Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 920829, 13 pages
http://dx.doi.org/10.1155/2014/920829

Research Article

Dynamic Request Routing for Online Video-on-Demand Service: A Markov Decision Process Approach

Jianxiong Wan,[1] Limin Liu,[1] and Jianwei Guo[2]

[1] *Inner Mongolia University of Technology, Hohhot 010051, China*
[2] *Liaoning Technical University, Fuxin 123000, China*

Correspondence should be addressed to Jianxiong Wan; jxwan@csnet1.cs.tsinghua.edu.cn

Received 22 January 2014; Accepted 4 May 2014; Published 1 June 2014

Academic Editor: Guoqiang Hu

We investigate the request routing problem in the CDN-based Video-on-Demand system. We model the system as a controlled queueing system including a dispatcher and several edge servers. The system is formulated as a Markov decision process (MDP). Since the MDP formulation suffers from the so-called "the curse of dimensionality" problem, we then develop a greedy heuristic algorithm, which is simple and can be implemented online, to approximately solve the MDP model. However, we do not know how far it deviates from the optimal solution. To address this problem, we further aggregate the state space of the original MDP model and use the bounded-parameter MDP (BMDP) to reformulate the system. This allows us to obtain a suboptimal solution with a known performance bound. The effectiveness of two approaches is evaluated in a simulation study.

1. Introduction

The advancements of Internet technology have remarkably improved the capacity of both core networks and access networks and enable complex bandwidth-consuming applications which are far beyond simple text-based web page browsing. Among all these applications, Video-on-Demand (VoD) service has gained great popularity over the past decade. According to the report of the world's largest content delivery network (CDN) provider, Akamai, the global average connection speed experienced over 20% growth during 2010 [1], partially due to the surging video content transferring via the Internet.

Nowadays, commercial VoD services on the Internet like YouTube [2] and Hulu [3] are typically supported by CDNs. In RFC 3466, Internet Engineering Task Force (IETF) defined CDN as a type of content network which is a virtual network on top of a generic IP network. CDNs contain many infrastructures (edge servers) distributed across vast geographical areas. Clients can fetch the video streaming data from edge servers instead of the source server, avoiding the network congestion and source server overloaded efficiency.

Modern CDN providers often build their system over cloud infrastructures to reduce the deployment and maintenance cost while extending the system scalability. From this sense, Lenk et al. [4] categorized Amazon's CDN service CloudFront and Akamai's EdgePlatform as higher infrastructure services. There is vast number of literatures concentrated on cloud-based CDN system. The recent AirLift system [5] and the CALMS system [6] demonstrated the feasibility of deploying video streaming applications in cloud-based CDN. Li et al. [7] regarded network congestion at cloud egress and latency to client as one of the major challenges for the current cloud system. They proposed an architecture which integrates CDN with cloud where global load balancing can be achieved with CDN. An experiment in the Microsoft Windows Azure public cloud demonstrated the effectiveness of their idea. Tran et al. [8] developed a content distribution network cloud architecture (CDNCA) based on quality of service (QoS) and quality of experience (QoE) criterions. Jin et al. [9] proposed a content-delivery-as-a-service (CoDaaS) framework to enable on-demand virtual content delivery service (vCDS) for user-generated content (UGC). Chia-Feng et al. [10] presented a cloud-based CDN (CCDN) platform which provides content

delivery features with cloud storage and platform as a service in cloud computing field. Chen et al. [11] investigated a joint problem of building distribution paths and placing Web server replicas in CDNs driven by storage cloud to minimize the cost. Clearly, the techniques to migrate CDN service from traditional distributed computing systems to the cloud are an active research area.

There are also many research efforts in industry focused on this area. A number of major enterprises develop their own cloud-based CDN systems. For example, Amazon CloudFront [12] is a web service for content delivery. It uses Amazon S3 (a cloud storage service) or EC2 (an infrastructure-as-a-service) as the underlying storage servers and edge servers. Akamai NetStorage [13] is another example of cloud storage service built for Akamai's CDN. It is a geographically distributed cloud storage system which provides multiple terabytes of storage capacity. Limelight Networks [14] also builds many data centers by itself all over the world to support its CDN service.

One of the key challenges for CDN providers is to design an appropriate request routing strategy so that CDN providers can obtain as much profit as possible while preserving good performance. In this paper, we investigate the request routing problem in the CDN-based VoD system. The system of our interest is shown in Figure 1. It contains a dispatcher and a number of edge servers. Like Amazon CloudFront, our system can be implemented on a cloud infrastructure. CDN providers can rent virtual machines (VMs) from geographically distributed cloud providers or build their own distributed data centers (DCs). These VMs or DCs act as edge servers which directly transfer VoD streaming data to the clients. The load balance component from cloud system and the geo-aware request routing component from CDN system are integrated into the dispatcher. We assume the dispatcher works in a time-slotted fashion.

Request routing problem is one of the most fundamental issues in the distributed online service providing system. It has received much attention in various environments over the years [15–18]. Among these works, some focused on probing for the design principles behind proprietary request routing algorithm of commercial enterprise and evaluating their effectiveness by using a measurement methodology [19, 20] and others preferred to treat the problem in a more analytical way [21–24]. Xu and Li [25, 26] used decomposition techniques in optimization theory to investigate the request routing problem in various settings. Tran et al. [27] developed a request routing scheme based on the QoE criterion. Qian and Rabinovich [28] developed a heuristic algorithm called permutation prefix clustering to solve the joint application placement and request routing problem. Dealer system [29] abstracts multilayer application into a directed graph where the vertices are application components and the edges are communications between vertices. Dealer tries to minimize response time by searching for a better combination approach of application components. CloudGPS system [30] and NetPaaS system [31] deal with the request routing problem in ISP-P2P and ISP-CDN environments. The highlight of the distributed cooperative request routing algorithms behind these systems is that they can benefit all entities (CDN, ISP, and P2P end user) within the system. In our work, we use the Markov decision process (MDP) [32] rather than static optimization techniques [33, 34] to model the system, since the dynamic nature of MDP can potentially yield remarkable performance improvements [21].

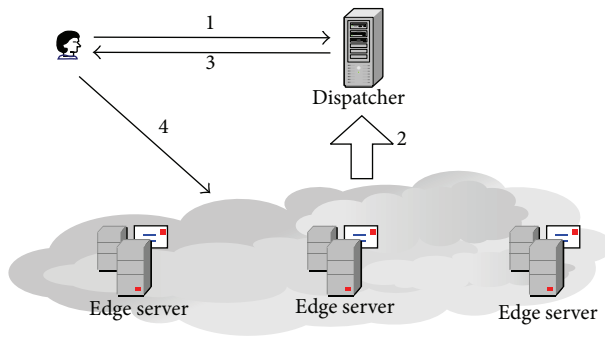


Figure 1: A typical CDN-based VoD service system. The request routing task is fulfilled by the dispatcher. The request routing contains 4 steps: (1) client requests arrive at the dispatcher; (2) the dispatcher samples the state of the servers and computes the request routing strategy; (3) the dispatcher returns the destination edge server information to the clients; (4) client requests are redirected to appropriate edge servers.

Load balance factors must be considered while designing a request routing strategy; it has been intensively studied by the researchers [35–40]. An unbalanced load allocation may cause network congestion and deplete server computational resources and thus greatly degrade user experience. However, starting from the perspective of load balance alone is not enough, since the final goal of the VoD provider is to earn as much profit as possible, rather than only to guarantee the user experience. In a practical CDN-based VoD service system, local costs in different geographical regions, for example, the bandwidth rental cost and the server maintenance cost, are not equal. How to select appropriate server for a request so that the profit is maximized is our focus.

There are many researches concentrated on reward-based dynamic request routing in web-based applications [22–24]. Unfortunately, neither of them can be directly applied to the VoD service for following reasons: (1) a video request will take up the resources of VoD servers throughout the whole viewing period, which is several orders of magnitude longer than a web-based request; (2) as long as a client begins to receive data from an edge server, it cannot be redirected to other edge servers arbitrarily, since this will disturb the continuity of the playback process and degrade the user experience.

