LP-SR: Approaching Optimal Storage and Retrieval for Video-on-Demand

S.-H. Gary Chan, *Senior Member, IEEE*, and Zhuolin Xu

***Abstract*—In a distributed large-scale video-on-demand (VoD) streaming network, a content provider often deploys local servers close to their users. A movie is partitioned into k segments which the servers collaboratively replicate and retrieve ($k \geq 1$). A critical but challenging problem is how to minimize overall system deployment cost consisting of server bandwidth, server storage, and network traffic among servers. In this paper, we address this problem through jointly optimizing movie storage and retrieval in the server network. We first formulate the optimization problem and show that it is NP-hard. To address the problem, we propose a novel, effective and implementable heuristic termed LP-SR. LP-SR decomposes the optimization problem into two computationally efficient linear programs (LPs) for segment storage and retrieval, respectively. The strength of LP-SR is that it is *asymptotically optimal* in terms of k, and k is not high to be closely optimal (around 5 to 10 in our study). For large movie pool, we propose a movie grouping algorithm to further reduce the computational complexity without compromising much on the performance. Through extensive simulation, LP-SR is shown to perform significantly the best as compared with other state-of-the-art and traditional schemes, reducing the deployment cost by a wide margin (by multiple times in many cases). It attains performance very close to the global optimum.**

***Index Terms*—Distributed video-on-demand, linear programming, optimization, segment storage and retrieval.**

I. Introduction

In order to provide cost-effective video-on-demand (VoD) streaming service scalable to large number of users, a content provider often deploys distributed servers placed close to user pools. These servers cooperatively replicate and retrieve movies given movie popularity. Such architecture is able to greatly reduce network load and scale up the streaming and storage capacity of the network [1]. In this paper, we consider the critical and challenging problem of minimizing the system deployment cost of VoD streaming through optimizing movie storage and retrieval in the servers. The cost model we use is general and comprehensive, capturing server storage, server bandwidth utilization and network traffic among the servers.

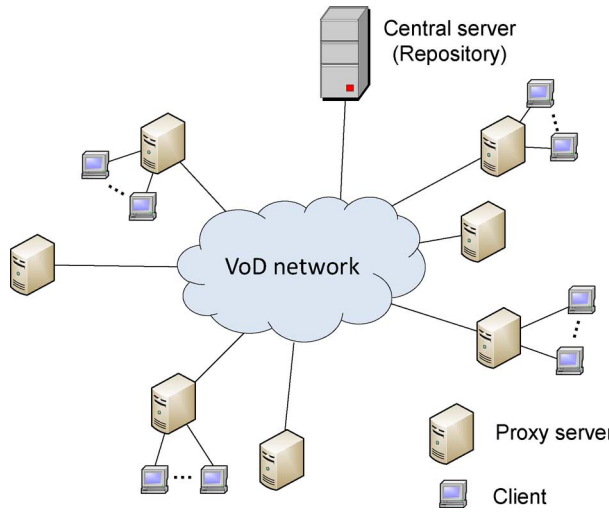


Fig. 1. A distributed servers architecture for VoD service.

We show in Fig. 1 a typical distributed and cooperative VoD network consists of a central server (or repository) storing all the movies and proxy servers placed close to user pools.[1] While the central server stores all the movies, the proxy servers are of possibly heterogeneous storage which may be able to replicate only a fraction of the movies. Each user has a home (or local) server to serve his request. If the request is a hit, the home server directly streams to the users. Otherwise (i.e., a miss), the home server pulls the content from a remote server (either a proxy server or the central server) to stream to the request. In other words, the bandwidth of the servers are used to stream not only its own home users (if any), but also remote servers requesting their contents.[2] Note that to minimize interactive delay, movie segments are *streamed* in the network, i.e., they are *not* downloaded at the clients before being played back. (To further reduce user startup delay, one may consider pre-storing the "prefixes" (the first, say, 30 seconds) of the movies at the servers. In any case, such prefix operation is orthogonal to our current study.)

The deployment cost of such a VoD network mainly consists of two major components, *server cost* due to the total storage and upload/streaming rate of a server, and *network cost* due to the bandwidth used between pairs of servers to serve the misses. Given movie popularity, a challenging problem is hence which

Manuscript received September 15, 2012; revised March 05, 2013; accepted June 03, 2013. Date of publication September 05, 2013; date of current version November 13, 2013. This work was supported in part by HKUST (FSGRF12EG05 and FSGRF13EG15) and Microsoft Research Asia Windows Phone Academic Program (MRA12EG01). The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Selcuk Candan.

S.-H. G. Chan is with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong (e-mail: gchan@cse.ust.hk; see http://www.cse.ust.hk/~gchan).

Z. Xu was with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, when this work was done.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2013.2280989

[1]In this paper, we use "client" and "user" interchangeably. We also use "movie," "video" and "content" interchangeably.

[2]In this paper, we use the term "servers" to collectively refer to the central and proxy servers.

movies to store (or replicate) and where to access them in order to minimize the deployment cost for interactive streaming.

For efficient server storage and retrieval, each movie is considered to be partitioned into k segments ($k \geq 1$). We formulate the cost-minimization problem of optimizing movie storage and retrieval and show that it is NP-hard. To make it tractable, we propose a novel and efficient heuristic termed LP-SR which decomposes the problem into two linear programs (LPs) for segment storage and retrieval, respectively. The salient feature and strength of LP-SR is that it is *asymptotically optimal* in k, i.e., as k increases, its performance approaches global optimum. Furthermore, our results show that k does not need to be large (say 5–10) for the system to be closely optimal (within 6.5% deviation in our simulation).

Note that LP-SR is orthogonal and applicable to any estimation and prediction algorithms of movie popularity and traffic [2]–[6]. It may be run regularly based on the estimation or prediction interval (e.g., on a nightly basis). Therefore, LP-SR can keep the deployment cost low as the popularity, number of movies and traffic in the system change over time.

Our contributions of this work are three-folds:

- *Comprehensive consideration of system deployment cost:* We present a realistic and general model on the deployment cost of VoD, which includes comprehensively server bandwidth utilization, storage and network transmission cost. (Previous work in VoD seldom considers all these factors together.) We formulate the joint optimization problem for movie storage and retrieval and show that it is NP-hard.
- *LP-SR: Achieving asymptotic optimality for video-on-demand:* We propose LP-SR which decomposes the original NP-hard problem into two linear programming (LP) problems for segment storage and retrieval, respectively. These LPs can be efficiently solved in polynomial time. LP-SR is asymptotically optimal in k, i.e., the system cost approaches the global minimum as k increases. With LP-SR, the network is able to make the best use of limited server storage, efficiently utilize the available server bandwidth, and substantially save network traffic cost. To reduce the computational complexity for large number of movies, we further propose an efficient movie grouping algorithm which achieves close optimality but reduces the complexity by a polynomial factor of $O(g^3)$, where g is the group size of the movies.
- *Extensive simulation study:* We conduct extensive simulation and comparison study of LP-SR with both state-of-the-art and traditional schemes. Our results show that LP-SR achieves substantially the lowest system cost, outperforming others significantly by a wide margin (by multiple times in many cases). Our results show that many existing heuristics are still quite far from the global optimum, and LP-SR can achieve performance very close to such optimum.

This work is organized as follows. We briefly review the previous work in Section II. In Section III, we formulate the joint optimization problem for VoD and show that it is NP-hard. We present LP-SR and its implementation framework in Section IV. In Section V, we present illustrative simulation results on the performance of LP-SR. In Section VI, we discuss how to further reduce the run-time complexity of LP-SR by movie grouping. We conclude in Section VII.

II. Related Work

We briefly discuss previous work as follows. The joint optimization problem of movie storage and retrieval is generally regarded as NP-hard [7]–[9]. As a result, many heuristics have been proposed to address it (e.g., [7]–[15]). It is often not clear how these heuristics relate to the optimum. In contrast, the proposed LP-SR has a much stronger optimality property: it is asymptotically optimal in k and can be designed to be arbitrarily close to the optimum by increasing k (with trade-off in computational overhead in finding the optimal solution). We show that even under realistic condition that k is far from large (e.g., 5 to 10), the performance is already very close to the optimum. Due to its highly optimal nature, LP-SR outperforms many state-of-the-art and traditional schemes by a wide margin of multiple times. In contrast with some previous algorithms based on iterations [12], [16], LP-SR is based on LP formulations and hence is highly efficient and guarantees to converge.

