of INFOCOM'10, IEEE, 2010.
[39] L. Rao, X. Liu, L. Xie, W. Liu, Coordinated energy cost management of distributed Internet data centers in smart grid, IEEE Trans. Smart Grid 3 (2012) 50–58.
[40] S. Ren, Y. He, F. Xu, Provably-efficient job scheduling for energy and fairness in geographically distributed data centers, in: Proc. of ICDCS'12, IEEE, 2012.
[41] N. Tziritas, S.U. Khan, C.Z. Xu, T. Loukopoulos, S. Lalis, On minimizing the resource consumption of cloud applications using process migrations, J. Parall. Distrib. Comput. 73 (12) (2013) 1690–1704. Elsevier.
[42] U.S. Energy Information Administration (EIA). <http://www.eia.doe.gov/>.
[43] Z. Xu, W. Liang, Minimizing the operational cost of data centers via geographical electricity price diversity, in: Proc. of CLOUD'13, IEEE, 2013.
[44] Y. Zhang, Y. Wang, X. Wang, GreenWare: greening cloud-scale data centers to maximize the use of renewable energy, in: Proc. of MIDDLEWARE'11, ACM, 2011.
[45] Y. Zhang, Y. Wang, X. Wang, Electricity bill capping for cloud-scale data centers that impact the power markets, in: Proc. of ICPP'12, September, 2012.
[46] Z. Zhang, M. Zhang, A. Greenberg, Y.C. Hu, R. Mahajan, B. Christian, Optimizing cost and performance in online service provider networks, in: Proc. of NSDI'10, ACM, 2010.



Zichuan Xu received his ME degree and BSc degree from Dalian University of Technology in China in 2011 and 2008, both in Computer Science. He is currently pursuing his PhD study in the Research School of Computer Science at the Australian National University. His research interests include cloud computing, wireless sensor networks, routing protocol design for wireless networks, algorithmic game theory, and optimization problems.

Weifa Liang received the PhD degree from the Australian National University in 1998, the ME degree from the University of Science and Technology of China in 1989, and the BSc degree from Wuhan University, China in 1984, all in computer science. He is currently an Associate Professor in the Research School of Computer Science at the Australian National University. His research interests include design and analysis of routing protocols for wireless ad hoc and sensor networks, cloud computing, graph databases, design and analysis of parallel and distributed algorithms, combinatorial optimization, and graph theory. He is a senior member of the IEEE.