In this paper, we investigate the request routing problem in the VoD service by formulating it as a standard MDP, which can be theoretically solved by the classical algorithms like value iteration, policy iteration, or their variations. However, this MDP formulation is practically intractable due to the so-called the curse of dimensionality; that is, the problem size grows exponentially with the underling parameters. To address this issue, we propose two alternative approaches to approximately solve the problem. The first approach is to

use a greedy strategy, that is, to focus on maximizing the reward which can be earned in current time slot, instead of the accumulated reward in the long run. The advantage of the greedy approach is its simplicity; for example, it can be implemented online. The drawback of this approach is that the performance deviation between the optimal strategy and the greedy strategy is unknown. To overcome it, we turn to the second approach—bounded-parameter Markov decision processes (BMDP). The BMDP was proposed by Givan et al. [41, 42] to approximately solve the MDP with a large state space. We use state aggregation technique to partition the overall state space and reformulate the request routing problem as a BMDP. The BMDP formulation has two advantages. (1) we can derive a B-optimal strategy aiming at maximizing the attainable reward in a BMDP model; (2) the upper bound value of the B-optimal strategy is the highest reward a feasible strategy can achieve in the original MDP model, so it can be served as a cornerstone to evaluate other strategies, for example, the greedy strategy. We further conduct experimental study to evaluate the effectiveness of these two algorithms. Simulation results show that the B-optimal strategy is a close-to-optimal request routing solution.

This paper proceeds as follows. Section 2 depicts the model in detail; Section 3 presents the MDP formulation of the request routing problem; in Section 4 we first provide a greedy algorithm to approximately solve the MDP and then reformulate the problem as a BMDP. The interval value iteration (IVI) algorithm is presented thereafter, and a comparison of the IVI algorithm with the traditional value iteration algorithm from the perspective of computational complexity is conducted; a numerical analysis is given in Section 5 followed by conclusions in Section 6.

2. System Model

We use a discrete time controlled queueing model to describe the VoD service system. The model is shown in Figure 2. It consisted of a dispatcher and several edge servers located across a geographic region (e.g., a country). The dispatcher is a centralized controller which collects video requests and redirects them to appropriate edge servers. At a given time slot, the VoD edge servers transmit the video streaming data to each of its clients. At the end of a time slot, some clients close their streaming sessions and leave the system, while others prefer to stay and wait for receiving video stream in the next time slot. Upon each accepted request, the CDN provider can get rewards. Our focus is to find the optimal strategy which maximizes such rewards. Notations are summarized in Notation section.

2.1. Customer Model. Customer categorization is a common approach for studying the online service providing system. In this paper, we regard the partition by the video file that the clients are asking for as impractical since there may be hundreds of thousands of video files in a VoD system. As explained in the following section, the increase of customer types will dramatically raise the state space and action space of the problem, making it hard to solve. We use an alternative

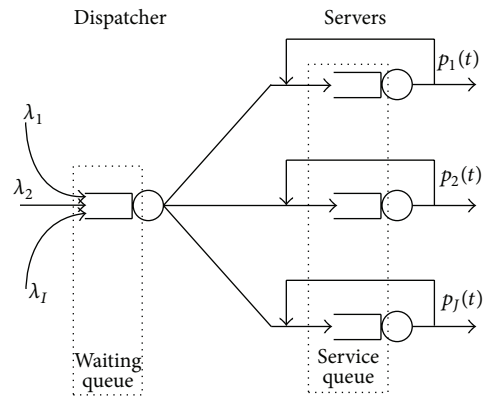


Figure 2: System model for a CDN-based VoD service.

approach, that is, categorizing the customer according to (1) the expected viewing time length and (2) the average bitrates. The first standard ensures that all requests in a given time have the same departure probability (as we can see later), and the second one implies that they consume the same amount of resources. This categorization can be achieved by the following steps.

(i) Measure the expected viewing time for each video file.

(ii) Group video files which have (approximately) the same expected viewing time and bitrates.

(iii) If a customer asks for a video file which belongs to the ith group, mark this request as type i.

The usage of average bitrate is not an exact model since VoD providers sometimes employ a variable bit rate (VBR) approach like MPEG-4 to encode video files for reduction of the file size. This may induce the system to run out of resources if all requests achieve their peak rate at the same time. However, we believe that this impreciseness does not affect our overall model that much for two reasons: (1) the video streaming itself can tolerate packet losses or delay jitters to some extent and (2) the alignment of the peak rate periods of the video streams is rare [43]; we can use some moderate resource reservation strategies to effectively avoid the resource depletion in most cases.

In each time slot, the number of type i video requests arriving at the dispatcher, that is, λ_i, is a random variable, which can be reasonably viewed as subject to some stationary probability distribution in VoD service. To avoid infinite state space problem, we assume that λ_i is a bounded nonnegative integer random variable.

2.2. Server Model. Each VoD edge server is modeled by a discrete time queueing system. The resource capacity of edge server j, that is, output bandwidth, and so forth, is C_j. We assume that a type i request consumes ω_i unit of bandwidth. The resource consumption of all requests in a server cannot exceed its capacity limitation.

In the VoD service, customers' requests tend to "reside in" the system since the playback time of a video file is several orders of magnitude longer than the one in web-based

application. We assume each accepted request must stay in the system for at least one time slot, and all departures happen at the end of a time slot. Each edge server does not log the state of an individual request, that is, the time a request has spent in the system, because this scheme is too resource consuming and not scalable as the number of requests grows. Therefore, we consider the probability that a request leaves the system p_i is equal in every time slot. We have the following result.

Lemma 1. *Suppose a type i request can stay in the system for at most $\overline{T}_i$ of time or $K = \lceil \overline{T}_i/\Gamma \rceil$ of time slot; then*

$$\frac{1}{p_i} - \frac{(Kp_i + 1)(1 - p_i)^K}{p_i} = \left\lceil \frac{T_i}{\Gamma} \right\rceil, \tag{1}$$

where Γ is the slot length and T_i is the average sojourn time of type i request in the system.

Proof. The proof is simply a calculation of expected sojourn time provided that a type i request can stay for at most K slots in the system. Consider

$$\begin{aligned} &\sum_{k=1}^{K} \left\{ p_i \cdot (1 - p_i)^{k-1} \cdot k\Gamma \right\} \\ &\quad = p_i\Gamma \left\{ \sum_{k=1}^{\infty} k(1 - p_i)^{k-1} - \sum_{k=K+1}^{\infty} k(1 - p_i)^{k-1} \right\} \\ &\quad = p_i\Gamma \left\{ \frac{1}{p_i^2} - \frac{(Kp_i + 1)(1 - p_i)^K}{p_i^2} \right\} \\ &\quad = T_i, \end{aligned} \tag{2}$$

completing the proof. □

It is not easy to get the analytical solution of p_i from (1). We propose the following methods to get numerical value of p_i.

(i) When K is small, we can plot the left hand side of (1) with p_i as a variable and locate p_i directly from the figure, as illustrated in Figure 3.

(ii) When K is large, the second term in the left hand side of (1) tends to 0 and can be omitted, yielding an approximation of

$$p_i \approx \left\lceil \frac{\Gamma}{T_i} \right\rceil. \tag{3}$$

3. Problem Formulation

The above model can be formulated as a standard discrete time Markov decision process (MDP). We illustrate the system dynamics in Table 1. During $(t - 1, t)$ the dispatcher buffers the incoming requests in its waiting queue. In time point t, the dispatcher samples the system state including

(1) the arrival vector $\boldsymbol{\lambda}(t) = \{\lambda_i(t)\}$, $i \in I$, which describes the number of type i requests in the waiting queue;

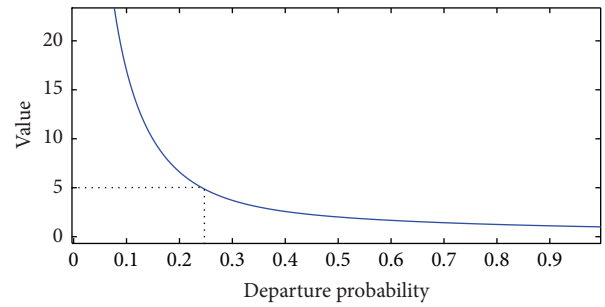


Figure 3: An example to find departure probability with a small $K = 10$. $T_i = 50$ s, $\Gamma = 10$ s. The upper bound of sojourn time is 100 s. We obtain that $p_i \approx 0.25$ in this scenario.

Table 1: System dynamics.

Time	Event
t	The dispatcher samples the system state $(\lambda(t), \mathbf{N}(t))$, and make the decision $a(t)$ forwarding the buffered requests to appropriate edge servers (or rejecting some requests).
$(t, t + 1)$	Requests arrive and are buffered in the dispatcher's waiting queue.
$(t + 1)^-$	Some requests in the service queue leave the system, and the system transits to the next state.