The work in [4], [6] considers how to support user interactivity through efficiently searching for movie segments. While the heuristics are strong and impressive, they have not considered cost optimization issue. For the work studying the cost issue of VoD [9], [10], [17]–[22], many of them have not comprehensively considered the cost components in the system. Our model captures various costs with general functions on network bandwidth, server storage *and* server uploading bandwidth. Therefore, our model and formulation is a more complete, realistic and practical consideration of a VoD network. There has not been previous study on VoD with such a general and comprehensive consideration on its deployment cost.

While some previous work presents impressive measurement work on commercially deployed VoD systems [23], [24], we address the *optimization* issue of a VoD network and propose a new scheme which is asymptotically optimal. Our optimality is shown to be significantly better than the state of the arts. The recent VoD work in [25]–[27] seeks to maximize the sharing of peers to offload the server load. While the objective of these works is to utilize the uploading bandwidth as much as possible, ours is to minimize the *deployment cost* of VoD. The difference in objectives leads to different system design and operation.

A preliminary version of this work has been reported in [28]. The current work extends it substantially by studying and proving the complexity of the problem, and proposing a movie grouping algorithm which greatly reduces the time complexity of the algorithm for large movie pool. We also present significantly more simulation results to validate the much stronger performance of LP-SR.

III. Problem Formulation and Its Complexity

In this section, we present the joint problem of movie storage and retrieval to minimize deployment cost and show that it is NP-hard. We show the important symbols used in Table I.

The overlay network is modeled as an undirected graph $G = (V, E)$, where V is the set of servers and repository and $E = V \times V$ is the set of overlay edges connecting nodes in V (the extension to directed graph is straightforward given our current

TABLE I
MAJOR SYMBOLS USED IN THIS PAPER

Notation	Definition
$G(V,E)$	A undirected graph representing the overlay network topology
V	The set of servers (repository and distributed proxy servers)
$\|V\|$	The number of servers
M	The set of movies
$\|M\|$	The number of movies
$L^{(m)}$	The movie length (in seconds)
$p^{(m)}$	Access probability of movie m
$I_v^{(m)}$	0 or 1 variable indicating whether server v stores movie m
B_v	Storage space of server v (in seconds)
$r_{uv}^{(m)}$	The probability of streaming movie m from server u to server v
λ_v	Request arrival rate at server v (requests per second)
$\alpha_m L^{(m)}$	Average holding (viewing) time of movie m $\alpha^{(m)} \geq 0$
Γ_{uv}	Network transmission bandwidth from server u to v (bits/s)
b	Movie streaming rate (bits/s)
R_v	Total uploading bandwidth of server v (bits/s)
C_v^S	Cost of server v (per second)
C^S	Aggregated server cost (per second)
C_{uv}^N	Network cost due to directed traffic flow from two servers u and v (per second)
C^N	Total network cost (per second)
C	Total deployment cost ($= C^S + C^N$)

formulation). Let M be the set of movies and $L^{(m)}$ be the movie length (in seconds). Let $p^{(m)}$ be the popularity of movie m, which is the probability that a user requests movie m, where $0 \leq p^{(m)} \leq 1$ and $\sum_{m\in M} p^{(m)} = 1$.

A server v has a certain storage space B_v (in seconds). Let

$$I_v^{(m)} \in \{0,1\}, \quad \forall v \in V, m \in M, \tag{1}$$

indicate whether server v stores movie m. Note that for the repository, $I_v^{(m)} = 1, \quad \forall m \in M$. We obviously must have

$$\sum_{m\in M} I_v^{(m)} L^{(m)} \leq B_v, \quad \forall v \in V. \tag{2}$$

Let

$$0 \leq r_{uv}^{(m)} \leq 1, \quad \forall u, v \in V, m \in M, \tag{3}$$

be the probability of supplying movie m from server u to server v. As the server cannot supply more than that it stores, we must have

$$r_{uv}^{(m)} \leq I_u^{(m)}, \quad \forall u, v \in V, m \in M. \tag{4}$$

In other words, when there is a miss, a remote server can only supply the movie it locally stores. And by definition, $r_{vv}^{(m)} = I_v^{(m)}$.

Each user retrieves data from the servers (including his home server), we hence must have

$$\sum_{u\in V} r_{uv}^{(m)} = 1, \quad \forall v \in V, m \in M. \tag{5}$$

Let λ_v be the total movie request rate at server v (requests per second); the request rate for movie m at server v is hence $p^{(m)}\lambda_v$.

Further let $\alpha^{(m)} L^{(m)}$ be the *average* holding (or viewing) time for movie m, where $\alpha^{(m)} \geq 0$. Note that we have considered user interactivity on movie segments through $\alpha^{(m)}$, which may be different for different movies (interesting movies may have $\alpha^{(m)} > 1$, or *vice versa*). Furthermore, we are interested in *average* holding time, as we are considering time-averaged cost at steady state (hence the distribution of the holding time may be different for different movies). While interacting with the movie, a user holds up a stream and may uniformly visit any segments of the movie over time. Given the above, the average amount of data streamed from server u to v for movie m is hence $r_{uv}^{(m)} \alpha^{(m)} L^{(m)}$.

Let b be the movie streaming bitrate (bits/s). Hence, the data rate the server v "pulls" from server u for movie m is $p^{(m)}\lambda_v \alpha^{(m)} r_{uv}^{(m)} L^{(m)} b$. Therefore, the total network transmission bandwidth (bits/s) from server u to v is

$$\Gamma_{uv} = \sum_{m\in M} p^{(m)}\lambda_v \alpha^{(m)} r_{uv}^{(m)} L^{(m)} b, \quad \forall u, v \in V, \tag{6}$$

for $u \neq v$, and, by definition, $\Gamma_{uu} = 0$.

We consider a general network cost model for the traffic *between* servers serving the misses. Let C_{uv}^N be a monotonically non-decreasing piece-wise linear function of network cost due to the *directed* traffic flow from server u to v, i.e.,

$$C_{uv}^N = \mathbb{C}_{uv}^{\mathbb{N}}(\Gamma_{uv}), \quad \forall u, v \in V, \tag{7}$$

with $C_{uu}^N = 0$. Note that our model is general as C_{uv}^N does not have to be the same as C_{vu}^N. The total network cost C^N is hence

$$C^N = \sum_{u,v\in V} C_{uv}^N. \tag{8}$$

The bandwidth used in a server to serve the other remote servers depends on where to store and how to retrieve a movie. For any server $v \in V$, the total uploading rate (bits/s) that it serves other servers is given by

$$R_v = \sum_{u\in V, u\neq v} \Gamma_{vu}, \quad \forall v \in V. \tag{9}$$

The servers help each other using "cache and stream" model, i.e., a remote server streams to a user *through* his home server. Therefore, the total bandwidth of server v to serve its local users is given by $\sum_{m\in M} p^{(m)}\lambda_v \alpha^{(m)} L^{(m)} b$. This is a fixed quantity given local traffic, and hence will not be considered in our cost optimization.

While network cost depends on the traffic between *pairs* of servers, the cost of a server depends on its total storage and uploading rate used (in order to serve the other servers in the network). Such storage and rate are limited by its disk capacities independent of other servers. Let C_v^S be the cost of operating the server v, which is a monotonically non-decreasing

piece-wise linear function in B_v (storage) and R_v (uploading bandwidth), i.e.,

$$C_v^S = \mathbb{C}_v^{\$}(B_v, R_v), \quad \forall v \in V. \tag{10}$$

In another words, the server cost is a function of its storage and streaming bandwidth. The aggregated server cost C^S is hence

$$C^S = \sum_{v \in V} C_v^S. \tag{11}$$

Therefore, the total system deployment cost C is

$$C = C^N + C^S. \tag{12}$$

We state our joint cost-optimization problem as follows:

1) JOSR: Joint Optimization On Movie Storage and Retrieval Problem to Minimize Deployment Cost: Given topology G, user demand $\{\lambda_v\}$, storage capacity $\{B_v\}$, movie popularity $\{p^{(m)}\}$, and cost functions $\{\mathbb{C}_{uv}^{\mathbb{N}}\}$ and $\{\mathbb{C}_v^{\$}\}$, we seek to minimize the total cost given by (12), subject to (1) to (5). The output is movie storage in each server (i.e., $\{I_v^{(m)}\}$) and movie retrieval between servers (i.e., $\{r_{uv}^{(m)}\}$).

2) Claim: The JOSR problem is NP-hard.

Proof: We prove its NP-hardness by deriving a polynomial reduction from the Domatic Number Problem, whose NP-complete version is stated as follows. Given $G = (V, E)$ and a positive integer K no larger than $|V|$, is the domatic number of G at least K, i.e., can V be partitioned into at least K disjoint dominating sets?