(2) the server load matrix $\mathbf{N}(t) = \{n_{ij}(t)\}$, $i \in I$ and $j \in J$, which describes the number of type i requests in edge server j,

and then it makes a decision. The action is a matrix $a(t) = \{a_{ij}(t)\}$ denoting the number of type i requests forwarded to edge server j. The system then proceeds to the time $(t + 1)^-$, where a reward of $R(\mathbf{N}(t), a(t))$ is received and some requests leave the system. The system transits to another state. The MDP formulation of the problem is given as follows.

States. Denote the state space by $\mathbf{S}$ and denote one element in $\mathbf{S}$ by $(\mathbf{N}, \boldsymbol{\lambda})$. We view the actual state is a function of time, that is, $(\mathbf{N}(t), \boldsymbol{\lambda}(t))$. Note that $\boldsymbol{\lambda}$ can be fully observed by the dispatcher.

Decision Epoch. Decision epoch t is at the start of a slot with length Γ; namely, $t \in \{0, \Gamma, 2\Gamma, \ldots, n\Gamma, \ldots\}$.

Actions. At the beginning of the tth slot, the dispatcher can make a deterministic action $a(t)$ subject to the following constraints:

$$\sum_{j \in J} a_{ij}(t) \le \lambda_i(t), \quad \forall i \in I \tag{4}$$

$$\sum_{i \in I} \omega_i \left(a_{ij}(t) + n_{ij}(t) \right) \le C_j, \quad \forall j \in J, \tag{5}$$

where (4) is the flow conservation constraint representing the forwarded requests which cannot exceed the arrived (and if

forwarded requests are less than the arrived, some requests must be rejected) and (5) implies the bandwidth constraints.

Transition Probability. Since the arrival vector is a stochastic vector and is independent with the server load matrix, we focus mainly on the changes of the latter one. The following dynamic equation can be obtained immediately:

$$\mathbf{N}(t+1) = \mathbf{N}(t) + a(t) - \mathbf{y}(t), \tag{6}$$

where $\mathbf{y}(t) = \{y_{ij}(t)\}$, with $0 \le y_{ij}(t) \le N_{ij}(t) + a_{ij}(t)$, $i \in I$, and $j \in J$, denotes the number of type i departures in server j at the end of tth time slot. Let $g_{ij}^t(n)$ be the probability distribution function of $y_{ij}(t)$; namely, $g_{ij}^t(n) = \Pr(y_{ij}(t) = n)$, and the explicit expression of $g_{ij}^t(n)$ is

$$g_{ij}^t(n) = \begin{cases} \binom{n_{ij}(t) + a_{ij}(t)}{n} (1-p_i)^{n_{ij}(t)+a_{ij}(t)-n} p_i^n, \\ \qquad n \le n_{ij}(t) + a_{ij}(t) \\ 0, \\ \qquad \text{otherwise.} \end{cases} \tag{7}$$

Let λ_i be subject to some discrete probability distribution function $f_i(x)$; that is, $\Pr(\lambda_i = n) = f_i(n)$; then the transition probability of the system is given by

$$P(\mathbf{N}(t+1), \boldsymbol{\lambda}(t+1) \mid \mathbf{N}(t), \boldsymbol{\lambda}(t), a(t)) = \begin{cases} 0, \\ \quad \text{if } \forall j,\ n_{ij}(t+1) > n_{ij} + a_{ij}(t) \\ \prod_{i \in I} f_i(\lambda_i(t+1)) \\ \quad \cdot \prod_{j \in J} \prod_{i \in I} g_{ij}^t\left(n_{ij}(t) + a_{ij}(t) - n_{ij}(t+1)\right), \\ \quad \text{otherwise.} \end{cases} \tag{8}$$

Rewards. The reward is defined by the VoD service provider, often in the form of $r-c$, where r and c are the revenue and the cost for serving a request, respectively. These two parameters can be defined from different perspectives. For example, from per request perspective, they can be earned right after a request is accepted (pay-per-view); from temporal perspective, they depend upon the sojourn time in the system of each request. In this paper we adopt the latter one. This is reasonable since the more time the client spends in the system, the more profits the VoD provider can potentially earn (e.g., by means of periodically popping up the embedded advertisements). The rewards earned in the tth slot are

$$R(\mathbf{N}(t), \boldsymbol{\lambda}(t), a(t)) = \sum_{i \in I} \sum_{j \in J} \left(n_{ij}(t) + a_{ij}(t)\right) \times \left(r_i - c_{ij}\right), \tag{9}$$

where $n_{ij}(t) + a_{ij}(t)$ is the number of type i clients in server j during the tth slot and $r_i - c_{ij}$ is the net gain for a type i request being served in server j.

Optimization Objective. The optimization objective is to maximize the expected accumulated long term system rewards. In practice the system is actually a finite horizon MDP since the arrival distribution is stationary only in a certain period (let us call it *stationary period*). However, we could use an infinite horizon MDP with discounted object function

$$\mathbb{E}\left\{\sum_{t=0}^{\infty} \delta^t R(\mathbf{N}(t), \boldsymbol{\lambda}(t), a(t))\right\} \tag{10}$$

to approximate it, where $0 \le \delta \le 1$ is the discounted factor, because the length of the stationary period (often in the order of several hours) is much longer than the length of the time slot (often in the order of seconds). This form of objective function is also used in [22]. Consider

$$V(\mathbf{N}(t), \boldsymbol{\lambda}(t)) = \max_{a(t)} \Big\{ R(\mathbf{N}(t), \boldsymbol{\lambda}(t), a(t)) + \delta \sum_{(\mathbf{N}(t+1), \boldsymbol{\lambda}(t+1)) \in \mathbf{S}} P(\mathbf{N}(t+1), \boldsymbol{\lambda}(t+1) \mid \mathbf{N}(t), \boldsymbol{\lambda}(t), a(t)) \cdot V \times (\mathbf{N}(t+1), \boldsymbol{\lambda}(t+1)) \Big\}. \tag{11}$$

The value function can be established as (11), and classical algorithms like value iteration or policy iteration can be used to solve this MDP. However, the above MDP model obviously suffers from the so-called state space explosion problem. To see that, suppose a system with p kinds of requests, q edge servers, and capacity r for each edge server; the total number of states with respect to $\mathbf{N}$ is

$$\left(\sum_{i=0}^{r} \binom{i+p-1}{i}\right)^q. \tag{12}$$

The computational cost of traditional algorithm, in which all states must be visited in each iteration, will be prohibitive. Therefore we need alternative approaches to obtain approximate solutions.

4. Solving the MDP Model Approximately

In this section we present two approaches to approximately solve the above MDP model of request routing problem. The first approach is the greedy strategy, which is an integer linear programming problem aiming at maximizing the reward in current time slot. The second approach is the bounded-parameter MDP (BMDP) strategy, which aggregates the state

space of the original MDP model and applies the interval value iteration algorithm on the aggregated model to obtain the B-optimal solution. We finally discuss the computational complexity of the BMDP strategy.

4.1. A Greedy Approximation Strategy. One of the simplest approximations is the greedy strategy, which focuses on maximizing the reward in each current slot instead of the cumulative reward in the long run. The problem can be summarized as an integer linear programming as follows:

$$\max_{a(t)} \sum_{i\in I} \sum_{j\in J} a_{ij}(t)\left(r_i - c_{ij}\right) \tag{13}$$

subject to constraints (4) and (5).

The idea behind the greedy strategy is straightforward: ignore the arrival pattern of requests and accept as many profitable requests as possible in a current slot. In fact, the greedy strategy does not require the server state **N**. Instead, it only needs to keep track of the total load of each individual server as

$$\mathbf{L} = \left\{L_j\right\}, \quad j \in J, \text{ where } L_j = \sum_{i\in I} n_{ij}(t). \tag{14}$$

We will evaluate the greedy strategy in Section 5.

The greatest advantage of the greedy strategy is its simplicity. Since it involves no iteration operations as in the classical iterative algorithm, its computational time is much smaller and thus can be implemented online. However, the biggest flaw of greedy strategy is that we do not know how far the profits of the greedy strategy deviate from the profits of the optimal strategy. This motivates us to find another approach which can provide some bounding information. In the following subsection, we provide the BMDP approach, which satisfies this property.

4.2. The Bounded-Parameter MDP (BMDP) Approximation Strategy. The BMDP was introduced by Givan et al. [41, 42] to provide an approximate approach for solving the MDP with a large state space; it can be categorized into a more general class known as Markov decision processed with imprecisely known transition probabilities (MDPIPs).

4.2.1. BMDP Preliminaries. A BMDP $M_\updownarrow$ is a four-tuple $\{S, A, R_\updownarrow, P_\updownarrow\}$. It is different from traditional MDP (also called *exact MDP*) in the sense that the reward function in each state $R_\updownarrow(s)$ and the transition probability $P_\updownarrow(s' \mid s, a)$ are specified by closed intervals $[R_\downarrow(p), R_\uparrow(p)]$ and $[P_\downarrow(s' \mid s, a), P_\uparrow(s' \mid s, a)]$, respectively, rather than exact point values. An exact MDP $M = \{S', A', R', P'\}$ is said to be contained in a BMDP $M_\updownarrow$ ($M \in M_\updownarrow$) as long as $S' = S, A' = A, R' \in R_\updownarrow$, and $P' \in P_\updownarrow$.

Given a policy π, the interval value function $V_{\updownarrow\pi}$ is defined by

$$V_{\updownarrow\pi}(s) = \left[\min_{M\in M_\updownarrow} V_{M,\pi}(s), \max_{M\in M_\updownarrow} V_{M,\pi}(s)\right], \tag{15}$$

where $V_{M,\pi}(s) = R_M(s) + \delta \sum_{s'\in S} P_M(s' \mid s, a) V_{M,\pi}(s')$ is the traditional value function for a specific exact MDP $M \in M_\updownarrow$. It can be proved that (see reference [41]) there exists a MDP $M \in M_\updownarrow$ which maximizes/minimizes $V_{M,\pi}(s)$ for all $s \in S$ simultaneously. We call such MDP *π-maximizing/π-minimizing* MDP.

The interval value functions cannot be compared using traditional MDP standard, which focuses on point values. To evaluate how "good" a policy π can achieve, we must define a scheme to compare interval value functions. In this paper we define the interval greater operator $\gtrsim$ given by

$$V_{\updownarrow 1} \gtrsim V_{\updownarrow 2} \Longleftrightarrow V_{\uparrow 1} \geq V_{\uparrow 2} \vee \left(V_{\uparrow 1} = V_{\uparrow 2} \wedge V_{\downarrow 1} \geq V_{\downarrow 2}\right). \tag{16}$$

The interval value iteration (IVI) algorithms with provable convergence property can be used to obtain the optimal solution to the BDMP; that is,

$$\begin{aligned} &\mathrm{IVI}_{\updownarrow\mathrm{opt}}\left(V_\updownarrow\right)(s) \\ &\quad = \max_{a\in A(s)} \left[\min_{M\in M_\updownarrow} \mathrm{VI}_{M,a}\left(V_\downarrow\right)(s), \max_{M\in M_\updownarrow} \mathrm{VI}_{M,a}\left(V_\uparrow\right)(s)\right] \end{aligned} \tag{17}$$

with $\mathrm{VI}_{M,a}(V)(s) = R_M(s) + \delta \sum_{s'\in S} P_M(s' \mid s, a) V(s')$.

The IVI algorithm (17) starts with an arbitrary interval value function $V_\updownarrow = [V_\downarrow, V_\uparrow]$ with $V_\downarrow \leq V_\uparrow$. The operation of finding exact MDPs $M \in M_\updownarrow$ inside the square bracket is equivalent to searching for order-maximizing *MDPs* with respect to state order sequences of decreasing $V_\uparrow$ and increasing $V_\downarrow$. Formal definition of the order-maximizing MDP for a specific state order sequence $O = s_1 s_2 \cdots s_n$ is given by the following definition.

Definition 2. The order-maximizing index r for state s and action a with respect to order O is

$$\arg\max_{1\leq r\leq n} \left\{\sum_{i=1}^{r-1} P_\uparrow\left(s_i \mid s, a\right) + \sum_{i=r}^{n} P_\downarrow\left(s_i \mid s, a\right)\right\}. \tag{18}$$

The order-maximizing MDP is an exact MDP $M_O \in M_\updownarrow$ satisfying

$$\begin{aligned} P_{M_O}\left(s_i \mid s, a\right) &= \begin{cases} P_\uparrow\left(s_i \mid s, a\right), & \text{if } i < r, \\ P_\downarrow\left(s_i \mid s, a\right), & \text{if } i > r, \end{cases} \\ P_{M_O}\left(s_r \mid s, a\right) &= 1 - \sum_{i=1, i\neq r}^{i=n} P_{M_O}\left(s_i \mid s, a\right). \end{aligned} \tag{19}$$

Note that the max operator in (17) uses (16) to compare interval value functions. It can be proved (see [41]) that the IVI algorithm will finally converge to an interval value function $[V_{\downarrow\mathrm{opt}}, V_{\uparrow\mathrm{opt}}]$ and an associated solution which we call B-optimal policy. The upper bound of interval value function $V_{\uparrow\mathrm{opt}}$ for the B-optimal policy is the possible best reward one can get from the BMDP $M_\updownarrow$; therefore, it can serve as a cornerstone to evaluate how far the value of a given strategy deviates from the one of possible optimal strategies for any exact MDP $M \in M_\updownarrow$. $V_{\downarrow\mathrm{opt}}$ is the possible worst case performance of the B-optimal strategy.

In our application, states in the BMDP $M_\updownarrow$ can be viewed as aggregations of states in the exact MDP M. The parameter

intervals in $M_{\updownarrow}$ represent the parameter ranges of states in M which belong to the same BMDP state. From this viewpoint $M_{\updownarrow}$ is a "smaller" approximation to the original M.

4.2.2. Model Reduction. The intuition behind our aggregation scheme is to overlook the request type in the server load matrix $\mathbf{N}$. We can construct an aggregate state space $\mathbf{S}'$ with an element $(\mathbf{L}, \boldsymbol{\lambda})$, where $\mathbf{L}$ is the server load vector specified in (14). The system dynamics equation becomes

$$\mathbf{L}(t+1) = \mathbf{L}(t) + a(t) \cdot \mathbf{e} - \mathbf{Y}(t), \tag{20}$$

where $\mathbf{e}$ is a j-dimension unit vector and $\mathbf{Y}$ is the total departures vector. Equation (20) means that the change of server state only depends upon the number of requests assigned to and departing from the server regardless of their types. With this abstraction, the number of state spaces with respect to $\mathbf{L}$ is r^q, a great reduction compared to (12).

4.2.3. The BMDP Reformulation. First we present the following lemma which is used to obtain the interval transition function of the BMDP formulation.

Lemma 3. *Suppose the system is in a particular aggregated state $(\mathbf{L}(t), \boldsymbol{\lambda}(t))$ at the tth slot; the probability of n clients leaving edge server j in this slot, namely, $Pr(Y_j(t) = n)$, can be bounded by*

$$\left[\binom{L_j(t) + \sum_{i \in I} a_{ij}(t)}{n} p_{\min}^n (1 - p_{\max})^{L_j(t) + \sum_{i \in I} a_{ij}(t) - n}, \binom{L_j(t) + \sum_{i \in I} a_{ij}(t)}{n} p_{\max}^n (1 - p_{\min})^{L_j(t) + \sum_{i \in I} a_{ij}(t) - n} \right] \tag{21}$$

provided that $n \le L_j(t) + \sum_{i \in I} a_{ij}(t)$, where $p_{\max} = \max_{i \in I} p_i$ and $p_{\min} = \min_{i \in I} p_i$.

Proof. We show how to derive the upper bound; the lower bound can be obtained using the same idea. Pick any exact MDP state $(\mathbf{N}(t), \boldsymbol{\lambda}(t))$ which belongs to the aggregated state $(\mathbf{L}(t), \boldsymbol{\lambda}(t))$; that is, $\sum_{i \in I} n_{ij}(t) = L_j(t)$. For server j in state $(\mathbf{N}(t), \boldsymbol{\lambda}(t))$, there are a total $\binom{L_j(t) + \sum_{i \in I} a_{ij}(t)}{n}$ number of cases such that n clients leave server j. Choose a particular case where the number of type i requests that leave server j is k_{ij}; we have $\sum_{i \in I} k_{ij}(t) = n$. The probability of the occurrence of this case is

$$\begin{aligned} &\prod_{i \in I} p_i^{k_{ij}} \times \prod_{i \in I} (1 - p_i)^{n_{ij}(t) + a_{ij}(t) - k_{ij}} \\ &\quad \le p_{\max}^{\sum_{i \in I} k_{ij}} (1 - p_{\min})^{\sum_{i \in I} n_{ij}(t) + a_{ij}(t) - k_{ij}} \\ &\quad = p_{\max}^n (1 - p_{\min})^{L_j(t) + \sum_{i \in I} a_{ij}(t) - n}. \end{aligned} \tag{22}$$

The results can be followed immediately. □

We can now reformulate the problem with the BMDP.