Given G and K, we construct an instance of our decision problem as follows. The network contains K equal-sized movies with equal popularity everywhere, with each server being able to store only one movie. Consider the case that movie length is homogeneous, average viewing time of a movie is the entire movie, request demand at each server is $D/|V|$, unit storage cost is zero and unit access cost of "pulling" a movie from a remote server (including the streaming cost of the remote server and the network cost from remote to the home server) is i units ($i \in Z^+$). Note that such instance construction can obviously be done in polynomial time. Our decision problem hence becomes: Given the instance, is there a joint SR strategy which achieves total cost of at most $D \times (K-1)/K$?

Clearly, at each node v, the minimum average access cost is $(K-1)/K$ when one movie is stored locally and $(K-1)$ movies are in the remote servers with 1-unit access cost, and the total cost collected from all the nodes is correspondingly $D \times (K-1)/K$. As the overlay VoD network in consideration is a connected graph, we construct its subgraph G' which maintains all the servers (i.e., nodes) and the edges with 1-unit access cost. This can also be done in polynomial time. We show that G' can be partitioned into K disjoint dominating sets if and only if there is such a strategy.

First, if there is such a strategy, any node v can access a movie within 1-unit access cost. Then if we could separate V into K disjoint sets with each corresponding to a certain movie, it would result in K dominating sets because the distance between a set and any node is either zero (i.e., contained in the set) or one unit (i.e., connected by an edge). Furthermore, if we have K disjoint dominating sets, we can easily derive such a strategy by assigning each set a distinct movie to store. Therefore, we reduce the Domatic Number Problem to our decision problem which proves that our optimization problem is NP-hard.

IV. LP-SR: LP-Based Segment Storage and Retrieval

In this section, we present our novel and efficient heuristic called LP-SR, which decomposes the optimization problem into two LPs for segment storage (LP-S) and retrieval (LP-R). LP-SR works as follows: we first relax the problem stated above to an LP which yields optimal movie storage (Section IV-A). Based on the LP solution, we then discretize segment storage (Section IV-B). Our discretization process for segment storage is shown to be asymptotically optimal. Given the storage, we solve the optimal segment retrieval problem by another LP (Section IV-C). We analyze the run-time complexity of LP-SR in Section IV-D. Finally we discuss the implementation framework of LP-SR in Section IV-E.

A. LP-S: Relaxation to a Linear Program for Movie Storage

In order to address the NP-hard problem, we relax the constraint in (1) as

$$0 \le I_v^{(m)} \le 1, \quad \forall v \in V, m \in M. \tag{13}$$

After such relaxations, it is clear that our problem contains only linear constraints of real numbers. Therefore, it becomes an LP, and its solution $I_v^{(m)}$ refers to the fraction of movie m that server v stores.

Note that for any arbitrary linear functions of C_{uv}^N ((10)) and C_v^S ((7)) the relaxed problem becomes a linear programming (LP) problem which can be solved efficiently.

B. Asymptotically Optimal Segment Storage and Placement

The LP solution above leads to optimal fraction of movies stored in each server (and hence called LP-S). For ease of replication and access, in reality each movie is partitioned into integral parts of k segments ($k \ge 1$). The major issue is then how to *discretize* the LP-S solution to obtain the number of and which movie segments to store in each server.

We present here an asymptotically optimal segment storage algorithm, i.e., as k increases, our algorithm approaches the JOSR exact solution (we show later in our simulation that k does not need to be large to achieve closely optimum). In order to do that, we place some of the k segments of a movie so as to match as closely as possible the optimal movie storage $I_v^{(m)}$ obtained from LP-S. The major issues are then how many segments of a movie should be stored (i.e., segment space allocation) and which of these segments should be stored (i.e., segment placement) at each server.

1) *Segment space allocation*: Based on LP-S solution, the optimal number of segments of movie m that server v should store is

$$n_v^{(m)} = I_v^{(m)} k, \quad \forall v \in V, m \in M. \tag{14}$$

For integral and finite k, $\{n_v^{(m)}\}$ needs to be discretized to integral values. We present below a simple discretization

TABLE II
SYMBOLS FOR SEGMENT RETRIEVAL

Notation	Definition
$I_v^{(ms)}$	0 or 1 variable indicating whether server v stores segment s of movie m
$r_{uv}^{(ms)}$	The probability of streaming segment s of movie m from server u to server v.
S	The set of segment indices of a movie, i.e. $\{1, 2, ..., k\}$
λ'_v	Request rate of segments at server v (req./sec.)
$p^{(ms)}$	Access probability of segment s of movie m
$L^{(ms)}$	Length of segment s of movie m (seconds)

approach where each server tries to match the optimal LP-S solution as much as possible through integer rounding.

We first round *down* the result $\{n_v^{(m)}\}$ as obtained in (14) to its closest integers $\{\lfloor n_v^{(m)} \rfloor\}$. For server v, it first allocates $\lfloor n_v^{(m)} \rfloor$ segment space according to these integers for movie m. This clearly does not violate its storage constraint (given in (2)).

Note that after the above process, the "unmatched" portion (in minutes) of movie m is given by

$$\left(n_v^{(m)} - \lfloor n_v^{(m)} \rfloor\right) \frac{L^{(m)}}{k}. \tag{15}$$

For the residual storage server v then allocates space in decreasing order of the unmatched portion until its total storage is exhausted.

It is clear from above that, with our discretization process, we can come arbitrarily close to the exact optimality of JOSR solution by increasing k, i.e., *asymptotically* approaches the optimal solution. This is because the rounding effect diminishes with k.

2) *Segment placement*: It is clear that after the above segment space allocation, each server stores integral number of movie segments. Each server then needs to *place* some of k segments to store. The guiding principle of our placement algorithm is that all the segments of a movie should has similar number of replicates in the whole network. Accordingly, we use *rarest first* in segment placement. Specifically, when a server makes a segment placement for a movie, it selects the segment which is the least globally stored until the space budget of the movie is fully consumed.

C. LP-R: Optimal Segment Retrieval as a Linear Program

The optimal solution of $\{r_{uv}^{(m)}\}$ given by LP-S is no longer appropriate due to our segment storage. We hence need to formulate another LP (called LP-R) to derive optimal segment retrieval given segment storage above. The major variables used for segment retrieval is shown in Table II.

Let $S = \{1, 2, \ldots, k\}$ be the set of segment indices of any movie. Let $I_v^{(ms)} \in \{0, 1\}$ indicates whether server v stores segment s of movie m, which has been derived the solution given in Section IV-A. We further let $r_{uv}^{(ms)}$ be the probability of requesting server u from server v segment s of movie m. The segment retrieval problem can then be stated as follows:

- *Arrival rate*: A request for a movie leads to streaming of all its k segments. Therefore, the request rate for *segments* at server v, given movie request rate λ_v, is

$$\lambda'_v = k\lambda_v, \quad \forall v \in V. \tag{16}$$

- *Length*: A movie is equally divided into k segments; hence we have

$$L^{(ms)} = \frac{L^{(m)}}{k}, \quad \forall m \in M, s \in S. \tag{17}$$

- *Popularity*: The popularity of the segments of the same movie is given by

$$p^{(ms)} = \frac{p^{(m)}}{k}, \quad \forall m \in M, s \in S. \tag{18}$$

Using the above, we can formulate optimal segment retrieval problem as a linear program (LP-R), i.e.,

$$\min \left(\sum_{u,v \in V} \mathbb{C}_{uv}^{\mathbb{N}}(\Gamma_{uv}) + \sum_{v \in V} \mathbb{C}_v^{\$}(B_v, R_v) \right),$$

subject to

$$0 \le r_{uv}^{(ms)} \le I_v^{(ms)}, \quad \forall u, v \in V, m \in M, s \in S,$$

$$\sum_{u \in V} r_{uv}^{(ms)} = 1, \quad \forall v \in V, m \in M, s \in S.$$

D. Run-Time Complexity of LP-SR

To solve LP-S and LP-R, we may use CVX which implements the wide-region centering-predictor-corrector algorithm (an interior-point method) to solve this problem [29], [30]. It has been proven that it has $O(\sqrt{N})$ worst-case iteration bound and $O(N^3)$ overall time complexity, where N is the number of variables.

Note that LP-S and LP-R has $O\left(|V|^2|M|\right)$ and $O\left(|V|^2|M|k\right)$ variables, respectively. Therefore, the time complexity of the two LPs is $O\left(|V|^6|M|^3\right)$ and $O\left(|V|^6|M|^3k^3\right)$, respectively. Furthermore, the time complexity of segment storage is $O\left(|V||M|k\right)$, because all the servers, movies and their constituent segments are traversed to determine which server to store which segment.