States. The aggregated state space is $\mathbf{S}'$, with an element denoted by $(\mathbf{L}, \boldsymbol{\lambda})$.

Decision Epoch and Actions. Decision epoch and actions are the same as in the original MDP model.

Interval Transition Probability. Note that, in the BMDP formulation, the transition probability is a closed interval. First observe that

$$\begin{aligned} &P(\mathbf{L}(t+1), \boldsymbol{\lambda}(t+1) \mid \mathbf{L}(t), \boldsymbol{\lambda}(t), a(t)) \\ &\quad = \prod_{i \in I} f_i(\lambda_i(t+1)) \cdot \prod_{j \in J} P\left(L_j(t+1) \mid L_j(t), \boldsymbol{\lambda}(t), a(t)\right) \\ &\quad = \prod_{i \in I} f_i(\lambda_i(t+1)) \\ &\qquad \cdot \prod_{j \in J} P\left(Y_j(t) = L_j(t) + \sum_{i \in I} a_{ij}(t) - L_j(t+1)\right). \end{aligned} \tag{23}$$

Combining with Lemma 3, the transition probability in (23) can be bounded by the interval transition shown in (24) with $\Theta_j(t) = L_j(t) + \sum_{i \in I} a_{ij}(t)$ and $Y_j(t) = L_j(t) + \sum_{i \in I} a_{ij}(t) - L_j(t+1)$ denoting the number of requests and the number of departures in tth slot in server j regardless of their types, respectively. Consider

$$\begin{aligned} &P_{\updownarrow}(\mathbf{L}(t+1), \boldsymbol{\lambda}(t+1) \mid \mathbf{L}(t), \boldsymbol{\lambda}(t), a(t)) \\ &\quad = \begin{cases} [0, 0], \\ \quad \text{if } L_j(t) + \sum_{i \in I} a_{ij}(t) \le L_j(t+1) \\ \left[p_{\min}^{\sum_{j \in J} Y_j(t)} (1 - p_{\max})^{\sum_{j \in J} L_j(t+1)} \right. \\ \quad \times \prod_{i \in I} f_i(\lambda_i(t+1)) \prod_{j \in J} \binom{\Theta_j(t)}{Y_j(t)}, \\ \quad \left. p_{\max}^{\sum_{j \in J} Y_j(t)} (1 - p_{\min})^{\sum_{j \in J} L_j(t+1)} \right. \\ \quad \left. \times \prod_{i \in I} f_i(\lambda_i(t+1)) \prod_{j \in J} \binom{\Theta_j(t)}{Y_j(t)} \right], \\ \quad \text{otherwise.} \end{cases} \end{aligned} \tag{24}$$

Interval Rewards. The reward consists of two parts. The first part is the reward for the action $a(t)$, which is a fixed value given by $\sum_{j \in J} \sum_{i \in I} a_{ij}(t)(r_i - c_{ij})$. The second part is a closed interval $[mL(t), ML(t)]$, with

$$\begin{aligned} M &= \max_{i \in I, j \in J} \{r_i - c_{ij}\}, \\ m &= \min_{i \in I, j \in J} \{r_i - c_{ij}\}. \end{aligned} \tag{25}$$

The interval reward function can then be expressed by

$$R_{\updownarrow}(\mathbf{L}(t), \boldsymbol{\lambda}(t), \mathbf{a}(t)) = \left[mL(t) + \sum_{j\in j}\sum_{i\in I} a_{ij}(t)\left(r_i - c_{ij}\right), \; ML(t) + \sum_{j\in j}\sum_{i\in I} a_{ij}(t)\left(r_i - c_{ij}\right) \right]. \quad (26)$$

4.2.4. The Interval Value Iteration (IVI) Algorithm. We illustrate the IVI algorithm in detail in Algorithms 1 and 2.

Algorithm 1 takes the above BMDP model $M_{\updownarrow}$ as input. It first chooses an initial interval value function $V_{\updownarrow}^{0}$ and then iteratively calls function IVI. The algorithm finishes with an interval value function $V_{\updownarrow}$ and a corresponding policy π.

Algorithm 2 captures the essence of (17). The increasing state order of $V_{\uparrow}$ and the decreasing state order of $V_{\downarrow}$ are stored in O_{up} and O_{down}. For each state s and action a, the algorithm computes the order-maximizing indices r_{up} and r_{down} for order sequences O_{up} and O_{down} and forms two order-maximizing MDPs with transition probabilities P'_{up} and P'_{down}. Using P'_{up}, the set of actions $\mathbf{a}$ which maximizes the upper bound $V'_{\uparrow}$ is identified. If $\mathbf{a}$ contains only one element, for example, $\mathbf{a} = \{a\}$, then a is the B-optimal action in this state and the lower bound can be obtained immediately. Otherwise, an action $a \in \mathbf{a}$ which maximizes the lower bound $V'_{\downarrow}$ is chosen as the B-optimal action.

4.2.5. A Word on Computational Complexity

(i) *Space Complexity*. Although the memory consumption of a single state in the BMDP model is a bit larger than the one in the MDP model due to the interval transition probability and the interval reward, but since the aggregation scheme in our BMDP formulation dramatically reduces the state space of edge servers from $(\sum_{i=0}^{r}\binom{i+p-1}{i})^{q}$ to r^q, the total memory space needed for storing the BMDP model is significantly less than the MDP model. On the other hand, the IVI algorithm only needs additional O_{up} and O_{down} to store order sequences in BMDP model, r_{up} and r_{down} to store order-maximizing indices, and P'_{up} and P'_{down} to store transition probabilities for the π-maximizing MDP and the π-minimizing MDP, respectively. In all, the memory space needed to store the BMDP model and implement the IVI algorithm decreases greatly compared with the memory space needed for the MDP model.

(ii) *Time Complexity*. In the IVI algorithm, the first optimization problem $(*)$ is much easier than problem (11) in traditional value iteration algorithm since the computation of the expectation in the Bellman equation involves much less states. There are three extra computational burdens in the IVI algorithm: (1) sorting states to obtain O_{up} and O_{down} (lines 5 and 6 in Algorithm 2), which take $O(2n^2)$ in worst case using Quicksort algorithm; (2) finding the order-maximizing indices and computing the transition probabilities for the π-maximizing MDP and the π-minimizing MDP (lines 9, 10, and 12 in Algorithm 2), which take $O(4n)$ in worst case; (3) finding a unique solution $a \in \mathbf{a}$ to the problem $(**)$ which maximizes the lower bound $V_{\downarrow}$ (line 20 in Algorithm 2). In this step we use the exhaustive search, so the time complexity is $O(|\mathbf{a}|)$. However, regarding $\mathbf{a}$ which is usually a small set, it will not take too long to complete the search. Thus, the total extra time complexity is $O(2n^2 + 4n + |\mathbf{a}|)$.

Input: a BMDP $M_{\updownarrow} = \{S, A, P_{\updownarrow}, R_{\updownarrow}\}$
Output: $V_{\updownarrow}$ and π
(1) Create π; {π holds the strategy in each iteration.}
(2) Create $V_{\updownarrow}$; {$V_{\updownarrow}$ is an initial interval value function.}
(3) **loop**
(4) $\quad V_{\updownarrow} = \text{IVI}_{\updownarrow\text{opt}}(M_{\updownarrow}, V_{\updownarrow}, \&\pi)$;
(5) **end loop**

Algorithm 1: Solutions to the BMDP formulation.

5. Numerical Analysis

In this section we study an illustrative system with two video request types and two edge servers. To alleviate the computational and simulation burden, we use a small scale system parameterization. We initially set the server capacity $C_1 = C_2 = 5$. The expected sojourn time of type 1 and type 2 requests (T_1 and T_2) is 2000 s and 1000 s, respectively. According to (3), the departure probabilities can be computed as $p_2 = 0.0025$ and $p_1 = 0.005$ if we set the length of the time slot to 5 s. Other parameters are summarized in Table 2.