Given the above, the overall time complexity of LP-SR is $O\left(|V|^6|M|^3k^3\right)$. Note that even though the run-time complexity of LP-SR is related to k, low k (say 5) already achieves closely optimal solution.

E. Implementation Framework

We show in Fig. 2 an implementation framework of LP-SR run in the central server (as there is no need to implement any decision algorithms in the proxy servers). Given the system parameters on the movies, servers, and network, the central sever can solve the LP-S problem as presented in Section IV-A. Based on its solution, it further calculates the required segment space and segment placement at each proxy server as presented in Section IV-B. The servers store the the computed movie segments according to this decision (given by I_v^{ms}). Given such storage decision, the retrieval decision can be made by solving

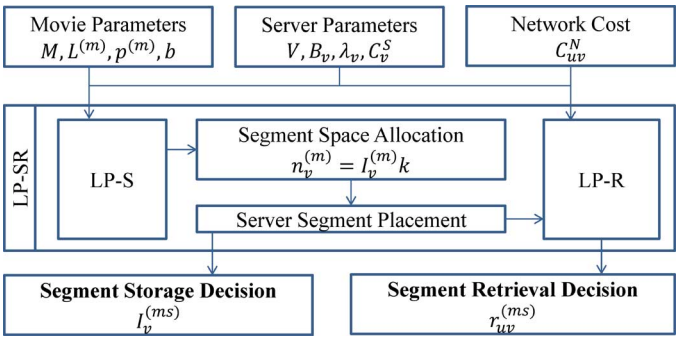


Fig. 2. An implementation framework of LP-SR at the central server.

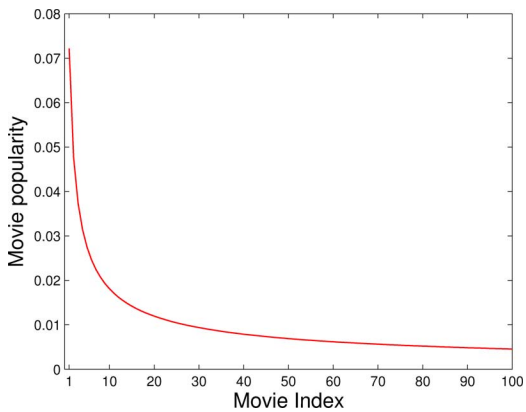


Fig. 3. Movie popularity with Zipf parameter 0.6 for 100 movies.

LP-R as presented in Section IV-C. The servers then retrieve their missing segments accordingly (given by r_{uv}^{ms}).

V. Illustrative Simulation Results

In this section, we first present our simulation environment and performance metrics to study the performance of LP-SR, followed by illustrative results at steady state.

A. Setup and Performance Metrics

LP-SR can be applied to any movie popularity. For concreteness in our simulation, we consider that movie popularity follows the Zipf distribution with parameter s, i.e., the request probability of the ith movie is proportional to $1/i^s$. Fig. 3 shows movie popularity given our baseline parameters of $s = 0.6$ and $m = 100$ movies, where the top 30% of the movies account for close to 60% (56.72%) of the traffic. In a Zipf distribution, movie popularity becomes more "skewed" as s increases; therefore in this paper, we sometimes call the Zipf parameter s the "skewness" of the distribution.

In our simulation, we consider that requests arrive at each proxy server according to a Poisson process with total rate λ (req./second). It is clear from Section IV that the optimization of LP-SR does not depend on the specific request process at server v but its arrival rate λ_v. Therefore, the results and conclusions may be extended to any request processes or traces so long as they share the same request rate.

The central server has no home users. The proxy servers have heterogeneous storage space and bandwidth following a Zipf distribution (independent of each other). The repository stores all the movies with a streaming capacity twice of the average streaming capacity of the proxy servers. Unless otherwise stated, we use the default values as shown in Table III for our system parameters (the baseline case).

TABLE III
Baseline Parameters Used in Our Study

Parameter	Default value
k	5
Number of proxy servers	10
Number of movies	100
Average server storage	10 movies
Zipf parameter (Skewness) of server storage	0.4
Average proxy server bandwidth capacity	160 Mbits/s
Zipf parameter (Skewness) of server bandwidth	0 (i.e., same bandwidth)
Zipf parameter (Skewness) of movie popularity	0.6
Movie length	90 minutes
Average movie holding time	Movie length (i.e., $\alpha^{(m)} = 1$)
Movie streaming rate	1 Mbits/s
Total request rate in the network	0.3 req./s (equally distributed to the proxies)
c_{uv} between central and proxy server	0.01 unit/s
c_{uv} between proxies	Zipf with parameter 0.6 and mean 0.005 unit/s

In the simulation, we consider the network cost function from server u to server v to be proportional to the bandwidth between them, i.e.,

$$C_{uv}^N(\Gamma_{uv}) = c_{uv}\Gamma_{uv}, \qquad \forall u, v \in V, \tag{19}$$

where c_{uv} is some constant (by definition, $c_{vv} = 0$).

The server cost is a function of its storage and its total upload bandwidth used to serve the remote servers, modeled as

$$C_v^S = \sigma_B B_v + C_v(R_v), \qquad \forall v \in V, \tag{20}$$

where σ_B is a constant ($\sigma_B = 10^{-5}$ in our baseline, and $C_v(R_v)$ is a piece-wise linear function monotonically increasing in R_v.

As $C_v(R_v)$ is limited by the bandwidth capacity of the server, we use a cost model inspired by queuing theory due to server congestion [31], [32]. We show in Fig. 4 streaming cost $C_v(R_v)$ versus R_v/U_v used in our simulation, where U_v is the maximum uploading capacity of the server and hence R_v/U_v is the bandwidth utilization of the server. The cost increases with the bandwidth utilization at the server. There are three linear segments formed by points (0, 0), (0.8, 0.125), (0.93, 0.4375) and (0.99, 1.925) (these coordinates can be easily verified using the queuing delay model $\sigma_S/(U_v - R_v)$, where σ_S is some constant). As the consumed bandwidth R_v approaches the bandwidth capacity U_v, the server cost increases sharply. (Note that although for concreteness we have used a cost model of a queuing function, LP-SR is by no means limited to that; any other monotonic cost function may be equally used.)

The performance metrics we are interested in are:

- *Total cost* (unit/s), which is the sum of server cost and network cost according to (12). This is the total deployment cost of the network.

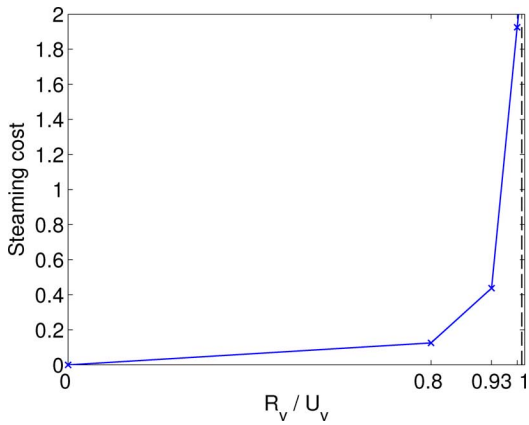


Fig. 4. Streaming cost model at a proxy server.

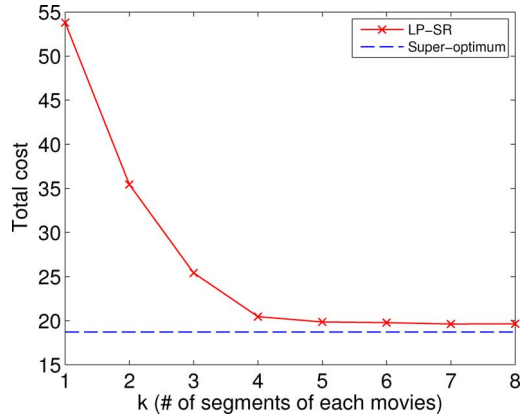


Fig. 5. Total cost versus k in LP-SR.

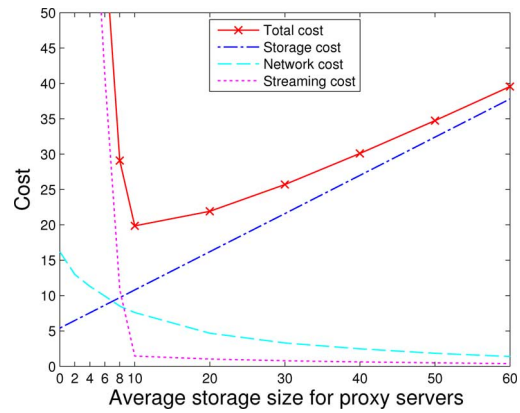


Fig. 6. Cost versus average proxy storage.

- *Server cost* (unit/s), which is the sum of its storage and streaming defined in (11) and (20). We further examine the following cost components:
  - — *Storage cost*, which is the total cost due to server storage; and
  - — *Streaming cost*, which is the server bandwidth cost to support other servers.
- *Network cost* (unit/s), which is network transmission cost defined by (8) and (19).
- *Cost of each movie* (unit/s), which is the average cost to access movie m by any user. It is the cost to provide on-demand streaming service of the movie to users, due to its use of network bandwidth, server storage and server uploading bandwidth.

We compare LP-SR with the following traditional and state-of-the-art movie replication schemes:

- *Random*, where each server randomly stores movies without considering their popularity. This is a simple storage strategy.
- *MPF (Most Popular First)*, where each server stores the most popular movies. This is a greedy strategy, but does not take advantage of cooperative replication.
- *Local Greedy [10]*, which divides the movies into three categories, those popular ones which all servers store (full replication), those medium popular ones which only one proxy server store (single copy), and those unpopular ones which only the repository stores (no copy). By formulating a LP problem, it seeks to minimize network cost. As Local Greedy assumes homogeneous access cost, we set its access cost to be equal to the average access cost between servers in our network.
- *Super-optimum*, which is the direct LP solution on Page 4 by simply relaxing the integral constraints from (1) to (13) of JOSR. Due to such relaxation on $I_v^{(m)}$, its solution of minimum cost, denoted as C^{LP^*}, must be lower than that of the original NP-hard solution denoted as C^{NP^*}. In other words, $C^{LP^*} \leq C^{NP^*}$. We call C^{LP^*} the *super-optimum* solution because it is no worse than the NP-hard solution. Note that LP-SR solution must be no better than the NP-hard solution, i.e., $C^{NP^*} \leq C^{LP-SR^*}$ We will see later in the section that the LP-SR solution C^{LP-SR^*} is very close to and asymptotically approaches the super-optimum C^{LP^*}, meaning that LP-SR is close to the NP-hard solution.

In all the comparison schemes, upon a miss request, the home server v chooses an available server u which has the requested content with a probability proportional to $1/c_{uv}$. It is a reasonable, simple and effective strategy because the server with lower access cost has higher chance to be chosen. With this probabilistic approach, a server with low access cost is not always selected so as to avoid congestion, and hence high network cost, at the server.

B. Illustrative Results

We plot in Fig. 5 the total cost versus k in LP-SR. As k increases, the network approaches the super-optimal case given by relaxing the integer constraints on movie storage $\{I_v^m\}$. (Note that the curve optimum lies between the LP-SR and the super-optimum.) However, for humble value of k, the performance is already very close to the optimum (less than 6.5% deviation in our default setting). This shows that our network is highly efficient, with closely optimal performance even for the practical finite value of k.

We show in Fig. 6 the cost components and total cost versus the average storage space for servers. The total cost falls off initially but rises up again, showing a minimum. At the beginning

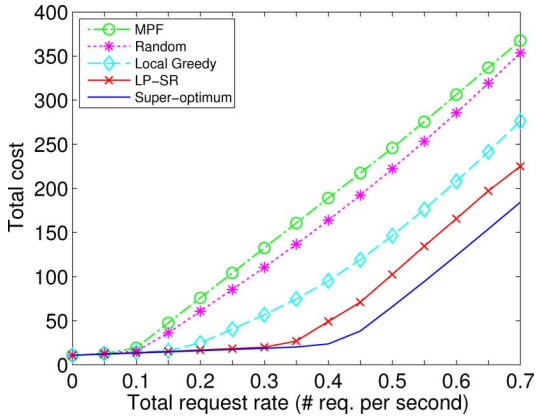


Fig. 7. Total cost versus request rate given different schemes.

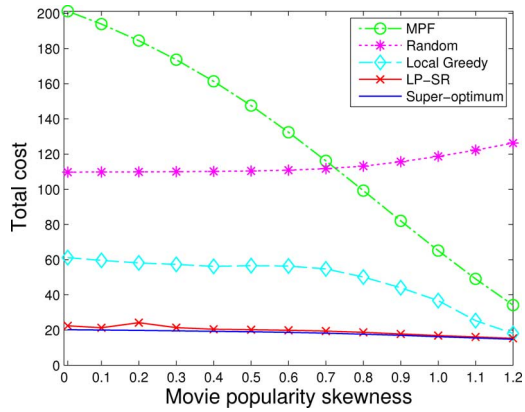


Fig. 9. Total cost versus the Zipf parameter (skewness) of movie popularity given different schemes.

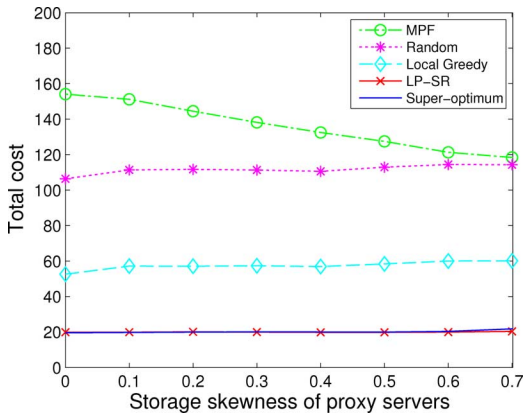


Fig. 8. Total cost versus the Zipf parameter (skewness) of storage capacity given different schemes.

when the proxy servers have little storage, all the traffic concentrates on the repository, leading to high overall streaming cost. As proxy storage increases, the repository load is reduced and hence the streaming and network transmission cost. As storage further increases, storage cost becomes a major component. It is clear that LP-SR can balance the cost between storage and bandwidth and achieve its optimality by provisioning optimal network resources.

We show in Fig. 7 the total cost versus the request rate given different schemes. Total cost grows with request rate mainly due to the increase in network traffic. LP-SR clearly achieves cost close to the optimum and much lower than any of the other schemes. In other words, given the same deployment budget, LP-SR can support much higher request rate (i.e., more concurrent users in the system). MPF does not perform well because it mainly relies on the central server to serve the requests for the unpopular movies. Random, due to its popularity-blind nature, stores insufficient copy of the popular movies, leading to considerable cost. Local Greedy has lower cost due to its network cost optimization. LP-SR achieves by far the best performance because it achieves near optimality by capturing not only the network transmission cost but also the server storage and streaming cost.

We show in Fig. 8 the total cost versus the Zipf parameter (skewness) of the storage capacity of proxies given different schemes. The total cost in general increases with storage skewness. It is because skewed storage means that many requests have to be indirectly served by the remote servers of large storage space. This consumes much of server bandwidth, and hence the streaming cost dramatically increases. Having a substantially lower and closely-optimal cost with a much slower growth rate in cost, LP-SR is much more scalable and can make good movie placement by best utilizing server storage. MPF decreases with the Zipf parameter of storage because the proxies of large storage share some load from the repository, hence reducing the streaming cost of the repository.

We plot in Fig. 9 the total cost versus the Zipf parameter (skewness) of movie popularity given different schemes. The total cost in general decreases with the popularity skewness. This is because skewed popularity means that more requests are concentrated on fewer popular movies. Consequently, there is lower miss rate, leading to lower streaming and network cost. LP-SR achieves substantially the lowest cost (which is closely optimal with little difference), even for low Zipf parameter, i.e., when the popularity is quite uniform. This shows that LP-SR makes virtually optimal movie placement and retrieval decisions. Local Greedy performs better than MPF because it takes network cost into consideration. The cost of Random increases with Zipf parameter because it is popularity-blind. As a result, the popular movies, because of their copies not increasing with their popularity, suffer from high streaming and network cost.

We plot in Fig. 10 the total cost versus the Zipf parameter (skewness) of proxy bandwidth given different schemes. In general, the total cost increases with the Zipf parameter because skewed bandwidth means that the servers with low bandwidth but large storage cannot support much remote streaming to other servers. This defeats the advantages on its locally stored movies. Furthermore, the servers with lower bandwidth more easily run out of bandwidth, leading to a sharp increase in streaming cost. LP-SR clearly has the lowest cost with little difference from the optimum, beating the other schemes by multiple times. It is also robust to system heterogeneity, and fully utilizes the storage and bandwidth resource for cooperative streaming.