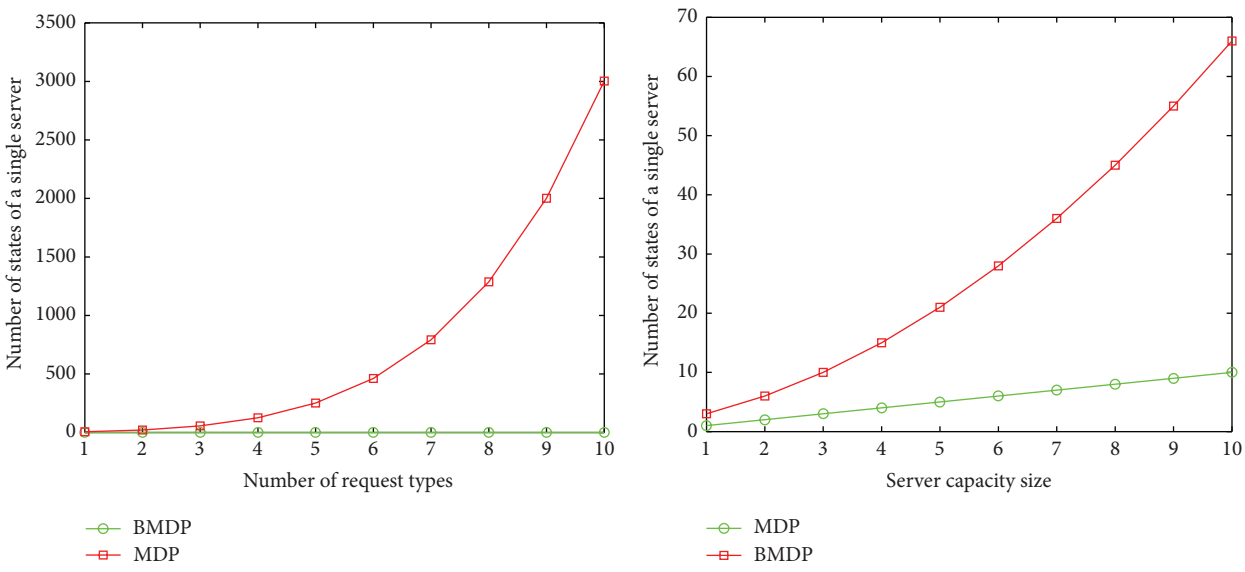
5.1. Problem State Space. First we will see the effectiveness of the state aggregation scheme of the proposed BMDP formulation. We compare the state space of a single server here with respect to MDP formulation and BMDP formulation. The results are shown in Figure 4.

In Figure 4(a), we fix the server capacity to 5 and vary the number of request types. In Figure 4(b), we fix the number of request types to 2 and vary the server capacity. The state space of the BMDP formulation does not change with the number of request types and grows linearly with the server capacity. In contrast, the state space of the MDP formulation increases exponentially with both the number of request types and the server capacity. There are over 3000 states when the number of request type reaches 10, demonstrating that the MDP model is practically intractable.

On the other hand, although the state space of a single server in the BMDP formulation is greatly reduced compared to the counterpart in the MDP formulation, the state space of the whole system in the BMDP formulation still grows

Input: a BMDP $M_{\updownarrow}$, a value function $V_{\updownarrow}$, and a place π for holding the policy in current iteration
Output: $V'_{\updownarrow \text{opt}}$ and π_{opt}
(1) Create O_{up}, O_{down}; {O_{up} and O_{down} hold order sequence of states in $M_{\updownarrow}$, i.e., $s_1 s_2, \ldots, s_n$.}
(2) Create P'_{up}, P'_{down}; {P'_{up} and P'_{down} hold the transition probabilities for the order-maximizing MDP with respect to order O_{up} and O_{down}, respectively.}
(3) Create r_{up}, r_{down}; {r_{up} and r_{down} is the order-maximizing index for order sequences O_{up} and O_{down}, respectively.}
(4) Create i; {i is the index into an ordering O.}
(5) O_{up} = Sort_Decreasing_Order($V_{\uparrow}$);
(6) O_{down} = Sort_Increasing_Order($V_{\downarrow}$);
(7) **for all** $s \in S$ **do**
(8) **for all** $a \in A$ **do**
(9) r_{up} = Order_Maximizing_Ind($M_{\updownarrow}, O_{\text{up}}, s, a$);
(10) r_{down} = Order_Maximizing_Ind($M_{\updownarrow}, O_{\text{down}}, s, a$);
{find order-maximizing index for transition probability in state s under action a according to (18).}
(11) **for** $i = 1$ to n **do**
(12) Update $P'_{\text{up}}(s_{O_{\text{down}}(i)} \mid s, a)$ and $P'_{\text{down}}(s_{O_{\text{up}}(i)} \mid s, a)$ according to (19);
(13) **end for**
(14) **end for**
(15) $V'_{\uparrow} = \max_{\mathbf{a} \subseteq A(s)} R_{\uparrow}(s, a) + \delta \sum_{s' \in S} P'_{\text{up}}(s's, a) V_{\uparrow}(s')$;(∗)
(16) **if** $|\mathbf{a}| = 1$ and $\mathbf{a} = \{a\}$ **then**
(17) $V'_{\downarrow} = R_{\downarrow}(s, a) + \delta \sum_{s' \in S} P'_{\text{down}}(s's, a) V_{\downarrow}(s')$;
(18) $\pi(s) = a$;
(19) **else**
(20) $V'_{\downarrow} = \max_{a \in \mathbf{a}} R_{\downarrow}(s, a) + \delta \sum_{s' \in S} P'_{\text{down}}(s's, a) V_{\downarrow}(s')$;(∗∗)
(21) $\pi(s) = a$;
(22) **end if**
(23) **end if**

Algorithm 2: Interval value iteration (IVI$_{\updownarrow}$) algorithm.



(a) State size versus number of request types, with server capacity = 5

(b) State size versus server capacity, with number of request types = 2

Figure 4: Problem state space size of MDP and BMDP formulations.

exponentially as the number of edge servers increases. This problem can be addressed by further aggregation of the state space of a single server. However, the greater the degree of aggregation is, the more the transition probability information is lost, which may degrade the performance of the generated solution. In practice, the CDN provider must make tradeoff between the quality of the generated solution and the computational cost. We leave this issue in our future works.

5.2. Convergence and Application of the IVI Algorithm. We initially set the interval value function for all states to $[0, 0]$ and other parameters are defined in Tables 2 and 3. The interval value function of state $[\mathbf{0}, \mathbf{0}]$ in each iteration is plotted in Figure 5. It converges after about 600 iterations.

The IVI algorithm is not suitable for online scheduling since it involves time-consuming iterative steps, which cannot cope with variations of online request arrival pattern. However, like the Internet traffic, the Internet video streaming also demonstrates a strong *temporal pattern* [44]. We could divide a day into several parts according to the request arrival patterns obtained by network measurement and compute a particular strategy for each part offline. The dispatcher can adopt the associated strategy in a given time interval to yield close-to-optimal rewards.

5.3. Performance for the Greedy Strategy and the B-Optimal Strategy. We evaluate the greedy strategy and the B-optimal strategy in two ways.

(1) The value function V_{greedy} for the greedy strategy in the MDP model and the interval value function $[V_{\downarrow\text{opt}}, V_{\uparrow\text{opt}}]$ for the B-optimal strategy in the BMDP model, which is plotted in Figure 6.

(2) The ratio of $V_{\downarrow\text{opt}}/V_{\uparrow\text{opt}}$ and $V_{\text{greedy}}/V_{\uparrow\text{opt}}$, which is plotted in Figure 7.

In both Figures 6 and 7 we vary the arrival probability of two request types from 0.01 to 0.2.

The first observation from Figure 6 is that as the arrival traffic grows heavier, the upper bound and the lower bound of the B-optimal strategy, as well as the value of the greedy strategy, all tend to a steady state. This is because the system approaches to a saturated state. Another phenomenon is that the B-optimal strategy always outperforms the greedy strategy since even the lower bound of the B-optimal strategy is greater than the greedy strategy, meaning that the B-optimal algorithm is effective in this system.

We can have a clearer view of the quality of the proposed strategies in Figure 7. The lower bound of performance $V_{\downarrow\text{opt}}$ in the worst case for B-optimal strategy can attain around 80% to over 95% of the upper bound $V_{\uparrow\text{opt}}$. For the greedy strategy, V_{greedy} can attain around 65% to 80% of the upper bound $V_{\uparrow\text{opt}}$. An interesting finding is that there exists a conspicuous "jumping line" which divides $V_{\downarrow\text{opt}}/V_{\uparrow\text{opt}}$ into two parts (a light load part and a heavy load part). The ratio $V_{\downarrow\text{opt}}/V_{\uparrow\text{opt}}$ grows in the light load part, surges to the peak at the jumping line, and begins to decline slowly in the heavy load part, as the request arrival probabilities increase.

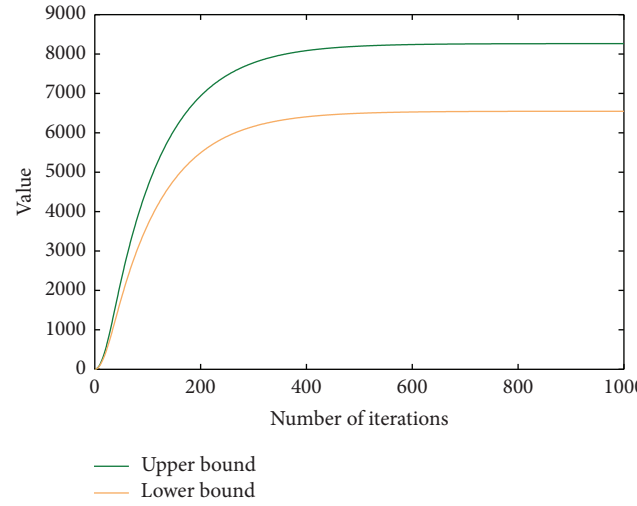


Figure 5: Convergence of IVI algorithm.

Table 2: Parameters setting.

	Type 1 request		Type 2 request	
	Server 1	Server 2	Server 1	Server 2
Cost	2	2.5	1	2
Reward	12		10	
Expected sojourn time	2000		1000	
Upper bound of sojourn time	2500		1200	

Table 3: Arrival and departure distribution.

	Type 1 request		Type 2 request	
Number of arrivals	0	1	0	1
Arrival probability	0.8	0.2	0.9	0.1
Departure probability	0.0025		0.005	

The heavy load part of $V_{\downarrow\text{opt}}/V_{\uparrow\text{opt}}$ still stands beyond 90%, suggesting that the B-optimal strategy has an outstanding performance under heavy load even in the worst case. On the other hand $V_{\text{greedy}}/V_{\uparrow\text{opt}}$ reaches the trough at two positions: (1) extreme light load; (2) nearly the same load region of the jumping line of $V_{\downarrow\text{opt}}/V_{\uparrow\text{opt}}$, indicating that the performance of the greedy strategy may degrade at these load regions. However, in most of cases, $V_{\text{greedy}}/V_{\uparrow\text{opt}}$ remains flat (around 80%).

6. Concluding Remarks

In this paper we consider the dynamic request routing issue in the Video-on-Demand system. We classify video requests by the expected viewing time and the average bitrates of the video files. The system can be abstracted into a controlled queueing system containing one dispatcher with its waiting queue and several VoD edge servers with their service queues. Our goal is to find the decision policy of the dispatcher

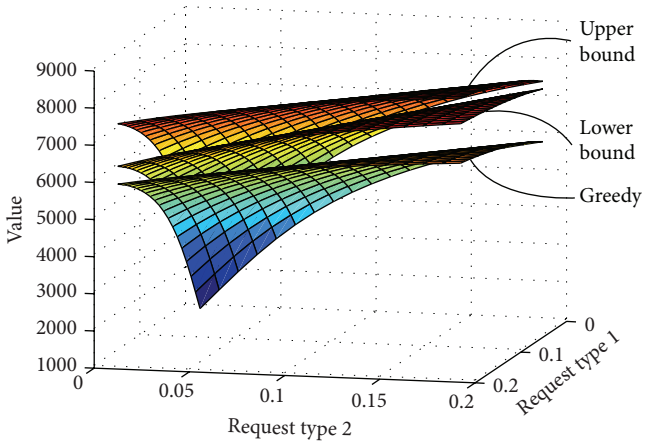


Figure 6: Upper bound and lower bound versus arrival probability.

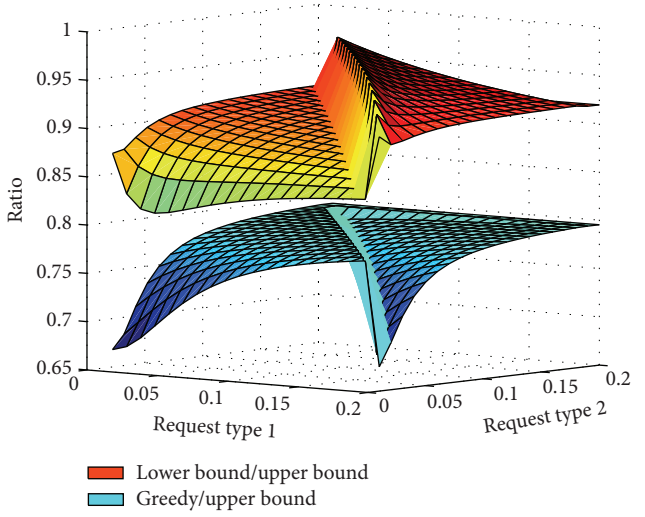


Figure 7: $V_{\downarrow opt}/V_{\uparrow opt}$ = lower bound/upper bound and $V_{greedy}/V_{\uparrow opt}$ = greedy/upper bound versus arrival probability.

which yields the highest reward. The dynamic request routing problem can be formulated as a Markov decision process, and classical iterative algorithm can be used to obtain the optimal solution.

However, the MDP formulation has its intrinsic drawback of the curse of dimensionality, which makes the problem intractable in practical scenario. To address this issue, we present two alternative approaches, that is, the greedy strategy and the bounded-parameter MDP reformulation, to approximately compute the suboptimal solution. These two approximation schemes start from different points: the greedy strategy ignores the request arrival patterns in the future and the BMDP reformulation overlooks the types of request in the server load. Although the greedy strategy is much simpler, the numerical results show that the B-optimal strategy can generate a higher reward than the greedy strategy. Our future research will concentrate on the distributed implementation of dynamic request routing strategy for VoD service.

Notation

I : Set of request types
J : Set of edge servers
$\mathbf{S}$: State space of the original MDP formulation
$\mathbf{S}'$: State space of the BMDP formulation
$\mathbf{N}$: System state matrix, with an element n_{ij} denoting the number of type i requests in edge server j
$\mathbf{L}$: System state vector, with an element L_j denoting the number of requests in edge server j
$\boldsymbol{\lambda}$: Arrival vector, with an element λ_i denoting the number of type i requests arrived at the dispatcher in one slot
$\mathbf{y}$: Request departure matrix, with element y_{ij} denoting the number of departures for type i request in edge server j
$\mathbf{Y}$: Request departure vector, with element Y_j denoting the number of departures in edge server j
Γ : Length of a time slot
T_i : Expected sojourn time in the system for type i request
$\overline{T}_i$: Upper bound of sojourn time in the system for type i request
p_i : Departure probability of type i request
C_j : Bandwidth capacity of edge server j
ω_i : Amount of bandwidth consumed by a type i request
c_{ij} : Costs of edge server j for serving a type i request in one slot
r_i : Rewards for serving a type i request in one slot
a : Action matrix at time t, with an element a_{ij} denoting the number of type i requests forwarded to edge server j in one slot.

Conflict of Interests

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

Acknowledgments

The work is funded in part by the National Natural Science Foundation of China (NSFC) under Grant no. 61363052, the Inner Mongolia Provincial Natural Science Foundation under Grant nos. 2010MS0913, and 2013MS0920 and the Science Research Project for Inner Mongolia College under Grant nos. NJZY14064 and NJZY13120.

References

[1] Akamai, "State of the Internet," http://www.akamai.com.
[2] YouTube, http://www.youtube.com.
[3] Hulu, http://www.hulu.com.

[4] A. Lenk, M. Klems, J. Nimis, S. Tai, and T. Sandholm, "What's inside the cloud? An architectural map of the cloud landscape," in *Proceedings of the ICSE Workshop on Software Engineering Challenges of Cloud Computing (CLOUD '09)*, pp. 23–31, May 2009.

[5] Y. Feng, B. Li, and B. Li, "Airlift: video conferencing as a cloud service using inter-datacenter networks," in *Proceeding of the 20th IEEE International Conference on Network Protocols (ICNP '12)*, pp. 1–11, Austin, Tex, USA, November 2012.

[6] F. Wang, J. Liu, and M. Chen, "CALMS: cloud-assisted live media streaming for globalized demands with time/region diversities," in *Proceeding of the IEEE Conference on Computer Communications (INFOCOM '12)*, pp. 199–207, Orlando, Fla, USA, March 2012.