We show in Fig. 11 the total cost versus total number of movies for different schemes. The cost increases with the movie size because of additional network load to stream

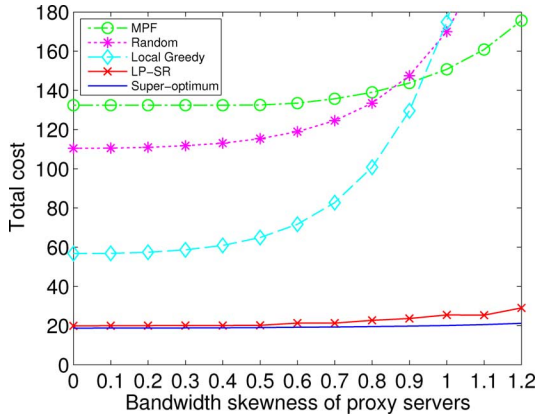


Fig. 10. Total cost versus the Zipf parameter (skewness) of bandwidth capacity given different schemes.

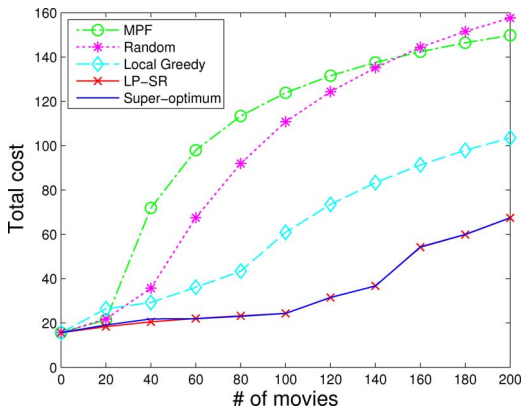


Fig. 11. Total cost versus the movie size given different schemes.

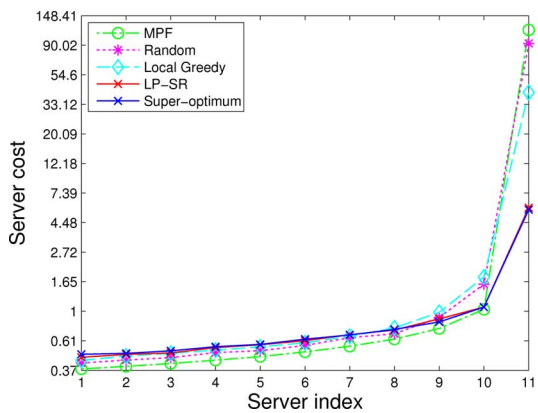


Fig. 12. Server cost distribution given different schemes.

movies. LP-SR enjoys substantially the lowest cost as it makes closely optimal decisions on movie storage and retrieval. MPF suffers from high cost because the repository needs to stream those unpopular movies, leading to high streaming cost.

We compare in Fig. 12 the server cost for different schemes. We sort the proxy servers according to their storage in ascending order (as their streaming capacity is the same in our baseline), and the last server is the repository. It is clear that LP-SR achieves costs of little difference from the optimum. Because of that, it utilizes the best the storage and bandwidth resources of proxy servers, leading to the lowest repository streaming

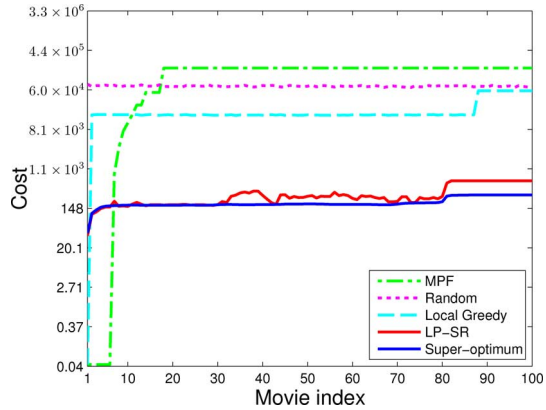


Fig. 13. Movie cost for different schemes.

cost as compared with other schemes. All the other schemes suffer from high repository cost (note that log scale) due to misses in the proxies. The figure shows that LP-SR has strong server cooperation to achieve near-optimal performance. As MPF only stores the most popular movies at the proxy servers, it has lower proxy server cost but much higher repository cost (due to miss traffic). In MPF, the proxies barely contribute their bandwidth and storage to help each other. Local Greedy, with network cost optimization, outperforms Random in both proxy server cost and repository cost.

We compare in Fig. 13 the cost to access a movie for different schemes. The movies are sorted according to their descending popularity. The popularity-based schemes (i.e., LP-SR, Local Greedy and MPF) tend to locally store the popular movies, and hence those movies enjoy lower cost. LP-SR makes much better decision by cooperatively storing the movies. LP-SR accomplishes much better optimality with a rather uniform movie cost. Its cost of medium to unpopular movies is strikingly much lower by orders of magnitude than the other schemes. Also shown is the movie cost for the super-optimal case. LP-SR clearly achieves performance of little difference from the optimum. For MPF, the costs of popular movies are negligible at steep sacrifice of less popular ones. Random treats each movie equally and thus has the most uniform cost distribution. The figure shows that LP-SR makes intelligent decisions on movie storage and retrieval to achieve low deployment cost.

We finally study the impact of popularity mis-estimation on the performance of the VoD. We consider a system optimally designed with the Zipf parameter $s = 0.6$, the estimated (or assumed) parameter for movie popularity. We show in Fig. 14 what the system cost is for different *actual* Zipf parameter (the movie ranks remain the same). Also shown in solid line is the case if the assumed parameter exactly matches with the actual one. Obviously, at $s = 0.6$, the two curves match. If the actual movie popularity is less skewed (lower s), the cost increases quite sharply because of the increase in access traffic. On the other hand, if the actual popularity is more skewed (higher s), the cost penalty due to such mis-estimation is not as high. This is because the servers storing those popular movies have lower miss rate. If the actual Zipf parameter further increases, the cost starts to increase again because those servers not storing those popular movies need to retrieve them from others, incurring

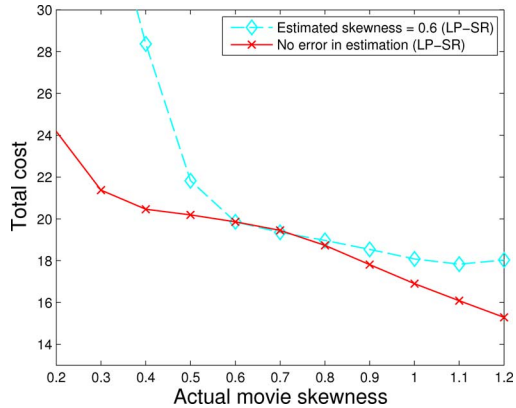


Fig. 14. Cost performance against actual Zipf parameter for movie popularity given that the system is designed with $s = 0.6$.

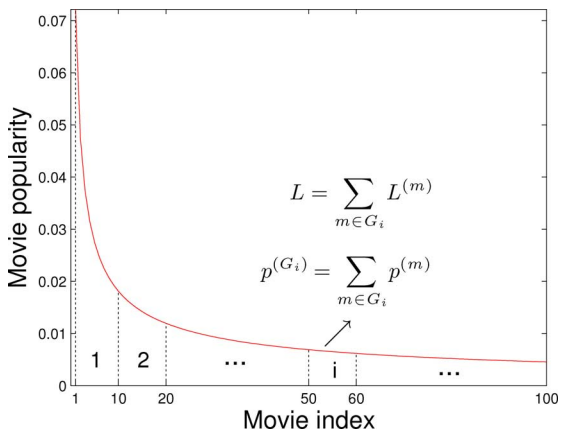


Fig. 15. Efficient computation method on movie grouping in the default baseline.

much network and server bandwidth costs. The result shows that in estimating Zipf parameter, over-estimation has higher impact than under-estimation (the performance is more sensitive to over-estimation). The estimation does not have to be very accurate in order to achieve closely optimal performance.

VI. Efficient Computation for Large Movie Pool

In terms of the number of movies $|M|$, the run-time complexity of LP-SR is $O(|M|^3)$ (Section IV-D). As the number of movies increases, we need to compute the solution more efficiently.

In this section, we propose an efficient computation based on movie grouping. The movies are divided into groups of size g (in seconds). Our algorithm achieves a polynomial factor $O(g^3)$ of reduction in complexity without sacrificing much on performance. We then illustrate its performance with simulation results.