[7] Y. Li, Y. Shen, and Y. Liu, "Utilizing content delivery network in cloud computing," in *Proceeding of the International Conference on Computational Problem-Solving (ICCP '12)*, pp. 137–143, Leshan, China, October 2012.

[8] H. A. Tran, A. Mellouk, and S. Hoceini, "QoE content distribution network for cloud architecture," in *Proceeding of the 1st IEEE Symposium on Network Cloud Computing and Applications (NCCA '11)*, pp. 14–19, Toulouse, France, November 2011.

[9] Y. Jin, Y. Wen, G. Shi, G. Wang, and A. V. Vasilakos, "CoDaaS: an experimental cloud-centric content delivery platform for user-generated contents," in *Proceeding of the International Conference on Computing, Networking and Communications (ICNC '12)*, pp. 934–938, Maui, Hawaii, USA, February 2012.

[10] L. Chia-Feng, L. Muh-Chy, C. Chih-Wei, and Y. Shyan-Ming, "The study and methods for cloud based CDN," in *Procedding of the 3rd International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC '11)*, pp. 469–475, Sanya, China, October 2011.

[11] F. Chen, K. Guoy, J. Liny, and T. La Porta, "Intra-cloud lightning: building CDNs in the cloud," in *Proceeding of the IEEE INFOCOM*, pp. 433–441, Orlando, Fla, USA, March 2012.

[12] Amazon CloudFront, http://aws.amazon.com/cloudfront/.

[13] Akamai NetStorage, http://www.akamai.com/html/technology/products/netstorage.html.

[14] http://www.limelight.com/.

[15] Z. Zhang, M. Zhang, A. Greenberg, Y. C. Hu, R. Mahajan, and B. Christian, "Optimizing cost and performance in online service provider networks," in *Proceeding of the 7th USENIX Symposium on Networked Systems Design and Implementation (NSDI '10)*, 2010.

[16] M. Andrews, B. Shepherd, A. Srinivasan, P. Winkler, and F. Zane, "Clustering and server selection using passive monitoring," in *Proceeding of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '02)*, vol. 3, pp. 1717–1725, New York, NY, USA, June 2002.

[17] O. Ardaiz, F. Freitag, and L. Navarro, " Improving the service time of web clients using server redirection," *ACM SIGMETRICS Performance Evaluation Review*, vol. 29, no. 2, pp. 39–44, 2001.

[18] P. Wendell, J. W. Jiang, M. J. Freedman, and J. Rexford, "DONAR: decentralized server selection for cloud services," in *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '10)*, New Delhi, India, September 2010.

[19] R. Torres, A. Finamore, J. R. Kim, M. Mellia, M. M. Munafo, and S. Rao, "Dissecting video server selection strategies in the YouTube CDN," in *Proceeding of the 31st International Conference on Distributed Computing Systems (ICDCS '11)*, pp. 248–257, Minneapolis, Minn, USA, July 2011.

[20] S. Ao-Jan, D. R. Choffnes, A. Kuzmanovic, and F. E. Bustamante, "Drafting behind Akamai (travelocitybased detouring)," in *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '06)*, pp. 435–446, Pisa, Italy, September 2006.

[21] N. Carlsson and D. L. Eager, "Server selection in large-scale video-on-demand systems," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 6, no. 1, article 1, 2010.

[22] L. Liu and Y. Lu, "Dynamic traffic controls for web-server networks," *Computer Networks*, vol. 45, no. 4, pp. 523–536, 2004.

[23] C. Lin, Y. Jun, P. Jianping, S. Xuemin (Sherman), and J. W. Mark, "Dynamic server selection using fuzzy inference in content distribution networks," *Computer Communications*, vol. 29, no. 8, pp. 1026–1038, 2006.

[24] Z. Fei, M. H. Ammar, and E. W. Zegura, "Optimal allocation of clients to replicated multicast servers," in *Proceedings of the 7th International Conference on Network Protocols (ICNP '99)*, pp. 69–76, October 1999.

[25] H. Xu and B. Li, "Joint request mapping and response routing for geo-distributed cloud services," in *Proceedings of IEEE INFOCOM*, pp. 854–862, Turin, Italy, April 2013.

[26] H. Xu and B. Li, "A general and practical datacenter selection framework for cloud services," in *Proceeding of the IEEE 5th International Conference on Cloud Computing (CLOUD '12)*, pp. 9–16, Honolulu, Hawaii, USA, June 2012.

[27] H. A. Tran, A. Mellouk, J. Perez, S. Hoceini, and S. Zeadally, "QoE-based server selection for content distribution networks," *IEEE Transactions on Computers*, 2013.

[28] H. Qian and M. Rabinovich, "Application placement and demand distribution in a global elastic cloud: a unified approach," in *Proceeding of the 10th International Conference on Autonomic Computing (ICAC '13)*, San Jose, Calif, USA, June 2013.

[29] M. Hajjat, P. N. Shankaranarayanan, and D. Maltz, "Dealer: application-aware request splitting for interactive cloud applications," in *Proceeding of the 8th ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT '12)*, pp. 157–168, Nice, France, December 2012.

[30] C. Ding, Y. Chen, T. Xu, and X. Fu, "CloudGPS: a scalable and ISP-friendly server selection scheme in cloud computing environments," in *Proceeding of the IEEE 20th International Workshop on Quality of Service (IWQoS '12)*, Coimbra, Portugal, June 2012.

[31] B. Frank, I. Poese, Y. Lin et al., "Pushing CDN-ISP collaboration to the Limit," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 3, pp. 34–44, 2013.

[32] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley-Interscience, New York, NY, USA, 1994.

[33] Z. Liu, M. S. Squillante, and J. L. Wolf, "On maximizing service-level-agreement profits," in *Proceedings of the 3rd ACM Conference on Electronic Commerce (EC '01)*, pp. 223–213, Tampa, Fla, USA, October 2001.

[34] L. Zhang and D. Ardagna, "SLA based profit optimization in autonomic computing systems," in *Proceedings of the 2nd International Conference on Service Oriented Computing (ICSOC '04)*, pp. 173–182, New York, NY, USA, November 2004.

[35] E. Casalicchio and M. Colajanni, "A client-aware dispatching algorithm for web clusters providing multiple services," in *Proceedings of the 10th ACM International Conference on World Wide Web (WWW '01)*, pp. 535–544, Hong Kong, May 2001.

[36] M. Colajanni and P. S. Yu, "Adaptive TTL schemes for load balancing of distributed web servers," *ACM SIGMETRICS Performance Evaluation Review*, vol. 25, pp. 36–42, 1997.

[37] M. Colajanni, P. S. Yu, and V. Cardellini, "Dynamic load balancing in geographically distributed heterogeneous web servers," in *Proceedings of the 18th International Conference on Distributed Computing Systems (ICDCS '98)*, pp. 295–302, Amsterdam, The Netherlands, May 1998.

[38] M. Conti, E. Gregori, and F. Panzieri, "Load distribution among replicated web servers: a QoS-based approach," *ACM SIGMETRICS Performance Evaluation Review*, vol. 27, no. 4, pp. 12–19, 2000.

[39] J. Cao, Y. Sun, X. Wang, and S. K. Das, "Scalable load balancing on distributed web servers using mobile agents," *Journal of Parallel and Distributed Computing*, vol. 63, no. 10, pp. 996–1005, 2003.

[40] G. Ciardo, A. Riska, and E. Smirni, "EquiLoad: a load balancing policy for clustered web servers," *Performance Evaluation*, vol. 46, no. 2-3, pp. 101–124, 2001.

[41] R. Givan, S. Leach, and T. Dean, "Bounded-parameter Markov decision processes," *Artificial Intelligence*, vol. 122, no. 1, pp. 71–109, 2000.

[42] R. Givan, S. Leach, and T. Dean, "Bounded parameter Markov decision processes," in *Computer Science*, S. Steel and R. Alami, Eds., vol. 1348 of *Lecture Notes*, pp. 234–246, Springer, Berlin, Germany, 1997.

[43] F. Thouin and M. Coates, "Video-on-demand networks: design approaches and future challenges," *IEEE Network*, vol. 21, no. 2, pp. 42–48, 2007.

[44] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng, "Understanding user behavior in large-scale video-on-demand systems," in *Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems (EuroSys '06)*, pp. 333–344, Leuven, Belgium, April 2006.