A. Efficient Movie Grouping

Fig. 15 illustrates the efficient computation method on movie grouping. The basic idea is to divide movies into *groups* such that the complexity of the LP-SR is reduced. We first divide the movies into fixed-size groups to be stored in servers. Let $\mathbb{L}$ be the group size (in seconds). In order to accommodate a group in any server, clearly group size $\mathbb{L}$ must be chosen such that

$$\mathbb{L} \leq \min_{v \in V} B_v. \tag{21}$$

Movies are grouped according to their decreasing popularity. The most popular movies are put into the 1*st* group to make up the size $\mathbb{L}$, fragmenting the last movie and store it across groups if necessary. The remaining movies are then assigned to group 2, 3, and so on. This process is repeated until the all the movies are grouped.

Let $\mathbb{G}$ be the set of groups and G_i be the i^{th} group. The above grouping process leads to the following:

- *Length*: Each group is of equal size $\mathbb{L}$ and satisfies

$$\mathbb{L} = \sum_{m \in G_i} L^{(m)}, \quad \forall G_i \in \mathbb{G}. \tag{22}$$

- *Group popularity*: The group popularity $p^{(G_i)}$ is given by

$$p^{(G_i)} = \sum_{m \in G_i} p^{(m)}, \quad \forall G_i \in \mathbb{G}. \tag{23}$$

  Note that if a movie is fragmented across groups, the popularity of the movie in a group is proportionally adjusted based on its fragment length in the group.

After this movie grouping is done, LP-SR is run on the groups by treating them as $|G|$ "movies" with popularity $p^{(G_i)}$. In the phase of segment storage, a group is partitioned into k *group-segments* and stored and accessed accordingly similar as before. The movie can be efficiently retrieved from a group-segment with a simple table lookup.

B. Time Complexity

The time complexity of movie grouping is clearly $O(|M|)$, as all the movies are traversed to be assigned to different groups. The total run-time complexity on obtaining the solution is hence $O(|V|^6|M|^3k^3/g^3)$, where g is the average group size. In other words, the complexity is reduced by a factor of $O(g^3)$.

C. Illustrative Results

We conduct simulation to study the performance of our grouping algorithm. We use the same baseline parameters as given in Section V.

We plot in Fig. 16 the total cost versus request rate given different group sizes (Note that because the default movie length is 90 minutes, a group size equal to that means no grouping). Total cost rises up with the request rate mainly because the increase of network load. Large group size can reduce the time complexity of LP-SR but increase the performance deviation from the optimum. LP-SR with movie grouping can still achieve closely optimum, outperforming Local Greedy by a wide margin (of multiple times when the request rate is high).

We next examine the total running time to compute the LP-SR solution after the movies are grouped with our algorithm. We show in Fig. 17 the running time of LP-SR given

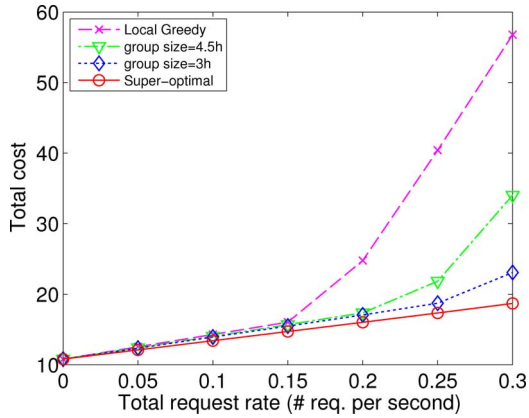


Fig. 16. Total cost versus request rate given different group sizes.

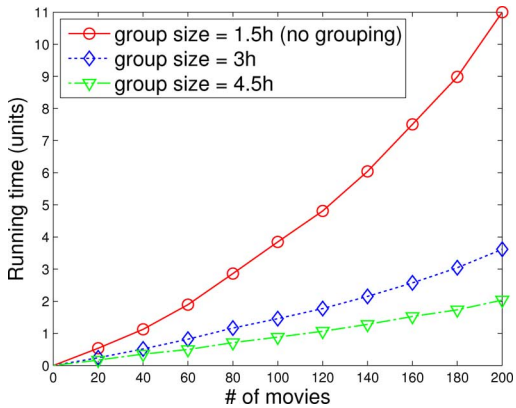


Fig. 17. The total running (computational) time of LP-SR given different group sizes and in no grouping case.

different group sizes.[3] The running time increases with the total number of movies. It is because LP-SR captures the information for every movie, and hence more movies introduces more information and decision variables for LP-SR to process. LP-SR with movie grouping can greatly reduce the time complexity (over triple times in the default baseline setting). The result shows that movie grouping leads to efficient computation for a large movie pool.

We plot in Fig. 18 the tradeoff curve between total cost and running time for movie grouping. We also illustrate the figure the corresponding group size of each tradeoff point. As running time increases, LP-SR achieves better performance. Total cost first decreases sharply and then converges to some value (corresponding to the optimal value of no grouping). It is clear that a good operating point of the system is one with low running time and cost, which is around the "knee" of the around (with group size of 3 to 4.5 hours corresponding to 2 to 3 movies per group respectively).

VII. Conclusion

In this work, we have studied optimal segment storage and retrieval to minimize VoD deployment cost for interactive

[3]As the running time depends on the machine used, we have normalized the time in terms of some unit. For a 64-bit ThinkCentre M90 with Intel i7 CPU 870 2.93 GHz and RAM 4.00 GB, the unit is in minute.

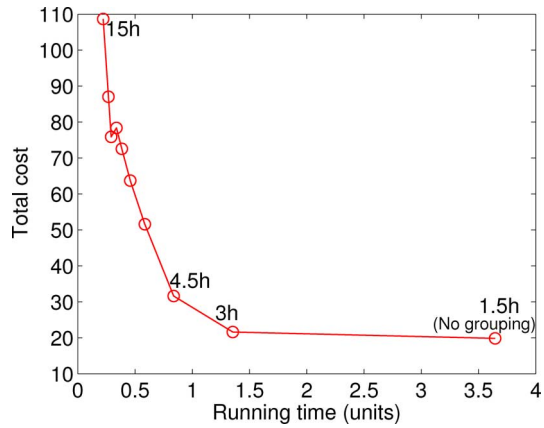


Fig. 18. The tradeoff curve between total cost and running time for movie grouping.

streaming with distributed proxy servers. The deployment cost captures comprehensively the costs of server streaming, server storage and network transmission.

For efficient server storage and retrieval, each movie is partitioned into k segments ($k \geq 1$). We first formulate the joint problem and show that it is NP-hard. To address this, we propose LP-SR, a novel and efficient heuristic which decomposes the problem into two linear programs (LP) for segment storage (LP-S) and retrieval (LP-R), respectively. In marked contrast with much of the previous work where heuristics are often proposed without knowing how they perform with respect to the optimum, our solution is *asymptotically optimal* in k, and k does not need to be large to achieve near optimality (k is around 5). To make our solution more efficient for large movie pool, we propose a movie grouping algorithm which achieves close to optimal performance with polynomial reduction in running time.

We have conducted extensive simulation to compare LP-SR performance with other traditional and state-of-the-art schemes. The results show that LP-SR substantially outperforms the other schemes by a wide margin (multiple times in many cases). LP-SR achieves close optimality with much lower deployment cost.

References

[1] S.-H. G. Chan and F. Tobagi, "Distributed servers architecture for networked video services," *IEEE/ACM Trans. Netw.*, vol. 9, no. 2, pp. 125–136, Apr. 2001.

[2] D. Niu, Z. Liu, B. Li, and S. Zhao, "Demand forecast and performance prediction in peer-assisted on-demand streaming systems," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 421–425.

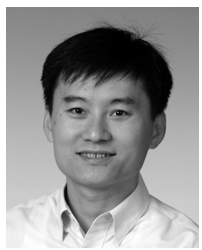
[3] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. K. Ramakrishnan, "Optimal content placement for a large-scale VoD system," in *Proc. 6th Int. Conf. Co-NEXT '10*, New York, NY, USA, 2010, pp. 4:1–4:12.

[4] Y. He, G. Shen, Y. Xiong, and L. Guan, "Optimal prefetching scheme in P2P VoD applications with guided seeks," *IEEE Trans. Multimedia*, vol. 11, no. 1, pp. 138–151, Jan. 2009.

[5] M. Mehyar, D. Spanos, J. Pongsajapan, S. H. Low, and R. M. Murray, "Asynchronous distributed averaging on communication networks," *IEEE/ACM Trans. Netw.*, vol. 15, no. 3, pp. 512–520, Jun. 2007.

[6] W.-P. K. Yiu, X. Jin, and S.-H. G. Chan, "VMesh: Distributed segment storage for peer-to-peer interactive video streaming," *IEEE J. Select. Areas Commun. Special Issue on Advances in Peer-to-Peer Streaming Systems*, vol. 25, no. 9, pp. 1717–31, Dec. 2007.

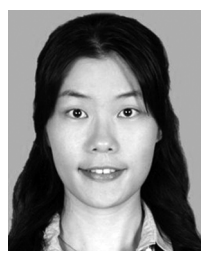
[7] S. Alam, M. Hasan, M. Hossain, and A. S. M. Sohail, "Heuristic solution of MMKP in different distributed admission control and QoS adaptation architectures for video on demand service," in *Proc. 2nd Int. Conf. Broadband Networks (BroadNets)*, Oct. 2005, vol. 2, pp. 896–903.
[8] L. Chang and J. Pan, "Reducing the overhead of view-upload decoupling in peer-to-peer video on-demand systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2011, pp. 1–5.
[9] A. Nimkar, C. Mandal, and C. Reade, "Video placement and disk load balancing algorithm for VoD proxy server," in *Proc. IEEE Int. Conf. Internet Multimedia Services Architecture and Applicat. (IMSAA)*, Dec. 2009, pp. 1–6.
[10] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proc. 2010 IEEE INFOCOM*, Mar. 2010, pp. 1–9.
[11] S. Zaman and D. Grosu, "A distributed algorithm for the replica placement problem," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 9, pp. 1455–1468, Sep. 2011.
[12] J. Kangasharju, K. W. Ross, and D. A. Turner, "Optimizing file availability in peer-to-peer content distribution," in *Proc. 26th IEEE Int. Conf. Comput. Commun. (INFOCOM)*, May 2007, pp. 1973–1981.
[13] S. Annapureddy, S. Guha, C. Gkantsidis, D. Gunawardena, and P. R. Rodriguez, "Is high-quality voD feasible using P2P swarming?," in *Proc. 16th Int. Conf. World Wide Web (WWW '07)*, New York, NY, USA, 2007, pp. 903–912.
[14] P. R. R. S. Annapureddy, C. Gkantsidis, and L. Massoulie, Providing Video-on-Demand using Peer-to-Peer Networks, 2005, Microsoft Research Tech. Rep., MSR-TR-2005-147.
[15] M. Hefeeda and B. Noorizadeh, "On the benefits of cooperative proxy caching for peer-to-peer traffic," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 7, pp. 998–1010, Jul. 2010.
[16] J. Dai, Z. Hu, B. Li, J. Liu, and B. Li, "Collaborative hierarchical caching with dynamic request routing for massive content distribution," in *Proc. IEEE INFOCOM 2012*, Mar. 2012, pp. 2444–2452.
[17] W. Wu and J. C. S. Lui, "Exploring the optimal replication strategy in P2P-VoD systems: Characterization and evaluation," in *Proc. IEEE INFOCOM 2011*, Apr. 2011, pp. 1206–1214.
[18] Y. R. Choe, D. L. Schuff, J. M. Dyaberi, and V. S. Pai, "Improving VoD server efficiency with bittorrent," in *Proc. 15th Int. Conf. Multimedia (MULTIMEDIA '07)*, New York, NY, USA, 2007, pp. 117–126.
[19] C. Dana, D. Li, D. Harrison, and C. N. Chuah, "BASS: Bittorrent assisted streaming system for video-on-demand," in *Proc. IEEE 7th Workshop Multimedia Signal Process., 2005*, Nov. 2006, pp. 1–4.
[20] J. Lv, X. Cheng, Q. Jiang, J. Ye, T. Zhang, S. Lin, and L. Wang, "LiveBT: Providing video-on-demand streaming service over bittorrent systems," in *Proc. Int. Conf. Parallel and Distributed Comput. Applicat. and Technol.*, 2007, vol. 0, pp. 501–508.
[21] A. Vinay, K. Bharath, P. Saxena, S. Chandra, and T. N. Anitha, "Bandwidth aware load balancing and optimal bandwidth allocation techniques for video-on-demand systems," in *Proc. IEEE Int. Conf. Commun. Control and Comput. Technol. (ICCCCT)*, Oct. 2010, pp. 425–430.
[22] S.-H. G. Chan, "Operation and cost optimization of a distributed servers architecture for on-demand video services," *IEEE Commun. Lett.*, vol. 5, no. 9, pp. 384–386, Sep. 2001.
[23] C. Huang, J. Li, and K. W. Ross, "Can internet video-on-demand be profitable?," in *Proc. 2007 Conf. Applicat., Technol., Architectures, and Protocols for Comput. Commun. (SIGCOMM '07)*, New York, NY, USA, 2007, pp. 133–144.
[24] H. Li, L. Zhong, J. Liu, B. Li, and K. Xu, "Cost-effective partial migration of VoD services to content clouds," in *Proc. 2011 IEEE Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2011, pp. 203–210.
[25] Y. Zhou, T. Fu, and D. M. Chiu, "A unifying model and analysis of P2P VoD replication and scheduling," in *Proc. IEEE INFOCOM 2012*, Mar. 2012, pp. 1530–1538.
[26] W. Wu and J. Lui, "Exploring the optimal replication strategy in P2P-VoD systems: Characterization and evaluation," in *Proc. IEEE INFOCOM 2011*, Apr. 2011, pp. 1206–1214.
[27] Y. Zhou, T. Z. J. Fu, and D. M. Chiu, "On replication algorithm in p2p vod," *IEEE/ACM Trans. Netw.*, vol. PP, no. 99, p. 1, 2012.
[28] Z. Xu and S.-H. G. Chan, "LP-based optimization of storage and retrieval for distributed video-on-demand," in *Proc. Globecom 2012—Commun. Software, Services and Multimedia Symp.*, Dec. 3–7, 2012.
[29] M. Grant and S. Boyd, CVX: Matlab Software for Disciplined Convex Programming, Version 1.21, 2011. [Online]. Available: cvxr.com/cvx.
[30] J. F. Sturm, "Primal-dual interior point approach to semidefinite programming," Ph.D. dissertation, Erasmus Universiteit Rotterdam, Rotterdam, The Netherlands, 1997.
[31] L. Kleinrock, *Queueing Systems, Volume 1: Theory*. New York, NY, USA: Wiley, 1976.
[32] H. Kobayashi, *Modeling and Analysis: An Introduction to System Performance Evaluation Methodology*. Reading, MA, USA: Addison Wesley, 1981.



S.-H. Gary Chan (S'89–M'98–SM'03) is currently Professor at the Department of Computer Science and Engineering, Director of Sino Software Research Institute, and Co-director of Risk Management and Business Intelligence program, The Hong Kong University of Science and Technology (HKUST), Hong Kong. He received MSE and PhD degrees in Electrical Engineering from Stanford University (Stanford, CA) in 1994 and 1999, respectively, with a minor in business administration. He obtained his B.S.E. degree (highest honor) in Electrical Engineering from Princeton University (Princeton, NJ) in 1993, with certificates in Applied and Computational Mathematics, Engineering Physics, and Engineering and Management Systems. His research interest includes multimedia networking, overlay streaming, and wireless communication networks.

Professor Chan was an Associate Editor of IEEE Transactions on Multimedia (2006–11), and is a Vice-Chair of Peer-to-Peer Networking and Communications Technical Sub-Committee of IEEE Comsoc Emerging Technologies Committee. He has been Guest Editors of IEEE Transactions on Multimedia (2011), IEEE Signal Processing Magazine (2011), IEEE Communication Magazine (2007), and Springer Multimedia Tools and Applications (2007). He was the TPC chair of IEEE Consumer Communications and Networking Conference (CCNC) 2010, Multimedia symposium in IEEE Globecom (2007 and 2006), IEEE ICC (2007 and 2005), and Workshop on Advances in Peer-to-Peer Multimedia Streaming in ACM Multimedia Conference (2005).

Professor Chan is the recipient of Google Mobile 2014 Award in 2010 and 2011, and is a member of honor societies Tau Beta Pi, Sigma Xi and Phi Beta Kappa. He has been a visiting professor or researcher in Microsoft Research Asia (2000–11), Princeton University (09), Stanford University (2008–09), and University of California at Davis (1998–1999). He was a William and Leila Fellow at Stanford University (1993–94). At Princeton, he was the 1993 recipient of the Charles Ira Young Memorial Tablet and Medal, and the POEM Newport Award of Excellence.



Zhuolin (Fannie) Xu got her MPhil degree in Computer Science and Engineering from the HKUST and Bachelor degree in Software Engineering from Sun Yat-sen University. Fannie was awarded many prizes during her undergraduate period, such as National Scholarship of China, Excellent Thesis Prize, Outstanding Graduate Prize, etc. Besides, she has participated in many research projects, one of which is called Near-duplicate Video Detection that was awarded excellent undergraduate diploma design.